Manuel PHP

Stig Sæther Bakken

Alexander Aulbach

Egon Schmid

Jim Winstead

Lars Torben Wilson

Rasmus Lerdorf

Zeev Suraski

Andrei Zmievski

Jouni Ahto

Publié par

Damien Seguy

16-08-2001

Manuel PHP

par Stig Sæther Bakken, Alexander Aulbach, Egon Schmid, Jim Winstead, Lars Torben Wilson, Rasmus Lerdorf, Zeev Suraski, Andrei Zmievski, et Jouni Ahto

par

Publié par Damien Seguy

Publié 16-08-2001

Copyright

Table des matières

Préface

PHP, est un acronyme récursif, qui signifie "PHP: Hypertext Preprocessor" : c'es un langage de script HTML, exécuté coté serveur. L'essentiel de sa syntaxe est emprunté aux langages C, Java et Perl, avec des améliorations spécifiques. L'objet de ce langage est de permettre aux développeurs web d'écrire des pages dynamiques rapidement.

Notez qu'aujourd'hui, les capacités de PHP vont bien au-dela de la génération de pages personnelles : PHP génère des pages PDF, des images ou même des animations Flash à la volée. PHP-GTK (http://gtk.php.net/) permet à PHP de faire des scripts utilisant des interfaces X.

A propos de ce manuel

Ce manuel est écrit en XML avec DocBook XML DTD (http://www.nwalsh.com/docbook/xml/), en utilisant DSSSL (http://www.jclark.com/dsssl/) (Document Style and Semantics Specification Language) pour l'affichage. Les outils utilisés pour les formats HTML et TeX sont Jade (http://www.jclark.com/jade/), écrit par James Clark (http://www.jclark.com/bio.htm) et The Modular DocBook Stylesheets (http://nwalsh.com/docbook/dsssl/) écrit par Norman Walsh (http://nwalsh.com/). Nous utilisons aussi Microsoft HTML Help Workshop (http://msdn.microsoft.com/library/en-us/htmlhelp/html/vsconhh1start.asp) pour générer le format HTML.

Vous pouvez télécharger le manuel courant dans divers langages et formats, y compris en texte seul, HTML, PDF, PalmPilot DOC, PalmPilot iSilo et WinHelp, depuis http://www.php.net/docs.php. Les manuels sont mis à jour quotidiennement.

La version française est traduite quotidiennement et disponible chez Nexen (nexen.net/ (http://www.nexen.net/)). Ce manuel a été généré à partir de la documentation originale en anglais du PHP Documentation Group, au format XML, grâce à une version adaptée de texi (http://www.texinfo.com/).

Vous pouvez avoir d'autres informations sur le téléchargement des sources XML de cette documentation à http://cvs.php.net/. La documentation est stockée dans le module `phpdoc`.

Partie I. Comment Commencer

Chapitre 1. Introduction

Qu’est ce que PHP?

PHP (officiellement "PHP: Hypertext Preprocessor") est un langage de script HTML, qui fonctionne coté serveur.

Réponse simple et claire, mais qu’est ce que cela veut dire? Un exemple :

Exemple 1-1. Exemple d’introduction

```
<html>
    <head>
        <title>Exemple</title>
    </head>
    <body>
        <?php
        echo "Bonjour, je suis un script PHP!";
        ?>
    </body>
</html>
```

Il est à noter la différence avec les autres scripts CGI écrit dans d’autres langages tels que le Perl ou le C : Au lieu d’écrire un programme avec de nombreuses lignes de commandes afin d’afficher une page HTML, vous écrivez une page HTML avec du code inclus à l’intérieur afin de réaliser une action précise (dans ce cas là, afficher du texte). Le code PHP est inclus entre une balise de début et une balise de fin qui permettent au navigateur de passer en "mode PHP".

Ce qui distingue le PHP des langages de script comme le Javascript est que le code est exécuté sur le serveur. Si vous avez un script similaire sur votre serveur, le client ne reçoit que le résultat du script, sans aucun moyen d’avoir accès au code qui a produit ce résultat. Vous pouvez configurer votre serveur web afin qu’il analyse tous vos fichiers HTML comme des fichiers PHP. Ainsi, il n’y a aucun moyen de distinguer les pages qui sont produites dynamiquement des pages statiques.

Que peut faire PHP?

Le langage PHP possède les même fonctionnalités que les autres langages permettant d’écrire des scripts CGI, comme collecter des données, générer dynamiquement des pages web ou bien envoyer et recevoir des cookies.

La plus grande qualité et le plus important avantage du langage PHP est le support d’un grand nombre de bases de données. Réaliser une page web dynamique interfacant une base de donnés est extrêmement simple. Les bases de données suivantes sont supportées par PHP:

Adabas D	InterBase	PostgreSQL
dBase	FrontBase	Sesam
Empress	mSQL	Solid
FilePro (lecutre seule)	Direct MS-SQL	Sybase
Hyperwave	MySQL	Velocis
IBM DB2	ODBC	Unix dbm
Informix	Oracle (OCI7 et OCI8)	
Ingres	Ovrimos	

Le langage PHP inclus le support des services utilisant les protocoles tels que IMAP, SNMP, NNTP, POP3 ou encore HTTP. Vous pouvez également ouvrir des connections et interagir en utilisant d’autres protocoles.

La génèse du PHP

Le langage PHP a été conçu durant l’automne 1994 par Rasmus Lerdorf. Les premières versions (qui restèrent privées) étaient utilisées afin de savoir qui venait consulter son CV en ligne. La première version publique fut disponible au début de l’année 1995. Elle fut connue sous le nom de "Personal Sommaire Page Tools". Elle était composée d’un analyseur extrêmement simple qui ne reconnaissait que quelques macros spéciales et d’un petit nombre d’utilitaires couramment utilisés dans les pages web. Un livre d’or, un compteur, etc... L’analyseur fut réécrit durant l’été 1995 et fut appelé PHP/FI Version 2. FI etaient les initiales d’un autre package que Rasmus avait écrit qui interprétait les formulaires HTML. C’est alors qu’il combina le "Personnal Sommaire Page tools" avec le "Form Interpreter" et il y ajouta le support de mSQL: c’est comme cela que naquît PHP/FI. PHP/FI grandit de manière spectaculaire et de nombreuses personnes commencèrent à contribuer à son amélioration.

Il est relativement peu aisé de donner des statistiques, mais on estime que PHP/FI est utilisé sur 15 000 sites web dans le monde entier, fin 1996. Ce chiffre atteint 50 000 durant l'été 1997. L'été 1997 voit aussi un profond changement dans le développemnt du PHP: d'un projet personnel (celui de Ramsus), on passe alors à une projet d'équipe. L'analyseur fut de nouveau réécrit par Zeev Suraski et Andi Gutmans et ce nouvel analyseur forma la base de la version 3 du PHP. Une grande partie du code de PHP/FI fut complètement réécrit alors que l'autre partie fut portée pour donner le PHP Version 3. La dernière version de PHP (PHP 4) utilise le moteur d'analyse Zend (http://www.zend.com/) pour atteindre de nouveaux niveaux de performance, et supporter un nombre encore plus grand de librairies et extensions. Il tourne de manière native sur tous les serveurs web les plus répandus.

Aujourd'hui (Janvier 2001) PHP 3 ou PHP 4 sont distribués avec de nombreux produits commerciaux comme "C2's StrongHold web server" et "RedHat Linux" et il est admis (d'après les chiffres de NetCraft (http://www.netcraft.com/), et leurs statistiques Netcraft Web Server Survey (http://www.netcraft.com/survey/)) que le PHP est utilisé sur 5 100 000 sites web dans le monde entier. Pour comparaison, ce chiffre est légèrement supérieur au nombre de serveurs tournant sous Microsoft Information server (IIS) : 5.03 millions.

Enfin, à l'heure oú ce document est rédigé, la nouvelle génération du PHP est en cours de création. Elle utilisera les qualités de Zend (http://www.zend.com/) pour améliorer les performances et améliorera le support des serveurs web autres que Apache.

Chapitre 2. Installation

Télécharger la dernière version

Les codes source et les exécutables compilés de certains OS (y compris Windows), sont disponibles à http://www.php.net/. Nous recommandons l'utilisation du miroir (http://www.php.net/mirrors.php) le plus proche pour accélérer les chargements.

Installation sous UNIX

Cette section va vous guider lors du processus d'installation et de configuration de PHP sous Unix. Commencez par étudier les sections spécifiques à votre plate-forme ou à votre serveur web avant de passer à l'installation.

Pré-requis :

- Connaissance de base d'UNIX (savoir faire un "make" et compiler en C, si besoin).
- Un compilateur ANSI C (pour les codes sources)
- flex (pour les codes sources)
- bison (pour les codes sources)
- Un serveur web
- Tous les composants nécessaires aux extensions (librairie GD, PDF, etc...)

Il y a plusieurs façons d'installer PHP sur une plate-forme UNIX : soit un processus de compilation-configuration, ou bien avec des packages déjà tout prêts. Cette documentation se concentre sur la première solution.

La première partie du processus est faite en ligne de commande, grâce aux options du script `configure`. Cette section présente l'utilisation des options les plus courantes, mais il y en a beaucoup d'autres à essayer. Reportez-vous à la liste complète des options de configuration pour une liste exhaustive. Voici les différentes méthodes d'installation de PHP :

- Comme module Apache
- Comme module fhttpd
- Pour l'utiliser avec AOLServer, NSAPI, phttpd, Pi3Web, Roxen, thttpd, ou Zeus.
- Comme exécutable CGI

Référence Module Apache

PHP peut être compilé de nombreuses manières différentes, mais la plus populaire est le module Apache. La liste suivante est un récapitulatif de l'installation.

Exemple 2-1. Instructions d'installation PHP 4 (Version Module Apache)

```
1.  gunzip apache_1.3.x.tar.gz
2.  tar xvf apache_1.3.x.tar
3.  gunzip php-x.x.x.tar.gz
4.  tar xvf php-x.x.x.tar
5.  cd apache_1.3.x
6.  ./configure --prefix=/www
7.  cd ../php-x.x.x
8.  ./configure --with-mysql --with-apache=../apache_1.3.x --enable-track-vars
9.  make
10. make install
11. cd ../apache_1.3.x
12. ./configure --activate-module=src/modules/php4/libphp4.a
13. make
14. make install
15. cd ../php-x.x.x
16. cp php.ini-dist /usr/local/lib/php.ini
17. "Editez votre fichier httpd.conf ou srm.conf et ajoutez : "
```

```
      AddType application/x-httpd-php .php
18. "Utilisez votre procédure habituelle pour redémarrer le serveur Apache.
    (vous devez arrêter puis redémarrer le serveur, et pas seulement forcer
    le serveur à relire la configuration initiale).
```

Compilation

Lorsque PHP est configuré, vous êtes prêts à compiler l'exécutable CGI. La commande **make** doit prendre tout en charge. Si ce n'est pas le cas et que vous restez bloqué, reportez-vous aux problèmes courants.

Installation sous Linux

Cette section contient les notes et conseils d'installation de PHP sur les distributions Linux.

Utilisation des packages

De nombreuses distributions Linux disposent d'un système d'installation par package, comme le fameux RPM. Ils vous permettent de faire des installations standard, mais si vous avez des configurations spécifiques (comme par exemple un serveur sécurisé, ou un pilote de base de données exotique), vous aurez probablement à compiler vous-même votre PHP et votre serveur web. Si vous n'êtes pas familier avec la compilation de vos propres logiciels, il vaut mieux rechercher le package qui pourra répondre à vos besoins.

Installation sous HP-UX

Cette section contient les notes et conseils d'installation de PHP sur les distributions HP-UX.

Exemple 2-2. Instructions d'installation pour HP-UX 10

```
From: paul_mckay@clearwater-it.co.uk
04-Jan-2001 09:49
(Ces conseils sont destinés à PHP 4.0.4 et Apache v1.3.9)
Vous voulez installer PHP et Apache sur une HP-UX 10.20?
1. Vous aurez besoin de gzip. Téléchargez la distribution compilée à
http://hpux.connect.org.uk/ftp/hpux/Gnu/gzip-1.2.4a/gzip-1.2.4a-sd-10.20.depot.Z,
puis décompressez la, et utilisez swinstall pour installer.
2. Vous aurez besoin de gcc. Téléchargez une distribution compilée à
http://gatekeep.cs.utah.edu/ftp/hpux/Gnu/gcc-2.95.2/gcc-2.95.2-sd-10.20.depot.gz,
puis décompressez la, et utilisez swinstall pour installer.
3. Vous aurez besoin de GNU binutils.
Téléchargez une distribution compilée à
http://hpux.connect.org.uk/ftp/hpux/Gnu/binutils-2.9.1/binutils-2.9.1-sd-10.20.depot.gz
, puis décompressez la, et utilisez swinstall pour installer.
4. Vous aurez besoin de bison. Téléchargez une distribution compilée à
http://hpux.connect.org.uk/ftp/hpux/Gnu/bison-1.28/bison-1.28-sd-10.20.depot.gz
, puis décompressez la, et utilisez swinstall pour installer.
5. Vous aurez besoin de flex. Téléchargez une distribution source sur l'un
des miroirs http://www.gnu.org. Il se trouve dans le dossier non-gnu du site FTP.
Téléchargez le fichier, décompressez leur,
puis utilisez tar -xvf avec. Allez dans le nouveau dossier flex ainsi créé,
et exécutez la commande "./configure", puis faites un "make", puis un "make install".
Si vous avez des erreurs à cette étape, c'est probablement par ce que gcc et
les autres ne sont pas inscrites dans votre PATH. Ajoutez les.
Maintenant, la partie délicate.
6. Téléchargez les sources d'Apache et de PHP.
7. Décompressez les avec gunzip puis faites "tar -xvf" avec les deux archives.
Nous devons modifier quelques fichiers avant de les compiler.
8. Premièrement, le fichier de configuration doit être modifié car il
semble oublier qu'il est sur une machine HP-UX. Il y a des méthodes plus
rusées, mais le plus simple et le plus efficace est d'ajouter
```

```
    "lt_target=hpux10.20" à la ligne 47286 du script de configuration.
9. Le fichier d'Apache GuessOS doit être modifié. Sous
apache_1.3.9/src/helpers, modifier la ligne 89, en remplaçant
    "echo "hp${HPUXMACH}-hpux${HPUXVER}"; exit 0"
par :
    "echo "hp${HPUXMACH}-hp-hpux${HPUXVER}"; exit 0"
10. Il n'est pas possible d'installer PHP sous forme de shared object sous
HP-UX, ce qui fait que vous devez le compiler en statique. Suivez
simplement les instructions de la section Apache.
11. PHP et Apache sont maintenant compilés correctement, mais Apache ne démarre pas.
    Vous devez créer un nouvel utilisateur Apache, par exemple www, ou apache.
    Puis, modifiez les lignes 252 et 253 de conf/httpd.conf pour remplacer
    User nobody
    Group nogroup
par vos valeurs, par exemple :
    User www
    Group sys
Il n'est pas possible d'éxécuter Apache avec l'utilisateur nobody sous
HP-UX. A partir de ce moment là, PHP et Apache doivent fonctionner.
J'espère que cela aidera quelqu'un.
Paul Mckay.
```

Installation sous Solaris

Cette section contient les notes et conseils d'installation de PHP sur les distributions Solaris.

Logiciels nécessaires

L'installation Solaris oublie généralement les compilateurs C, et leurs utilitaires. Voici la liste des outils nécessaires :

- gcc (recommandé, mais d'autres compilateurs C peuvent fonctionner)
- make
- flex
- bison
- m4
- autoconf
- automake
- perl
- gzip
- tar

De plus, vous devrez aussi installer (et peut être aussi compiler) toutes les librairies nécessaires aux extensions (MySQL, ORACLE..).

Utilisation des packages

Vous pouvez simplifier l'installation Solaris en utilisant pkgadd pour installer la plupart des composants.

Installations Unix/OpenBSD

Cette section contient les notes spécifiques à l'installation de PHP sous OpenBSD (http://www.openbsd.org/).

Utilisation des ports

Ceci est la méthode recommandée d'installation de PHP sous OpenBSD, car elle prend en compte les dernières modifications et mises à jour de sécurité. Pour utiliser cette méthode, assurez vous que vous avez bien ports tree (http://www.openbsd.org/ports.html) récent. Choisissez alors simplement la version que vous souhaitez installer, et utilisez la commande **make install**. Ci-dessous, voici un exemple.

Exemple 2-3. Exemple d'installation de PHP sous OpenBSD avec Ports

```
$ cd /usr/ports/www/php4
$ make show VARNAME=FLAVORS
 (choisissez les versions que vous souhaitez sur votre liste).
$ env FLAVOR="imap gettext ldap mysql gd" make install
$ /usr/local/sbin/php4-enable
```

Utilisation des Packages

Il existe des packages pré-compilés disponibles en téléchargement à OpenBSD (http://www.openbsd.org/). Ils s'intègrent automatiquement avec la version d'Apache installée sur votre OS. Cependant, comme il y a un grand nombre d'options (appelées *flavors*) disponibles, vous trouverez peut-être plus facile de le compiler à partir de l'arbre de ports. Lisez le manuel packages(7) (http://www.openbsd.org/cgi-bin/man.cgi?query=packages) pour plus de détails sur les packages disponibles (en anglais).

Installation sous Mac OS X

Cette section contient les notes et conseils d'installation de PHP sur les distributions Mac OS X.

Utilisation des packages

Il existe quelques versions pré-packagée et pré-compilées de PHP pour Mac OS X. Ils permettent de réaliser rapidement des installations standard, mais si vous avez des configurations personnelles, (comme un serveur sécurisé SSL ou un pilote de base de données exotique), vous devrez compiler PHP et/ou votre serveur web vous-même. Si vous n'êtes pas familier avec la compilation de vos propres logiciels, il vaut mieux rechercher le package qui pourra répondre à vos besoins. Lightyear Design (http://homepage.mac.com/LightyearDesign/MacOSX/Packages/) propose une version pré-compilé de PHP pour OS X tout comme Tenon Intersystems (http://www.tenon.com/products/webten/).

Compilation pour serveur OS X

Il existe deux versions légèrement différentes de Mac OS X, client et serveur. Cette installation est faite pour le OS X Serveur.

Exemple 2-4. Installation sous Mac OS X serveur

```
1. Téléchargez la dernière version de Apache et PHP
2. Décompressez puis désarchivez la, puis configurez Apache comme ceci :
    ./configure --exec-prefix=/usr \
    --localstatedir=/var \
    --mandir=/usr/share/man \
    --libexecdir=/System/Library/Apache/Modules \
    --iconsdir=/System/Library/Apache/Icons \
    --includedir=/System/Library/Frameworks/Apache.framework/Versions/1.3/Headers \
    --enable-shared=max \
    --enable-module=most \
    --target=apache
4. Vous aurez peut être besoin d'ajouter ces lignes ci, pour optimiser la
compilation :
    setenv OPTIM=-O2
5. Puis, allez dans le dossier source de PHP 4, et configurez le :
    ./configure --prefix=/usr \
```

```
    --sysconfdir=/etc \
    --localstatedir=/var \
    --mandir=/usr/share/man \
    --with-xml \
    --with-apache=/src/apache_1.3.12
    Si vous avez d'autres composants (MySQL, GD, etc.), n'oubliez pas de les
    ajouter à ce moment la. Pour l'option --with-apache, ajoutez le chemin
    jusqu'au dossier source d'Apache, par exemple "/src/apache_1.3.12".
6. Exécutez un "make"
7. Exécutez un "make install"
    Cette commande ajoutera un dossier dans le dossier Apache :
    src/modules/php4.
8. Maintenant, reconfigurez Apache pour compiler PHP 4.
    ./configure --exec-prefix=/usr \
    --localstatedir=/var \
    --mandir=/usr/share/man \
    --libexecdir=/System/Library/Apache/Modules \
    --iconsdir=/System/Library/Apache/Icons \
    --includedir=/System/Library/Frameworks/Apache.framework/Versions/1.3/Headers \
    --enable-shared=max \
    --enable-module=most \
    --target=apache \
    --activate-module=src/modules/php4/libphp4.a
    Vous pouvez rencontrer un message qui vous dira que libmodphp4.a est
    obsolète. Si c'est le cas, allez dans le dossier
    src/modules/php4 de votre dossier Apache et exécutez la commande suivante :
    ranlib libmodphp4.a
    Puis, revenez à la racine de la distribution Apache, et recommencez la
    configuration. Cela aura mis à jour la table de liens.
9. Exécutez un "make"
10. Exécutez un "make install"
11. Copiez et renommez le fichier php.ini-dist de votre distribution PHP 4
    dans votre dossier "bin":
    cp php.ini-dist /usr/local/bin/php.ini
    ou (si vous n'avez pas de dossier local)
    cp php.ini-dist /usr/bin/php.ini
```

D'autres exemples pour Mac OS X client (http://www.stepwise.com/Articles/Workbench/Apache-1.3.14-MacOSX.html) et Mac OS X server (http://www.stepwise.com/Articles/Workbench/Apache-1.3.14-MacOSX.html) sont disponibles à Stepwise (http://www.stepwise.com/).

Compilation pour MacOS X client

Ces conseils sont gracieusement fourni par Marc Liyanage (http://www.entropy.ch/software/macosx).

Le module PHP pour Apache est inclus dans Mac OS X. Cette version inclus le support des bases de données MySQL et PostgreSQL.

NOTE: Soyez prudent avec cette manipulation, vous risquez de mettre votre serveur Apache à terre!

Insctructions :

- 1. Ouvrez un terminal
- 2. Tapez "wget http://www.diax.ch/users/liyanage/software/macosx/libphp4.so.gz", attendez la fin du téléchargement.
- 3. Tapez "gunzip libphp4.so.gz"
- 4. Tapez "sudo apxs -i -a -n php4 libphp4.so"

Maintenant, tapez "`sudo open -a TextEdit /etc/httpd/httpd.conf`" TextEdit ouvrira le fichier de configuration. Recherchez ces deux lignes, vers la fin du fichier (Utilisez la commande Find)

```
*    #AddType application/x-httpd-php .php
*    #AddType application/x-httpd-php-source .phps
```

Supprimez les deux marques de commentaires (#), puis sauvez le fichier, et quittez TextEdit.

Finalement, tapez "`sudo apachectl graceful`" pour redémarrer le serveur Apache.

PHP devrait fonctionner. Vous pouvez le tester en placant un script dans le dossier "Sites". Par exemple, le fichier "test.php", qui contient la simple ligne : "`<?php phpinfo() ?>`".

Ouvrez l'URL `127.0.0.1/~your_username/test.php` dans votre navigateur. Vous obtiendrez le tableau de bord de PHP.

Liste complète des options de configuration

Cette section rassemble la liste complète des options de configuration supportées par PHP 3 et PHP 4, à utiliser avec le fichier `configure`, lors de la configuration sous Unix. Certaines options sont disponibles sous PHP 3, d'autres sous PHP 4 et certains sous PHP 3 et PHP 4, comme inidiqué. Il y a de nombreuses options dont le nom a changé entre PHP 3 et PHP 4. Ces options ont des liens entre elles : si vous vous souvenez d'un nom d'option en PHP 3, regardez si le nom a changé.

Base de données

`--with-adabas[=DIR]`

PHP 3, PHP 4: Inclut le support Adabas D. DIR est le dossier d'installation de Adabas (par défaut, `/usr/local`).

Adabas home page (http://www.adabas.com/)

`--enable-dba=shared`

PHP 3: Option non disponible en PHP 3

PHP 4: Compile DBA comme module partagé

`--enable-dbx`

PHP 3: Option non disponible en PHP 3

PHP 4: Inclut le support DBX.

`--enable-dbase`

PHP 3: Option non disponible; utilisez plutôt --with-dbase instead.

PHP 4: Active la librairie dbase livrée avec PHP. Aucune librairie supplémentaire n'est nécessaire.

--with-dbase

PHP 3: Active la librairie dbase livrée avec PHP. Aucune librairie supplémentaire n'est nécessaire.

PHP 4: Option non disponible; utilisez plutôt --enable-dbase instead.

--with-db2[=DIR]

PHP 3, PHP 4: Active le support Berkeley DB2.

--with-db3[=DIR]

PHP 3: Option non disponible en PHP 3

PHP 4: Active le support Berkeley DB3.

--with-dbm[=DIR]

PHP 3, PHP 4: Active le support DBM.

--with-dbmaker[=DIR]

PHP 3: Option non disponible en PHP 3.

PHP 4: Inclut le support DBMaker. DIR est le dossier d'installation DBMaker (par défaut, c'est le dossier de la dernière installation DBMaker, comme `/home/dbmaker/3.6`).

--with-empress[=DIR]

PHP 3, PHP 4: Inclut le support Empress. DIR est le dossier d'installation Empress (par défaut, `$EMPRESSPATH`).

--enable-filepro

PHP 3: Option non disponible; utilisez plutôt --with-filepro instead.

PHP 4: Active la librairie filePro (lecture seule) livrée avec PHP. Aucune librairie supplémentaire n'est nécessaire.

--with-fbsql[=DIR]

PHP 3: Option non disponible.

PHP 4: Inclut le support de FrontBase SQL. DIR est le chemin jusqu'à l'installation de FrontBase base. Par défaut, c'est le dossier standard d'installation Frontbase. L'installation dépend de votre OS : Solaris: `/opt/FrontBase`, WinNT: `\usr\FrontBase`, Linux: `/usr/frontbase`, Mac OSX: `/Library/FrontBase`.

--with-filepro

PHP 3: Inclut le support IBM DB2. Aucune librairie supplémentaire n'est nécessaire.

PHP 4: Option non disponible; utilisez plutôt --enable-filepro.

--with-gdbm[=DIR]

PHP 3, PHP 4: Active le support GDBM.

--with-hyperwave

PHP 3, PHP 4: Active le support Hyperwave.

--with-ibm-db2[=DIR]

PHP 3, PHP 4: Inclut le support IBM DB2. DIR est le dossier d'installation de DB2 (par défaut, `/home/db2inst1/sqllib`).

IBM DB2 (http://www.ibm.com/db2/)

--with-informix[=DIR]

PHP 3, PHP 4: Inclut le support Informix. DIR est le dossier d'installation d'Informix (par défaut, aucune valeur).

--with-ingres[=DIR]

PHP 3: Option non disponible en PHP 3

PHP 4: Inclut le support Ingres II. DIR est le dossier d'installation d'Ingres (par défaut, `/II/ingres`).

--with-interbase[=DIR]

PHP 3, PHP 4: Inclut le support InterBase. DIR est le dossier d'installation d'InterBase (par défaut, `/usr/interbase`).

Fonctions Interbase

Interbase (http://www.interbase.com/)

--with-ldap[=DIR]

PHP 3: Inclut le support LDAP. DIR est le dossier d'installation de LDAP (par défaut `/usr` et `/usr/local`).

PHP 4: Inclut le support LDAP. DIR est le dossier d'installation de LDAP. (par défaut; `/usr/local/ldap`).

Plus de détails sur LDAP sont disponibles à RFC1777 (http://www.faqs.org/rfcs/rfc1777.html) et RFC1778 (http://www.faqs.org/rfcs/rfc1778.html).

--with-msql[=DIR]

PHP 3, PHP 4: Active le support mSQL. DIR est le dossier d'installation de mSQL (par défaut `/usr` et `/usr/local/Hughes`, pour la version 2.0). **configure** détecte automatiquement la version de mSQL qui fonctionne. PHP supporte les versions 1.0 et 2.0, mais si vous compilez PHP avec mSQL 1.0, vous ne pourrez accéder qu'à des bases mSQL 1.0, et vice-versa.

Voir aussi Configuration mSQL dans le fichier de configuration.

mSQL (http://www.hughes.com.au/)

--with-mysql[=DIR]

PHP 3: Inclut le support MySQL. DIR est le dossier d'installation de MySQL (par défaut, il cherche dans différents dossiers où MySQL a coutume d'être installé).

PHP 4: Inclut le support MySQL. DIR est le dossier de l'installation MySQL. S'il est omis, la librairie MySQL livrée en standard avec PHP sera utilisée par défaut.

Voir aussi Configuration MySQL dans le fichier de configuration.

MySQL (http://www.mysql.com/)

--with-ndbm[=DIR]

PHP 3, PHP 4: Active le support NDBM.

--with-ovrimos

PHP 3, PHP 4: Inclut le support Ovrimos.

--with-oci8[=DIR]

PHP 3: Option non disponible en PHP 3

PHP 4: Inclut le support Oracle-oci8. DIR est le dossier d'installation de Oracle-oci8 (par défaut, `ORACLE_HOME`).

--with-oracle[=DIR]

PHP 3: Inclut le support Oracle database. DIR est le dossier d'installation de Oracle (par défaut, `$ORACLE_HOME`).

PHP 4: Inclut le support Oracle-oci7. DIR est le dossier d'installation de Oracle-oci7 (par défaut, `ORACLE_HOME`).

Inclut le support Oracle. Ce support a été testé et permet de travailler avec les versions d'Oracle de 7.0 à 7.3. Le paramètre est le dossier ORACLE_HOME. Vous n'avez pas à spécifier ce paramètre si votre environnement Oracle a été configuré.

Oracle (http://www.oracle.com/)

--with-pgsql[=DIR]

PHP 3: Inclut le support PostgresSQL. DIR est le dossier d'installation de PostgresSQL (par défaut, `/usr/local/pgsql`).

PHP 4: Inclut le support PostgreSQL. DIR est le dossier d'installation de PostgreSQL (par défaut, `/usr/local/pgsql`). Pour compiler en "dl", utilisez la valeur "shared", ou "shared,DIR", pour compiler en "dl", mais spécifier DIR malgré tout.

Voir aussi Postgres dans le fichier de configuration.

PostgreSQL (http://www.postgresql.org/)

--with-solid[=DIR]

PHP 3, PHP 4: Inclut le support Solid. DIR est le dossier d'installation de Solid (par défaut, `/usr/local/solid`).

Solid (http://www.solidtech.com/)

--with-sybase-ct[=DIR]

PHP 3, PHP 4: Inclut le support Sybase-CT. DIR est le dossier d'installation de Sybase (par défaut, `/home/sybase`).

Voir aussi Sybase-CT dans le fichier de configuration.

--with-sybase[=DIR]

PHP 3, PHP 4: Inclut le support Sybase-DB. DIR est le dossier d'installation de Sybase (par défaut, `/home/sybase`).

Voir aussi Sybase dans le fichier de configuration.

Sybase (http://www.sybase.com/)

--with-openlink[=DIR]

PHP 3, PHP 4: Inclut le support OpenLink ODBC. DIR est le dossier d'installation d'OpenLink (par défaut `/usr/local/openlink`). A partir de PHP 4.0.6, cette option n'est plus valable. Utilisez plutôt `--with-iodbc` si vous voulez utiliser l'ODBC de OpenLink Software.

OpenLink Software (http://www.openlinksw.com/)

--with-iodbc[=DIR]

PHP 3, PHP 4: Inclut le support iODBC. DIR est le dossier d'installation d'iODBC (par défaut, `/usr/local`).

Cette fonctionnalité a d'abord été développée avec le gestionnaire iODBC Driver Manager, un pilote ODBC librement distribuable, qui fonctionne sous divers UNIX.

FreeODBC (http://users.ids.net/~bjepson/freeODBC/) ou iODBC (http://www.iodbc.org/)

--with-custom-odbc[=DIR]

PHP 3, PHP 4: Inclut le support ODBC, avec une librairie tierce. Le paramètre DIR est le nom du dossier d'installation de cette librairie. Par défaut, il vaut `/usr/local`.

Cette option implique qus vous avez défini CUSTOM_ODBC_LIBS lorsque vous éxécutez le script de configuration. Vous devez aussi avoir une en-tête `odbc.h` valide dans vos dossiers d'Inclution. Si vous n'en avez pas, créez le, et ajoutez-y vos en-têtes spécifiques. Votre en-tête peut aussi réclamer d'autres définitions, surtout si elle est multi-plate-forme. Définissez les dans CFLAGS.

Par exemple, vous pouvez utilisez Sybase SQL Anywhere sous QNX comme ceci : `CFLAGS=-DODBC_QNX LDFLAGS=-lunix CUSTOM_ODBC_LIBS="-ldblib -lodbc" ./configure --with-custom-odbc=/usr/lib/sqlany50`

--disable-unified-odbc

PHP 3: Inactive le support unified ODBC. Uniquement valable si iODBC, Adabas, Solid, Velocis ou une interface spéciale ODBC a été activée.

PHP 4: Option non disponible en PHP 4

Le module Unified ODBC est commun à toutes les bases de données ODBC, comme par exemple Solid, IBM DB2 et Adabas D. Il fonctionne aussi avec les librairies ODBC normales. Des tests ont été menés avec iODBC, Solid, Adabas D, IBM DB2 et Sybase SQL Anywhere. Il requiert une (et une seule) de ces extensions, ou l'extension Velocis, ou une librairie ODBC spéciale. Cette option n'est possible qu'avec l'utilisation de l'une des options suivantes : --with-iodbc, --with-solid, --with-ibm-db2, --with-adabas, --with-velocis, ou --with-custom-odbc.

Voir aussi Unified ODBC dans le fichier de configuration.

--with-unixODBC[=DIR]

PHP 3: Option non disponible en PHP 3

PHP 4: Inclut le support unixODBC. DIR est le dossier d'installation d'unixODBC (par défaut, `/usr/local`).

--with-velocis[=DIR]

PHP 3, PHP 4: Inclus le support Velocis. DIR est le dossier d'installation de Velocis (par défaut, `/usr/local/velocis`).

Velocis (http://www.raima.com/)

E-commerce

--with-ccvs[=DIR]

PHP 3: Option non disponible en PHP 3

PHP 4: Ajoute le support CCVS. DIR est le dossier d'installation de CCVS.

--with-cybermut[=DIR]

PHP 3: Option non disponible en PHP 3

PHP 4: Inclut le support de Cybermut pour PHP 4. DIR est le dossier du SDK Cybermut, qui contient les deux fichiers `libcm-mac.a` et `cm-mac.h`.

--with-mck[=DIR]

PHP 3: Inclut le support Cybercash MCK. DIR est le dossier d'installation de cybercash mck (par défaut, `/usr/src/mck-3.2.0.3-linux`). Plus d'aide dans le dossier `extra/cyberlib`.

PHP 4: Option non disponible; utilisez plutôt --with-cybercash instead.

--with-cybercash[=DIR]

PHP 3: Option non disponible; utilisez plutôt --with-mck instead.

PHP 4: Inclut le support CyberCash. DIR est le dossier d'installation de CyberCash MCK.

--with-pfpro[=DIR]

PHP 3: Option non disponible en PHP 3

PHP 4: Active le support Verisign Payflow Pro.

Images

--enable-freetype-4bit-antialias-hack

PHP 3: Option non disponible en PHP 3

PHP 4: Inclut le support de FreeType2 (expérimental).

--with-gd[=DIR]

PHP 3: Inclut le support GD. DIR est le dossier d'installation de GD.

PHP 4: Inclut le support GD. DIR est le dossier d'installation de GD. Pour compiler en "dl", utilisez la valeur "shared", ou "shared,DIR", pour compiler en "dl", mais spécifier DIR malgré tout.

--without-gd

PHP 3, PHP 4: Inactive le support GD .

--with-imagick[=DIR]

PHP 3: Inclut le support ImageMagick. DIR est le dossier d'installation de ImageMagick. S'il est omis, PHP essaiera de le trouver de lui-même (expérimental).

PHP 4: Option non disponible en PHP 4

--with-jpeg-dir[=DIR]

PHP 3: dossier JPEG pour pdflib 2.0

PHP 4: dossier JPEG pour pdflib 3.x et 4.x

--with-png-dir[=DIR]

PHP 3: Option non disponible en PHP 3

PHP 4: dossier PNG pour pdflib 3.x et 4.x

--enable-t1lib

PHP 3: Active le support t1lib.

PHP 4: Option non disponible; utilisez plutôt --with-t1lib

`--with-t1lib[=DIR]`

PHP 3: Option non disponible; utilisez plutôt --enable-t1lib.

PHP 4: Inclut le support T1lib.

`--with-tiff-dir[=DIR]`

PHP 3: dossier TIFF pour pdflib 2.0

PHP 4: dossier TIFF pour pdflib 3.x et 4.x

`--with-ttf[=DIR]`

PHP 3, PHP 4: Active le support FreeType.

`--with-xpm-dir[=DIR]`

PHP 3: Option non disponible en PHP 3

PHP 4: dossier XPM pour GD-1.8+

Divers

Ces options seront classées ultérieurement, lorsqu'une catégorie adéquate apparaîtra.

`--disable-bcmath`

PHP 3: Inactive la librairie BCmath.

PHP 4: Option non disponible en PHP 4. La librairie BCmath n'est pas compilée par défaut. Utilisez --enable-bcmath pour l'inclure.

`--with-gmp`

PHP 3, PHP 4 : Inclut le support GMP.

`--disable-display-source`

PHP 3: Compile sans afficher le support des sources

PHP 4: Option non disponible en PHP 4

`--disable-libtool-lock`

PHP 3: Option non disponible en PHP 3

PHP 4: Empêche le verrouillage (risque d'empêcher certaines compilations paralléles).

`--disable-pear`

PHP 3: Option non disponible en PHP 3

PHP 4: N'installe pas PEAR

`--disable-pic`

PHP 3: Option non disponible en PHP 3

PHP 4: Inactive PIC pour les shared objects

`--disable-posix`

PHP 3: Option non disponible en PHP 3; Utilisez plutôt --without-posix

PHP 4: Inactive les fonctions POSIX.

`--disable-rpath`

PHP 3: Option non disponible en PHP 3

PHP 4: Inactive le passage de chemins supplémentaires pour la recherche de librairie lors de l'exécution.

`--disable-session`

PHP 3: Option non disponible en PHP 3

PHP 4: Inactive le support session.

`--enable-bcmath`

PHP 3: Option non disponible en PHP 3; bcmath est compilée par défaut. Utilisez plutôt --disable-bcmath, pour l'inactiver.

PHP 4: Active le support de l'extension bc maths. Voir aussi les fonctions BCMath.

`--enable-c9x-inline`

PHP 3: Option non disponible en PHP 3

PHP 4: Active les sémantiques C9x-inline

`--enable-calendar`

PHP 3: Option non disponible en PHP 3

PHP 4: Active le support des conversions calendaires

`--enable-debug`

PHP 3, PHP 4: Compile sans les symboles de débuggages

`--enable-debugger`

PHP 3: Compile avec les fonctions de débuggage à distance

PHP 4: Option non disponible en PHP 4

`--enable-discard-path`

PHP 3, PHP 4: Si cette option est activée, le CGI PHP peut être placé hors de l'arborescence web, pour que personne ne puisse l'atteindre, même en contournant les .htaccess.

`--enable-dmalloc`

PHP 3, PHP 4: Active dmalloc

`--enable-exif`

PHP 3: Option non disponible en PHP 3

PHP 4: Active le support exif.

--enable-experimental-zts

PHP 3: Option non disponible en PHP 3

PHP 4: Cela risque fortement de ne plus compiler du tout!

--enable-fast-install[=PKGS]

PHP 3: Option non disponible en PHP 3

PHP 4: Optimise pour les installations rapides for fast installation (par défaut, `no`).

--enable-force-cgi-redirect

PHP 3, PHP 4: Active la vérification interne des redirections serveurs. Il est recommandé d'utiliser cette option si vous avez compilé PHP en CGI.

--enable-inline-optimization

PHP 3: Option non disponible en PHP 3

PHP 4: Si vous avez beaucoup de mémoire disponible et que vous utilisez gcc, essayez donc ça.

--enable-libgcc

PHP 3: Option non disponible en PHP 3

PHP 4: Active explicitement les liens avec libgcc

--enable-maintainer-mode

PHP 3, PHP 4: Active des règles de make et de dépendances qui sont parfois absconses et ne servent pas aux utilisateurs habituels (Bref, ne l'utilisez pas).

--enable-memory-limit

PHP 3, PHP 4: Compile avec le support de la limitation de mémoire (par défaut, `no`).

--enable-safe-mode

PHP 3, PHP 4: Active le SAFE_MODE (par défaut, `yes`).

--enable-satellite

PHP 3: Option non disponible en PHP 3

PHP 4: Active le support CORBA via Satellite (Requiert ORBit)

--enable-shared[=PKGS]

PHP 3: Option non disponible en PHP 3

PHP 4: Compile les librairies partagées (par défaut, `yes`).

--enable-sigchild

PHP 3, PHP 4: Active le gestionnaire SIGCHLD propre à PHP.

--enable-static[=PKGS]

PHP 3: Option non disponible en PHP 3

PHP 4: Compile les librairies en statique (par défaut, `yes`).

--enable-sysvsem

PHP 3, PHP 4: Active le support des sémaphores System V.

--enable-sysvshm

PHP 3, PHP 4: Active le support de partage de mémoire System V.

--enable-trans-sid

PHP 3: Option non disponible en PHP 3

PHP 4: Active la propagation transparente des identifiants de session.

--with-cdb[=DIR]

PHP 3, PHP 4: Active le support CDB.

--with-config-file-path=PATH

PHP 3: Indique le chemin dans lequel aller lire le fichier `php3.ini`. Par défaut, c'est `/usr/local/lib`.

PHP 4: Indique le chemin dans lequel allet lire le fichier `php.ini`. Par défaut, c'est `/usr/local/lib`.

--with-cpdflib[=DIR]

PHP 3: Inclut le support ClibPDF. DIR est le dossier d'installation de ClibPDF (par défaut, `/usr/local`).

PHP 4: Inclut le support ClibPDF.(requires cpdflib >= 2). DIR est le dossier d'installation de cpdfllib (par défaut, `/usr`).

--with-esoob[=DIR]

PHP 3: Option non disponible en PHP 3

PHP 4: Inclut le support Easysoft OOB. DIR est le dossier d'installation de OOB (par défaut, `/usr/local/easysoft/oob/client`).

--with-exec-dir[=DIR]

PHP 3, PHP 4: N'autorise que les exécutables placés dans le dossier DIR, lorsque le SAFE MODE est activé (par défaut, c'est `/usr/local/php/bin`).

--with-fdftk[=DIR]

PHP 3, PHP 4: Inclut le support fdftk. DIR est le dossier d'installation de fdftk (par défaut, `/usr/local`).

--with-gnu-ld

PHP 3: Option non disponible en PHP 3

PHP 4: Suppose que le compilateur C utilise GNU ld (par défaut, `no`).

--with-icap[=DIR]

PHP 3: Option non disponible en PHP 3

PHP 4: Inclut le support ICAP.

--with-imap[=DIR]

PHP 3, PHP 4: Inclut le support IMAP. DIR est le dossier d'include d'IMAP (et aussi c-client.a).

--with-imsp[=DIR]

PHP 3: Inclut le support IMSP.(DIR est le dossier d'installation IMSP, là où il y a les dossiers d'include et libimsp.a).

PHP 4: Option non disponible en PHP 4

--with-java[=DIR]

PHP 3: Option non disponible en PHP 3

PHP 4: Inclut le support Java. DIR est le dossier d'installation du JDK). Cette extension peut uniquement être compilée comme "dl".

--with-kerberos[=DIR]

PHP 3: Option non disponible en PHP 3

PHP 4: Inclut le support Kerberos dans IMAP.

--with-mcal[=DIR]

PHP 3, PHP 4: Inclut le support MCAL.

--with-mcrypt[=DIR]

PHP 3, PHP 4: Inclut le support mcrypt. DIR est le dossier d'installation de mcrypt.

--with-mhash[=DIR]

PHP 3, PHP 4: Inclut le support mhash. DIR est le dossier d'installation de mhash.

--with-mm[=DIR]

PHP 3: Option non disponible en PHP 3

PHP 4: Inclut le support mm pour le stockage de session.

--with-mod_charset

PHP 3, PHP 4: Active le transfert des tables depuis le module Apache mod_charset (Rus Apache).

--with-pdflib[=DIR]

PHP 3: Inclut le support pdflib (testé avec 0.6 et 2.0). DIR est le dossier d'installation de pdflib (par défaut, c'est `/usr/local`).

PHP 4: Inclut le support pdflib 3.x/4.x. DIR est le dossier d'installation de pdflib. Par défaut, c'est `/usr/local`.

PHP 4 et PDFlib 3.x/4.x requiert les librairies JPEG et TIFF. Lors de la compilation du support PDFlib utilise les options --with-jpeg-dir et --with-tiff-dir. Vous pouvez aussi utiliser --with-png-dir et --with-zlib-dir, pour compiler le support PNG et Zlib avec PDFlib.

--enable-shared-pdflib

PHP 3, PHP 4: Inclut pdflib comme shared librairy.

--with-readline[=DIR]

PHP 3: Option non disponible en PHP 3

PHP 4: Inclut le support readline. DIR est le dossier d'installation de readline.

--with-regex=TYPE

PHP 3: Option non disponible en PHP 3

PHP 4: Type de librairie d'expressions régulières : système, apache, php

--with-servlet[=DIR]

PHP 3: Option non disponible en PHP 3

PHP 4: Inclut le support servlet. DIR est le dossier d'installation de JSDK. Ce SAPI demande que l'extension Java soit compilée comme shared dl.

--with-ming

PHP 3: Option non disponible en PHP 3

PHP 4: Active le support Flash 4 avec Ming.

--with-swf[=DIR]

PHP 3: Option non disponible en PHP 3

PHP 4: Active le support SWF.

--with-system-regex

PHP 3: Inactive la librairie d'expressions régulières livrée avec PHP.

PHP 4: (Obsolète) Utilise la librairie d'expressions régulières système.

--with-tsrm-pth[=pth-config]

PHP 3: Option non disponible en PHP 3

PHP 4: Utilise GNU Pth.

--with-tsrm-pthreads

PHP 3: Option non disponible en PHP 3

PHP 4: Utilise les threads POSIX (par défaut).

--with-x

PHP 3: Utilise X Window System

PHP 4: Option non disponible en PHP 4

--with-bz2[=DIR]

PHP 4: Ajoute le support bzip2. DIR est le dossier d'installation de bzip2.

--with-zlib-dir[=DIR]

PHP 3: Dossier zlib pour pdflib 2.0 ou active le support zlib.

PHP 4: Dossier zlib pour pdflib 3.x/4.x ou active le support zlib.

--with-zlib[=DIR]

PHP 3, PHP 4: Inclut le support zlib. (requiert zlib >= 1.0.9). DIR est le dossier d'installation de zlib (par défaut, `/usr`).

--with-zziplib[=DIR]

PHP 4: Inclut le support ZZIPlib (requiert ZZIPlib >= 0.10.6). DIR est le dossier d'installation de ZZIPliby (par défaut, `/usr/local`).

La dernière version de ZZIPlib est disponible à http://zziplib.sourceforge.net/.

--without-pcre-regex

PHP 3: Inactive le support des expressions régulières Perl.

PHP 4: Inactive le support des expressions régulières Perl. Utilisez --with-pcre-regex=DIR pour spécifier le dossier d'installation de PCRE, si vous n'utilisez pas la librairie livrée en standard.

--without-posix

PHP 3: N'Inclut pas lrs fonctions POSIX.

PHP 4: Option non disponible en PHP 4; utilisez plutôt --disable-posix.

Réseau

--with-curl[=DIR]

PHP 3: Option non disponible en PHP 3

PHP 4: Active le support CURL.

--enable-ftp

PHP 3: Option non disponible; utilisez plutôt --with-ftp

PHP 4: Active le support FTP.

--with-ftp

PHP 3: Inclut le support FTP.

PHP 4: Option non disponible; utilisez plutôt --enable-ftp instead

--disable-url-fopen-wrapper

PHP 3, PHP 4: Inactive le support des URL avec **fopen()**.

Avertissement

Cette option n'est disponible que jusqu'à la version 4.0.3. Les versions plus récentes fournissent un paramètre dans le fichier `php.ini` appelé *`allow_url_fopen`*, afin de vous éviter de faire ce choix au moment de la compilation.

--with-mod-dav=DIR

PHP 3, PHP 4: Inclut le support DAV, grâce au module Apache mod_dav. DIR est le dossier d'installation de mod_dav (valable uniquement pour les serveurs Apache).

--with-openssl[=DIR]

PHP 3, PHP 4: Inclut le support OpenSSL avec SNMP.

--with-snmp[=DIR]

PHP 3, PHP 4: Inclut le support SNMP. DIR est le dossier d'installation de SNMP (par défaut, il scanne un nombre de dossiers habituels de l'installation SNMP). Utilisez la valeur de "shared" pour compiler sous forme de "dl", ou "shared,DIR" pour compiler sous forme de "dl" tout en spécifiant un dossier.

--enable-ucd-snmp-hack

PHP 3, PHP 4: Active le hack UCD SNMP

--enable-sockets

PHP 3: Option non disponible en PHP 3

PHP 4: Active le support des sockets.

--with-yaz[=DIR]

PHP 3: Option non disponible en PHP 3

PHP 4: Inclut le support YAZ.(ANSI/NISO Z39.50). DIR est le dossier d'installation de YAZ (dossier bin).

--enable-yp

PHP 3: Option non disponible; utilisez plutôt --with-yp

PHP 4: Active le support YellowPages (YP).

--with-yp

PHP 3: Active le support YellowPages (YP).

PHP 4: Option non disponible; utilisez plutôt --enable-yp

--with-mnogosearch

PHP 3, PHP 4: Inclut le support mnoGoSearch.

Comportement PHP

--enable-magic-quotes

PHP 3, PHP 4: Active les magic quotes par défaut.

--disable-short-tags

PHP 3, PHP 4: Désactive la forme courte des balises PHP (<? ?>).

--enable-track-vars

PHP 3: Active le suivi des variables GET/POST/Cookie par défaut.

PHP 4: Option non disponible en PHP 4; à partir de PHP 4.0.2, cette option est toujours activée.

Serveur

--with-aolserver-src=DIR

PHP 3: Option non disponible en PHP 3

PHP 4: Indique le chemin jusqu'à la distribution source de AOLserver

--with-aolserver=DIR

PHP 3: Option non disponible en PHP 3

PHP 4: Indique le chemin jusqu'à la distribution installée de AOLserver.

--with-apache[=DIR]

PHP 3, PHP 4: Compile PHP en module Apache. DIR est le dossier d'installation supérieur d'Apache (par défaut, `/usr/local/etc/httpd`.

--with-apxs[=FILE]

PHP 3, PHP 4: Compile PHP comme module partagé d'Apache module. FILE est le chemin optionnel jusqu'à Apache apxs tool; par défaut, c'est apxs).

--enable-versioning

PHP 3: Tire profit du système de versionnage et de scoping fourni par Solaris 2.x et Linux

PHP 4: Exporte uniquement les symboles nécessaires. Voyez l'installation pour plus de détails.

--with-caudium[=DIR]

PHP 3: Option non disponible en PHP 3

PHP 4: Compile PHP sous forme de module Pike pour être utilisé avec le serveur web Caudium. DIR est le dossier d'installation de Caudium (par défaut, `$prefix/caudium/server`. Le préfixe est paramétré par l'option --prefix (par défaut, `/usr/local`).

--with-fhttpd[=DIR]

PHP 3, PHP 4: Compile PHP comme module fhttpd. DIR est le dossier d'installation de fhttpd (par défaut, `/usr/local/src/fhttpd`).

--with-nsapi=DIR

PHP 3: Option non disponible en PHP 3

PHP 4: Indique le chemin jusqu'au serveur Netscape

--with-phttpd=DIR

PHP 3: Option non disponible en PHP 3

PHP 4:

--with-pi3web=DIR

PHP 3: Option non disponible en PHP 3

PHP 4: Compile PHP comme module pour Pi3Web.

--with-roxen=DIR

PHP 3: Option non disponible en PHP 3

PHP 4: Compile PHP comme module pour Pi3Web Pike. DIR est le dossier d'installation de Roxen (par défaut, `/usr/local/roxen/server`).

--enable-roxen-zts

PHP 3: Option non disponible en PHP 3

PHP 4: Compile le module Roxen en utilisant Zend Thread Safety.

--with-thttpd=SRCDIR

PHP 3: Option non disponible en PHP 3

PHP 4:

--with-zeus=DIR

PHP 3: Option non disponible en PHP 3

PHP 4: Compile PHP comme module ISAPI pour Zeus.

Texte et langue

--with-aspell[=DIR]

PHP 3, PHP 4: Inclut le support ASPELL.

--with-gettext[=DIR]

PHP 3, PHP 4: Inclut le support GNU gettext. DIR est le dossier d'installation de gettext (par défaut, `/usr/local`).

--with-iconv[=DIR]

PHP 3: Option non disponible en PHP 3

PHP 4: Inclut le support iconv.

--with-pspell[=DIR]

PHP 3: Option non disponible en PHP 3

PHP 4: Inclut le support PSPELL.

--with-recode[=DIR]

PHP 3: Inclut le support GNU recode.

PHP 4: Inclut le support recode. DIR est le dossier d'installation de recode.

--enable-shmop

PHP 3, PHP 4 : Inclut le support shmop.

XML

--with-dom[=DIR]

PHP 3: Option non disponible en PHP 3

PHP 4: Active le support DOM. (requiert libxml >= 2.0). DIR est le dossier d'installation de libxml (par défaut, `/usr`).

--enable-sablot-errors-descriptive

PHP 3: Option non disponible en PHP 3

PHP 4: Active les erreurs descriptives.

--with-sablot[=DIR]

PHP 3: Option non disponible en PHP 3

PHP 4: Active le support Sablotron.

`--enable-wddx`

PHP 3: Option non disponible en PHP 3

PHP 4: Active le support WDDX.

`--disable-xml`

PHP 3: Option non disponible en PHP 3; Les fonctions XML ne sont pas construites par défaut. Utilisez plutôt --with-xml pour les activer.

PHP 4: Inactive le support XML, qui utilise la librairie expat, livrée avec PHP.

`--with-xml`

PHP 3: Active le support XML.

PHP 4: Option non disponible en PHP 4; Le support XML est activé par défaut. Utilisez plutôt --disable-xml pour l'inactiver.

Installation sous Windows 9x/ME/NT/2000

Il y a deux méthodes principales pour installer PHP sous Windows : soit manuellement, soit avec InstallShield.

Si vous avez Microsoft Visual Studio, vous pouvez aussi compiler PHP à partir des sources.

Une fois que PHP est installé sur votre Windows, vous pouvez aussi ajouter diverses extensions.

InstallShield sous Windows

L'installeur Windows de PHP disponible depuis les pages de téléchargement (http://www.php.net/), installe la version CGI de PHP, et configure les serveurs web IIS, PWS, et Xitami.

Installez votre serveur HTTP favori sur votre système et assurez-vous qu'il fonctionne.

Exécutez l'installeur et suivez les instructions fournies par le wizard. Deux types d'installation sont fournis : standard, qui utilise toutes les configurations par défaut les plus pratiques, et avancée, qui pose un maximum de questions pour paramétrer le plus finement.

Le wizard d'installation rassemble suffisamment d'informations pour configurer `php.ini` et le serveur web qui utilisera PHP. Pour IIS, mais aussi PWS sous NT Workstation, une liste de l'arboresence web est affichée, et vous pouvez sélectionner les dossiers qui utiliseront PHP.

Une fois l'installation terminée, l'installeur vous informera que vous devez redémarrer. Suivez ce conseil, ou commencez à utiliser PHP immédiatement.

Instructions Générales d'installation

Ce guide vous permet d'installer et de configurer manuellement PHP sur vos stations Windows 9x/Me/NT/2000. La première version de ce guide a été compilée par Bob Silva (mailto:bob_silva@mail.umesd.k12.or.us). La version originale est disponible (en anglais) à http://www.umesd.k12.or.us/php/win32install.html.

Ce guide fournit une aide d'installation pour :

- Personal Web Server (Version la plus récente recommandée)
- Internet Information Server 3 ou 4
- Apache 1.3.x
- Omni HTTPd 2.0b1 et plus récent
- Oreilly Website Pro
- Xitami

PHP 4 pour Windows est décliné en deux versions : un exécutable CGI (`php.exe`), et plusieurs modules SAPI (par exemple php4isapi.dll). Cette dernière forme est nouvelle pour PHP 4 et founit des performances améliorées ainsi que des fonctionnalités supplémentaires. Notez cependant que les modules SAPI *ne sont pas* considéré comme ayant atteint une qualité de production. La raison à cela est que les modules SAPI utilisent le système de thread sécurisé de PHP, ce qui est nouveau en PHP 4, et qui n'a pas été testé et torturé suffisamment pour être considérés comme stable. Il y a encore quelques bugs qui trainent. D'un autre côté, certains d'entre vous ont rapporté des résultats significativement meilleurs avec les modules SAPI, même si nous ne connaissons actuellement personne qui le fasse fonctionner en production. En clair, faites votre choix : soit vous avez absolument besoin de stabilité, et il vaut mieux laisser les performances SAPI de côté; soit vous avez besoin de performances, et alors c'est l'occasion de tester en production et de nous rapporter vos résultats.

Si vous choisissez l'un des modules SAPI et utilisez Windows 95, pensez à télécharger la mise à jour DCOM à Microsoft DCOM pages (http://download.microsoft.com/msdownload/dcom/95/x86/en/dcom95.exe). Pour le module ISAPI, comme un serveur web compatible est nécessaire (testé avec IIS 4.0, PWS 4.0 et IIS 5.0). IIS 3.0 *n'est pas* supporté; vous devez télécharger et installer le Windows NT 4.0 Option Pack avec IIS 4.0 si vous voulez le support natif de PHP.

Voici les différentes étapes d'installation avant les étapes spécifiques au serveur.

- Extrayez la distribution dans le dossier de votre choix. `"C:\PHP\"` est une bonne idée.
- L'exécutable binaire PHP, les modules SAPI, et certaines extensions utilisent des DLL externes. Assurez vous que ces DLL sont dans votre distribution, et dans un dossier qui est cité dans le PATH Windows. Le mieux à faire est de copier les fichiers ci-dessous dans votre dossier système, qui est généralement :
  c:\windows\system pour Windows 95/98
  c:\winnt\system32 pour Windows NT/2000
Les fichiers à copier sont :
  'php4ts.dll', s'il existe, écrasez le
  Les fichiers 'dlls' de votre distribution. Si vous les avez déjà installé, ne les remplacez pas, sauf si quelquechose ne fonctionne pas
- Copiez le fichier '`php.ini-dist`' dans votre dossier '%WINDOWS%' sous Windows 95/98, ou vers votre dossier '%SYSTEMROOT%' sous Windows NT ou Windows 2000 et renommez le en '`php.ini`'. Votre dossier '%WINDOWS%' ou '%SYSTEMROOT%' est généralement :
  c:\windows pour Windows 95/98
  c:\winnt ou c:\winnt40 pour les serveurs NT/2000
- Editez votre fichier `php.ini` :
  - Vous devez changer votre option 'extension_dir' pour qu'il pointe sur votre dossier d'installation PHP, ou vers l'endroit où vous avez installé vos 'php_*.dll'. ex: c:\php
  - Si vous utilisez Omni Httpd, sautez l'étape suivante. Modifiez 'doc_root' pour qu'il pointe sur votre racine de serveur web. ex: `c:\apache\htdocs` ou `c:\webroot`.
  - Choisissez les modules que vous voulez charger lorsque PHP démarre. Vous pouvez décommenter les lignes 'extension=php_*.dll' pour charger ces modules. Certains modules requièrent que des librairies supplémentaires soient installées sur votre système. La FAQ (http://www.php.net/FAQ.php) PHP a plus d'informations sur ces librairies. Vous pouvez aussi charger dynamiquement ces librairies avec **dl("php_*.dll");**. Voyez la section sur les extensions Windows.
  - Sous PWS et IIS, vous pouvez modifier le fichier `browscap.ini` pour qu'il pointe sur : `c:\windows\system\inetsrv\browscap.ini` sous Windows 95/98 et `c:\winnt\system32\inetsrv\browscap.ini` sous NT. Plus de détails sur l'utilisation de browscap sont accessibles sur ce miroir (http://php.netvision.net.il/browser-id.php3), sélectionnez le bouton "source" pour le voir en action.

Compilation des sources

Avant de commencer, il est bon de se poser la question suivante : "Pourquoi la compilation de PHP sous Windows est si difficile?". Deux raisons viennent immédiatement à l'esprit :

1. Windows ne dispose pas (encore) d'une grande communauté de développeurs qui partagent librement leurs sources. La conséquence directe est que les investissements nécessaires en infrastructure pour supporter ce type de

développement n'ont pas été faits. Ce qui fait que le portage des utilitaires Unix a été la solution pour pallier ce manque. Ne soyez donc pas surpris de rencontrer cette parenté de temps en temps.

2. La majorité des instructions que vous allez rencontrer sont du type : "faire et oublier". Alors, asseyez-vous confortablement et suivez aussi scrupuleusement que possible les instructions.

Préparation

Avant de commencer, il faut télécharger un maximum de fichiers!

- Pour commencer, téléchargez le Cygwin depuis le miroir cygwin (http://sources.redhat.com/cygwin/download.html) le plus proche. Cela vous donnera les utilitaires GNU les plus populaires, utilisés durant le processus de compilation.
- Téléchargez le reste des utilitaires de compilation dont vous aurez besoin depuis le site PHP à http://www.php.net/extra/win32build.zip (http://www.php.net/extra/win32build.zip).
- Téléchargez le code source du DNS utilisé par PHP à http://www.php.net/extra/bindlib_w32.zip (http://www.php.net/extra/bindlib_w32.zip). Il remplacera le fichier `resolv.lib` inclut dans `win32build.zip`.
- Si vous n'avez pas d'utilitaire de dézippage, vous devez en télécharger un. Une version libre est disponible à InfoZip (http://www.cdrom.com/pub/infozip/UnZip.html).

Finalement, vous aurez besoin des sources PHP 4 elles-mêmes!! Les dernières versions sont accessibles sur le serveur CVS anonyme (http://www.php.net/anoncvs.php). Si vous téléchargez une version intermédiaire (http://snaps.php.net/) ou la source (http://www.php.net/downloads.php), vous devez non seulement extraire les fichiers, mais aussi convertir les nouvelles lignes en leur équivalent windows (crlf) dans les fichiers `*.dsp` et `*.dsw` avant que Microsoft Visual C++ ne soit capable de les comprendre.

> **Note :** Placez les dossiers `Zend` et `TSRM` dans le dossier `php4` pour que les projets puissent les trouver durant la compilation.

Mettre tout ensemble

- Suivez les instructions pour installer l'utilitaire d'unzip de votre choix.
- Exécutez `setup.exe` et suivez les instructions d'installation. Si vous décidez d'installer dans un autre dossier que `c:\cygnus`, indiquez le au processus de compilation en modifiant la variable d'environnement Cygwin. Sous Windows 95/98, modifier une variable d'environnement se fait en ajoutant une ligne dans le fichier `autoexec.bat`. Sous Windows NT, allez dans le menu "Démarrer => Paramètres => Panneau de contrôle => Système " ("My Computer => Control Panel => System ") et sélectionnez l'onglet "environnement" ("environment").

> **Avertissement**
>
> Créez un dossier temporaire pour Cygwin, sinon de nombreuses commandes (comme bison) échoueront. Sous Windows 95/98, **`mkdir C:\TMP`**. Sous Windows NT, **`mkdir %SystemDrive%\tmp`**.

- Créez un dossier et dézippez `win32build.zip` dedans.
- Lancez Microsoft Visual C++, et allez dans le menu "select Tools => Options". Dans le dialogue, sélectionnez l'onglet "directories". Assurez-vous que `cygwin\bin`, `win32build\include`, et `win32build\lib` sont bien dans les menus déroulants "Executables", "Include", et "Library". (Pour ajouter une entrée, sélectionnez une ligne blanche, et tapez). Une entrée typique ressemble à ceci :
  - `c:\cygnus\bin`
  - `c:\php-win32build\include`
  - `c:\php-win32build\lib`

  Pressez "OK", et sortez de Visual C++.

- Créez un autre dossier et dézippez `bindlib_w32.zip` dedans. Décidez si vous avez besoin des symboles de débuggage (bindlib - Win32 Debug) ou non (bindlib - Win32 Release). Compilez la configuration adéquate :
  - Pour les utilisateurs de GUI, lancez VC++, puis sélectionnez le menu "File => Open Workspace" et "bindlib". Puis sélectionnez "Build=>Set Active Configuration" et sélectionnez la configuration voulue. Enfin, sélectionnez "Build=>Rebuild All".
  - Pour les utilisateurs en ligne de commande, assurez-vous que vous avez enregistré les variables d'environnement C++, ou que vous avez exécuté **vcvars.bat**. Exécutez maintenant l'une des commandes suivantes :
    - **`msdev bindlib.dsp /MAKE "bindlib - Win32 Debug"`**
    - **`msdev bindlib.dsp /MAKE "bindlib - Win32 Release"`**
  - A ce stade, vous avez une librairie `resolv.lib` utilisable, soit dans votre dossier `Debug`, soit sans le dossier `Release`. Copiez ce fichier dans votre dossier `win32build\lib`, en remplaçant le fichier du même nom.

Compilation

La meilleure façon de compiler est de commander par la version CGI/exécutable.

- Pour les utilisateurs GUI, lancez VC++, puis sélectionnez le menu "File => Open Workspace" et sélectionnez "php4ts". Ensuite, sélectionnez le menu "Build=>Set Active Configuration", et sélectionnez la configuration voulue. Finalement, sélectionnez le menu "Build=>Rebuild All".
- Pour les utilisateurs en ligne de commande, assurez-vous que vous avez enregistré les variables d'environnement C++, ou que vous avez exécuté **vcvars.bat**. Exécutez maintenant l'une des commandes suivantes :
  - **`msdev php4ts.dsp /MAKE "php4ts - Win32 Debug_TS"`**
  - **`msdev php4ts.dsp /MAKE "php4ts - Win32 Release_TS"`**
  - A ce stade, vous avez une librairie `php.exe` utilisable, soit dans votre dossier `Debug_TS` soit sans le dossier `Release_TS`.

Répétez les instructions ci-dessus avec `php4isapi.dsp` (qui est dans `sapi\isapi`) pour compiler le code nécessaire pour intégrer PHP avec Microsoft IIS.

Installation des extensions sous Windows

Après avoir installé PHP et votre serveur web sous Windows, vous voudrez sÛrement ajouter quelques extensions bien pratiques. La table suivante liste une partie des extensions disponibles. Comme indiqué dans le manuel, vous pouvez choisir quelles extensions vous voulez charger en décommentant la ligne 'extension=php_*.dll' dans le fichier `php.ini`. Vous pouvez aussi charger dynamiquement un module avec la fonction **dl()**.

Les fichiers DLLs des extensions PHP sont préfixés par 'php_' en PHP 4, et 'php3_' en PHP 3. Cela évite la confusion des extensions PHP et de leurs librairies.

> **Note :** En PHP 4.0.4pl1, les extensions BCMath, BCMath,Calendar, COM, FTP, MySQL, ODBC, PCRE, Sessions, WDDX et XML sont activées *par défaut*. Vous n'avez rien à faire pour qu'elles soient inclues. Lisez le fichier `README.txt` ou `install.txt` dans votre distribution pour connaître la liste des modules par défaut.

Tableau 2-1. Extensions PHP

php_bz2.dll	Fonctions de compression Bzip2
php_calendar.dll	Fonctions de conversions calendaires (Depuis PHP 4.03, elles sont activées par défaut)
php_cpdf.dll	Fonctions ClibPDF

php_crypt.dll	Fonctions de cryptage
php_ctype.dll	Fonctions ctype
php_curl.dll	Fonctions CURL
php_cybercash.dll	Fonctions de paiement Cybercash
php_db.dll	Fonctions DBM
php_dba.dll	Fonctions dbm-style
php_dbase.dll	Fonctions DBase
php3_dbm.dll	Librairie d'émulation GDBM via Berkely DB2
php_domxml.dll	Fonctions DOM XML
php_dotnet.dll	Fonctions .NET
php_exif.dll	Entêtes EXIF des images JPEG
php_fbsql.dll	Fonctions FrontBase
php_fdf.dll	Fonction FDF (Forms Data Format)
php_filepro.dll	Lecture des bases filepro
php_ftp.dll	Fonctions FTP(Depuis PHP 4.0.3, elles sont activées par défaut)
php_gd.dll	Bibliothèque GD (pour les manipulations d'images)
php_gettext.dll	Fonctions GNU Gettext
php_hyperwave.dll	Fonctions HyperWave
php_iconv.dll	Fonctions de conversions ICONV
php_ifx.dll	Fonctions Informix
php_iisfunc.dll	Fonctions IIS
php_imap.dll	Fonctions IMAP 4(en PHP 3: php3_imap4r1.dll)
php_ingres.dll	Fonctions Ingres II
php_interbase.dll	Fonctoins InterBase
php_java.dll	Extension Java
php_ldap.dll	Fonctions LDAP
php_mhash.dll	Fonctions Mhash
php_ming.dll	Fonctions Ming pour Flash
php_msql.dll	Fonctions mSQL
php3_msql1.dll	Fonctions mSQL 1
php3_msql2.dll	Fonctions mSQL 2
php_mssql.dll	Fonctions MSSQL (anciennement php_mssql70.dll, requiert MSSQL DB-Libraries)
php3_mysql.dll	Fonctions MySQL (Activées par défaut en PHP 4)
php_nsmail.dll	Fonctions Netscape mail
php3_oci73.dll	Fonctions Oracle
php_oci8.dll	Fonctions Oracle 8
php_openssl.dll	Fonctions OpenSSL
php_oracle.dll	Fonctions Oracle
php_pdf.dll	Fonctions PDF
php_pgsql.dll	Fonctions PostgreSQL
php_printer.dll	Fonctions d'impression
php_sablot.dll	Focntions XSLT
php_snmp.dll	Fonctions SNMP get et walk (NT uniquement!)

php_sybase_ct.dll	Fonctions Sybase
php_yaz.dll	Fonctions YAZ
php_zlib.dll	Fonctions ZLib

Installation du serveur Apache

Cette section contient des notes spécifiques pour l'installation de PHP avec Apache, aussi bien pour la version Unix que Windows.

Détails pour l'installation de PHP sous Apache sous Unix.

Vous pouvez sélectionner des options à ajouter au fichier **configure** à la ligne 8 depuis la liste complète des options de configuration.

Exemple 2-5. Instructions d'installation (version module)

```
1.  gunzip apache_1.3.x.tar.gz
2.  tar xvf apache_1.3.x.tar
3.  gunzip php-x.x.x.tar.gz
4.  tar xvf php-x.x.x.tar
5.  cd apache_1.3.x
6.  ./configure --prefix=/www
7.  cd ../php-x.x.x
8.  ./configure --with-mysql --with-apache=../apache_1.3.x --enable-track-vars
9.  make
10. make install
11. cd ../apache_1.3.x
12. for PHP 3: ./configure --activate-module=src/modules/php3/libphp3.a
    for PHP 4: ./configure --activate-module=src/modules/php4/libphp4.a
13. make
14. make install
  Au lieu de cette étape, vous pouvez aussi copier le binaire
  httpd et remplacer votre exécutable actuel. Assurez-vous tout
  de même que le serveur est bien éteint.
15. cd ../php-x.x.x
16. for PHP 3: cp php3.ini-dist /usr/local/lib/php3.ini
    for PHP 4: cp php.ini-dist /usr/local/lib/php.ini
  Vous pouvez éditer votre fichier php.ini pour modifier
  certaines options PHP. Si vous préférez placer ce fichier ailleurs,
  utilisez --with-config-file-path=/path lors de l'étape 8.
17. Editez votre fichier httpd.conf ou srm.conf file et ajoutez :
     Pour PHP 3:   AddType application/x-httpd-php3 .php3
     Pour PHP 4:   AddType application/x-httpd-php  .php
  Vous pouvez choisir n'importe quelle extension que vous voulez ici. .php
  est uniquement une suggestion. Vous pouvez aussi inclure .html.
18. Utilisez votre procédure habituelle pour démarrer votre serveur Apache.
(vous devez l'éteindre et le redémarrer, pas seulement lui envoyer
un signal HUP ou USR1.)
```

Suivant votre installation d'Apache et votre variante d'Unix, il existe de nombreuses façons d'arrêter et redémarrer Apache. Voici une liste des commandes typiques, pour différentes installations. Remplacez `/path/to/` par le chemin d'accès à vos applications sur votre système.

```
1. Nombreuses variantes Linux SysV :
/etc/rc.d/init.d/httpd restart
2. Avec les scripts apachectl :
/path/to/apachectl stop
/path/to/apachectl start
```

```
3. httpdctl et httpsdctl (utilisant OpenSSL), similaire à apachectl:
/path/to/httpsdctl stop
/path/to/httpsdctl start
4. En utilisant mod_ssl, ou un autre seveur SSL, manuellement :
/path/to/apachectl stop
/path/to/apachectl startssl
```

Les exécutables apachectl et http(s)dctl peuvent être situés dans différents dossiers. Si votre système a `locate` ou `whereis` ou `which`, utilisez-les pour retrouver vos programmes.

Différents exemples de compilation PHP pour Apache suivent :

```
./configure --with-apxs --with-pgsql
```

Cette commande va créer une librairie partagée `libphp4.so` qui sera chargée par Apache avec une ligne LoadModule dans le fichier `httpd.conf`. Le support PostgreSQL est aussi inclut dans `libphp4.so`.

```
./configure --with-apxs --with-pgsql=shared
```

Cette commande va créer une autre librairie partagée `libphp4.so`, mais va aussi créer une librairie partagée `pgsql.so` qui sera chargée dans PHP avec les options de configurations du fichier `php.ini` ou par chargement dynamique avec **dl()**.

```
./configure --with-apache=/path/to/apache_source --with-pgsql
```

Cette commande va créer une autre librairie partagée `libmodphp4.a`, un fichier `mod_php4.c` et quelques fichiers compagnons dans le dossier `src/modules/php4` de dossier Apache. Puis, vous devez compiler Apache avec `--activate-module=src/modules/php4/libphp4.a` et le système de compilation d'Apache va créer un fichier `libphp4.a` et le lien statiquement avec `httpd`. Le support PostgreSQL est alors inclut directement dans l'exécutable `httpd`, ce qui fait que le résultat final est un fichier unique `httpd`, qui inclus Apache et PHP.

```
./configure --with-apache=/path/to/apache_source --with-pgsql=shared
```

Identique à la version précédente, mais au lieu d'inclure le support PostgreSQL directement dans l'exécutable final `httpd`, vous allez obtenir une librairie partagée `pgsql.so` que vous pouvez charger dans PHP soit grâce au fichier de configuration `php.ini` ou dynamiquement avec **dl()**.

Lorsque vous faites le choix entre les différents modes de compilation de PHP, vous devez prendre en compte leurs avantages et inconvénients respectifs. Les objets partagés permettent de compiler PHP et Apache de manière séparée, et vous n'aurez pas à compiler l'ensemble pour faire évoluer PHP. La compilation statique permet de charger et d'éxécuter plus rapidement PHP. Pour plus d'informations, voyez webpage on DSO support (http://www.apache.org/docs/dso.html).

Détails sur l'installation de PHP sous Windows avec Apache 1.3.x

Il y a deux méthodes pour faire fonctionner PHP avec Apache 1.3.x sous Windows. La première est d'utiliser l'exécutable CGI (php.exe), l'autre est d'utiliser les modules Apache DLL. Dans les deux cas, vous devez arrêter le serveur Apache, éditer votre fichier `srm.conf` ou `httpd.conf` pour configurer Apache.

Bien qu'il puisse y avoir quelques différences de configurations de PHP sous Apache, le processus reste simple et à la portée du néophyte. Reportez-vous aux documentations Apache pour plus de détails sur ces directives.

Si vous avez dézippé le package dans `C:\PHP\` comme indiqué dans Instructions Générales d'installation, vous devez insérer les lignes suivantes dans votre fichier `srm.conf` ou `httpd.conf` pour qu'il fonctionne en CGI :

- `ScriptAlias /php/ "c:/php/"`

- `AddType application/x-httpd-php .php .phtml`
- `Action application/x-httpd-php "/php/php.exe"`

N'oubliez pas de redémarrer le serveur, avec la commande `NET STOP APACHE` suivie de `NET START APACHE`.

Si vous voulez utiliser PHP comme module Apache, vous devez déplacer le fichier `php4ts.dll` dans le dossier `windows/system` (pour Windows 9x/Me) ou `winnt/system32` (pour Windows NT/2000), en écrasant les anciennes versions. Puis, vous devez ajouter les deux lignes suivantes dans le fichier de configuration Apache :

- `LoadModule php4_module c:/php/sapi/php4apache.dll`
- `AddType application/x-httpd-php .php .phtml`

Pour utiliser les fonctionnalités de mise en évidence du code source, créez simplement un script PHP et ajoutez le code suivant : `<?php show_source("original_php_script.php"); ?>`. Remplacez le fichier `original_php_script.php` par le fichier que vous voulez afficher : c'est la seule manière de le faire.

> **Note :** Sous Win-Apache tous les antislash des noms de chemins tels que "`c:\directory\file.ext`", doivent être convertis en slash.

CGI/ Installation pour exécution en ligne de commande

Par défaut, PHP est compilé comme une CGI. Si vous voulez que votre serveur web supporte le PHP, compiler le PHP comme un CGI permet d'obtenir de meilleures performances. Cependant, la version CGI permet aux utilisateurs de lancer des scripts PHP sous leur UID respectives. Lisez attentivement le chapitre consacré à la sécurité si vous souhaitez utiliser cette solution.

Tests

Si vous avez compilé PHP comme programme CGI, vous pouvez tester votre produit en tapant : **make test**. C'est toujours une bonne chose de tester le résultat d'une compilation. Cela vous permet de repérer des problèmes entre PHP et votre plate-forme, bien plus facilement que si vous attendez.

Performances

Si vous avez compilé PHP comme programme CGI, vous pouvez évaluer les performances de PHP 3 avec la commande **make bench**. Notez que si le safe mode est activé (par défaut), vous ne risquez pas de voir l'évaluation s'arrêter une fois les 30 secondes réglementaires écoulées. En effet, la fonction **set_time_limit()** ne peut pas être utilisée si le safe mode fonctionne. Utilisez l'option max_execution_time pour contrôler le temps d'éxécution de vos scripts. **make bench** ignore le fichier de configuration file.

> **Note : make bench** n'est disponible qu'en PHP 3.

Installation avec les serveurs fhttpd

Pour compiler PHP comme un module fhttpd, répondre "yes" à la question "Build as an fhttpd module ?" (cela correspond à l'option de configuration `--with-fhttpd=`*`DIR`* et spécifier la racine de la distribution fhttpd. Le répertoire par défaut est: `/usr/local/src/fhttpd`. Si vous utilisez fhttpd, compiler PHP en module vous permettra d'obtenir des performances supérieures, plus de contrôle et la possibilité d'exécution à distance.

Installation sur serveur Caudium

PHP 4 peut être compilé comme module Pike pour le serveur web Caudium. Notez que ce mode n'est pas supporté en PHP 3. Suivez simplement les instructions suivantes pour installer PHP 4 sur un serveur Caudium.

Exemple 2-6. Instructions d'installation Caudium

```
1.  Assurez-vous que vous avez un serveur Caudium installé avant de tenter
    l'installation PHP 4. Pour que PHP 4 fonctionne correctement, vous devez
    installer Pike 7.0.268 ou plus récent. Pour cet exemple, nous supposerons
    que vous avez installé Caudium dans le dossier /opt/caudium/server/.
2.  Renommez le dossier en php-x.y.z (où x.y.z est le numéro de version).
3.  ./configure --with-caudium=/opt/caudium/server
4.  make
5.  make install
6.  Redémarrez Caudium s'il était en fonctionnement
7.  Connectez-vous à l'interface de configuration graphique et allez
    dans le serveur virtuel auquel vous voulez ajouter le support PHP.
8.  Cliquez sur "Add Module" et recherchez puis ajoutez le module
    "PHP 4 Script Support".
9.  Si la documentation dit que 'PHP 4 interpreter isn't
    available', assurez-vous que vous avez bien redémarré le serveur.
    Si vous l'avez fait, vérifiez le fichier
    /opt/caudium/logs/debug/default.1 : il contient peut-être des
    erreurs liées à PHP4.so. De même, assurez-vous
    que caudium/server/lib/[pike-version]/PHP4.so
    est présent.
10. Configurez le module "PHP Script Support" si nécessaire.
```

Vous pouvez bien sÛr compiler votre module Caudium avec les diverses extensions disponibles. Voyez la liste complète des options de configuration pour une liste exhaustive.

> **Note :** Lorsque vous ajoutez le support MySQL à PHP 4, vous devez-vous assurer que le client MySQL normal est utilisé. Sinon, il peut y avoir des conflits avec Pike, qui dispose déjà du support MySQL. Vous pouvez le faire en spécifiant le dossier d'installation de MySQL grâce à l'option --with-mysql.

Installation avec les serveurs IIS/PWS

Cette section contient des notes sur l'installation de PHP avec IIS (Microsoft Internet Information Server) : PWS/IIS 3, PWS 4 ou plus récent et IIS 4 ou plus récent.

Windows et PWS/IIS 3

La méthode recommandée pour configurer ces serveurs est d'utiliser le fichier INF inclus dans la distribution (`php_iis_reg.inf`). Vous pouvez éditer ce fichier, pour vous assurer que les extenstions et les dossiers d'installation de PHP sont bien ceux de votre configuration. Ou alors, vous pouvez suivre les instructions suivantes :

Avertissement

ATTENTION: Ces instructions requièrent la manipulation du fichier de registry de Windows. Une erreur peut laisser votre système dans un état instable. Nous vous recommandons vivement de sauvegarder ce fichier en lieu sÛr. L'équipe de développement et les traducteurs de cette documentation ne pourront pas être tenus responsable d'un quelconque dommage qui pourrait survenir dans votre registry.

- Lancez Regedit.
- Naviguez jusqu'à : `HKEY_LOCAL_MACHINE /System /CurrentControlSet /Services /W3Svc /Parameters /ScriptMap`.
- Dans le menu "edit", sélectionnez : `New->String Value`.

- Entrez l'extension que vous voulez utiliser pour les scripts PHP. ex: `.php`
- Double cliquez sur la chaîne, et entrez le chemin jusqu'à `php.exe` dans le champ "value data". ex: `c:\php\php.exe %s %s`. Les '%s %s' sont TRES importants, PHP ne fonctionnera pas sans.
- Répétez ces instructions pour toutes les extensions que vous voulez associer aux scripts PHP.
- Naviguez jusqu'à : `HKEY_CLASSES_ROOT`
- Dans le menu edit, sélectionnez: `New->Key`.
- Donnez le nom de votre extension à la clé : ex: `.php`
- Sélectionnez le nom de la nouvelle clée dans le panneau de droite, et double cliquez dans "default value", puis entrez `phpfile`.
- Répétez ces instructions pour toutes les extensions que vous avez associées aux scripts PHP.
- Créez une autre `New->Key` sous `HKEY_CLASSES_ROOT` et nommez-la `phpfile`.
- Sélectionnez la nouvelle clé `phpfile` et dans le panneau de doite, double cliquez dans "default value" et entrez `PHP Script`.
- Faites un clic droit dans `phpfile` et sélectionnez `New->Key`, appelez-le `Shell`.
- Faites un clic droit dans `Shell` et sélectionnez `New->Key`, appelez-le `open`.
- Faites un clic droit dans `open` et sélectionnez `New->Key`, appelez-le `command`.
- Sélectionnez la nouvelle clé `command` et dans le panneau de droite, faites un double clic dans "default value", puis entrez le chemin jusqu'à `php.exe`. ex: `c:\php\php.exe -q %1`. (n'oubliez pas le `%1`).
- Quittez Regedit.
- Si vous utilisez PWS sous Windows, redémarrez pour prendre en compte la nouvelle registry.

Les utilisateurs de PWS et IIS 3 sont prêts à utiliser leur serveur. Avec IIS 3, vous pouvez utiliser un outil (http://www.genusa.com/iis/iiscfg.html) bien pratique de Steven Genusa pour configurer votre carte des scripts.

Windows et PWS 4 ou plus récent

Pour installer PHP sous Windows avec PWS 4 ou plus récent, vous avez deux options : l'une est d'avoir PHP sous forme de CGI, l'autre est d'utiliser les modules SAPI, sous forme de DLL.

Si vous optez pour le CGI, faites ceci :

- Editez le fichier `pws-php4cgi.reg` (dans le dossier `sapi`) pour indiquer la localisation de votre fichier `php.exe`. Les slash doivent être échappés. Par exemple :
  ```
  [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\w3svc\parameters\Script Map]
  ".php"="C:\\PHP\\php.exe"
  ```
- Dans le gestionnaire PWS Manager, faites un clic droit sur les dossiers qui supporteront PHP, et sélectionnez "Properties". Cochez l'option "Execute" et confirmez.

Si vous optez pour les modules ISAPI, faites ceci :

- Editez le fichier `pws-php4isapi.reg` (dans le dossier `sapi`) pour indiquer la localisation de votre fichier `php4isapi.dll`. Les slash doivent être échappés. Par exemple :
  ```
  [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\w3svc\parameters\Script Map]
  ".php"="C:\\PHP\\sapi\\php4isapi.dll"
  ```
- Dans le gestionnaire PWS Manager, faites un clic droit sur les dossiers qui supporteront PHP, et sélectionnez "Properties". Cochez l'option "Execute" et confirmez.

Windows NT/2000 et IIS 4 ou plus récent

Pour installer PHP sous Windows NT/2000 serveyr avec IIS 4 ou plus récent, vous avez deux options : l'une est d'avoir PHP sous forme de CGI, l'autre est d'utiliser les modules SAPI, sous forme de DLL.

Dans les deux cas, vous devez lancer la console "Microsoft Management" (elle peut aussi s'appeler "Internet Services Manager". Elle est située soit dans "Windows NT 4.0 Option Pack" ou dans "Control Panel=>Administrative Tools" sous Windows 2000). Puis, faites un clic droit sur votre dossier web (qui apparaitra probablement comme `Default Web Server`), et sélectionnez "Properties".

Si vous optez pour le CGI, faites ceci :

- Sous "Home Directory", "Virtual Directory", ou "Directory", cliquez sur le bouton "Configuration", et sélectionnez l'onglet "App Mappings".
- Cliquez sur "Add", puis dans la boîte "Executable", tapez : `c:\php\php.exe %s %s` (en supposant que vous avez dézippé PHP dans `c:\php\`). Vous DEVEZ ajouter `%s %s` à la fin : PHP ne fonctionnera pas correctement sans.
- Dans la boîte "Extension", tapez le nom de l'extension que vous voulez associer aux scripts PHP. Laissez "Method exclusions" vide, et cochez "Script engine". Répétez les étapes 3 et 4 pour chaque extension que vous souhaitez associer aux scripts PHP. (`.php` et `.phtml` sont les plus répandus.)
- Configurer la sécurité nécessaire (dans "Internet Service Manager"), et si votre serveur NT utilise NTFS, ajoutez les droits adéquats pour I_USR_, au dossier qui contient `php.exe`.

Si vous optez pour les modules ISAPI, faites ceci :

- Si vous ne voulez pas effectuer des authentifications HTTP avec PHP, vous pouvez (et devez) sauter cette étape. Avec ISAPI Filters, ajoutez un nouveau filtre ISAPI. Utilisez PHP comme nom de filtre, et ajoutez simplement le chemin jusqu'à `php4isapi.dll`.
- Sous "Home Directory", cliquez sur le bouton "Configuration". Ajoutez une nouvelle entrée dans "Application Mappings". Utilisez le chemin jusqu'à `php4isapi.dll` comme "Executable", indiquez ".php" comme extension, laissez "Method exclusions" vide, et cochez "Script engine".
- Arrêtez totalement IIS
- Démarrez IIS

Installation sous Netscape et iPlanet Enterprise Serveur

Pour compiler PHP avec NES ou iPlanet web, ajoutez le dossier d'installation dans l'option de configuration `--with-nsapi = DIR` . Par défaut, le dossier est `/opt/netscape/suitespot/`. Lisez aussi le fichier `/php-xxx-version/sapi/nsapi/nsapi-readme.txt` pour plus de détails.

Exemple 2-7. Installation de Netscape Enterprise sous Solaris

```
Instructions pour Sun Solaris 2.6 avec Netscape Enterprise Server 3.6
From: bhager@invacare.com
1. Installez les packages suivants depuis le serveur www.sunfreeware.com
ou un miroir ad hoc :
    flex-2_5_4a-sol26-sparc-local
    gcc-2_95_2-sol26-sparc-local
    gzip-1.2.4-sol26-sparc-local
    perl-5_005_03-sol26-sparc-local
    bison-1_25-sol26-sparc-local
    make-3_76_1-sol26-sparc-local
    m4-1_4-sol26-sparc-local
    autoconf-2.13
    automake-1.4
    mysql-3.23.24-beta (if you want mysql support)
    tar-1.13 (GNU tar)
```

```
2. Assurez-vous que le path inclut bien les dossiers nécessaires :
    PATH=.:/usr/local/bin:/usr/sbin:/usr/bin:/usr/ccs/bin
    export PATH
3. gunzip php-x.x.x.tar.gz (si vous avez une distribution .gz, ou bien allez en 4)
4. tar xvf php-x.x.x.tar
5. cd ../php-x.x.x
6. Pour les étapes suivantes, assurez-vous que /opt/netscape/suitespot/ correspond
bien à votre installation du serveur netscape. Sinon, indiquez le chemin
correct :
    /configure
    --with-mysql=/usr/local/mysql
    --with-nsapi=/opt/netscape/suitespot/
    --enable-track-vars --enable-libgcc
7. make
8. make install
```

Après avoir fait l'installation de base et lu les fichiers readme.txt, vous pouvez avoir besoin de faire des configurations supplémentaires.

D'abord, vous devez ajouter des chemins dans la variable LD_LIBRARY_PATH pour que PHP trouve toutes les librairies partagées. Le mieux est de le faire dans le script de démarrage de votre serveur Netscape. Les utilisateurs Windows peuvent probablement ignorer cette étape. Le script de démarrage est situé dans : `/path/to/server/https-servername/start`

Vous pouvez aussi avoir besoin d'éditer les fichiers de configuration qui sont situés dans : `/path/to/server/https-servername/config/`.

Exemple 2-8. Exemple de configuration pour Netscape Enterprise

```
Instructions  de configuration for Netscape Enterprise Server
From: bhager@invacare.com
1. Ajoutez les lignes suivantes dans mime.types:
    type=magnus-internal/x-httpd-php exts=php
2. Ajoutez les lignes suivantes dans obj.conf. shlib peut dépendre de
votre OS, pour Unix c'est quelque chose de proche de
/opt/netscape/suitespot/bin/libphp4.so.
    Il est conseillé de placer les lignes suivantes après les
    lignes de types mime :
    Init fn="load-
modules" funcs="php4_init,php4_close,php4_execute,php4_auth_trans" shlib="/php4/nsapiPHP4.dll"
    Init fn=php4_init errorString="Failed to initialize PHP!"
    <object name="default">
    .
    .
    .
    .#NOTE La ligne suivante doit être placée après toutes
    .#les lignes 'ObjectType'
    .# et avant les lignes 'AddLog'
    Service fn="php4_execute" type="magnus-internal/x-httpd-php"
    .
    .
    </Object>
    <Object name="x-httpd-php">
    ObjectType fn="force-type" type="magnus-internal/x-httpd-php"
    Service fn=php4_execute
    </Object>
    Configuration d'authentification
    L'authentification PHP ne peut pas être utilisée avec d'autre authentification.
    TOUTES LES FORMES D'AUTHENTIFICATION SONT PASSEES AU SCRIPT PHP.
    Pour configurer l'authentification PHP pour le serveur entier, ajoutez
    la ligne suivante :
    <Object name="default">
    AuthTrans fn=php4_auth_trans
    .
    .
    .
```

```
    .
    </Object>
    Pour configurer l'authentification PHP pour un dossier, ajoutez
    la ligne suivante :
    <Object ppath="d:\path\to\authenticated\dir\*">
    AuthTrans fn=php4_auth_trans
    </Object>
```

Si vous utilisez Netscape Enterprise 4.x, alors, il faut utiliser ceci :

Exemple 2-9. Exemple de configuration pour Netscape Enterprise 4.x

```
     Placez ces lignes après les types MIME, et tout le reste ressemble
     à l'exemple ci-dessus :
From: Graeme Hoose (GraemeHoose@BrightStation.com)
Init fn="load-modules" shlib="/path/to/server4/bin/libphp4.so"
    funcs="php4_init,php4_close,php4_execute,php4_auth_trans"
Init fn="php4_init" LateInit="yes"
```

Installation OmniHTTPd

Cette section contient des notes et conseils spécifiques à OmniHTTPd.

OmniHTTPd 2.0b1 et plus récent pour Windows

La méthode la plus simple pour configurer le serveur est :

- Step 1: Installez Omni server.
- Step 2: Faites un clic-droit sur l'icone bleue d'OmniHTTPd, sur le bureau, et sélectionnez `Properties`
- Step 3: Cliquez sur `Web Server Global Settings`
- Step 4: Dans l'onglet 'External', entrez: `virtual = .php | actual = c:\path-to-php-dir\php.exe`, et utilisez le bouton "Add".
- Step 5: Dans l'onglet `Mime`, entrez: `virtual = wwwserver/stdcgi | actual = .php` et utilisez le bouton "Add".
- Step 6: Cliquez sur `OK`.

Réptez les étapes 2 à 6 pour chaque extension que vous voulez associer à PHP.

> **Note :** Certains package OmniHTTPd sont livrés avec le support PHP déjà intégré. Vous pouvez choisir au moment de la configuration de faire un paramétrage poussé et de décocher le support PHP. Nous vous conseillons d'utiliser les dernières versions de PHP. Certains serveurs OmniHTTPd sont encore livrés avec des versions beta de PHP : il est recommandé de ne pas les installer, mais d'installer votre propre version. Si le serveur est déjà sur votre machine, vous pouvez utiliser le bouton "Replace" dans les étapes 4 et 5 pour en choisir un nouveau et à jour.

Installation Oreilly Website Pro Server

Cette section contient les conseils d'installation spécifiques à Oreilly Website Pro.

Oreilly Website Pro 2.5 et plus récent pour Windows

Cette liste décrit comment installer PHP comme CGI exécutable ou module ISAPI avec Oreilly Website Pro sous Windows.

- Editez les "Server Properties" et sélectionnez l'onglet "Mapping".
- Dans la "List" sélectionnez "Associations" et entrez le nom de l'extension voulue (".php") et le chemin jusqu'à l'exécutable (ex. c:\php\php.exe) ou la DLL ISAPI (ex. c:\php\sapi\php4isapi.dll).
- Sélectionnez "Content Types", ajoutez la même extension ".php" et entrez le "content type". Si vous choisissez la forme CGI, entrez "wwwserver/shellcgi"; si vous choisissez la forme module ISAPI, entrez "wwwserver/isapi" (sans les guillemets).

Installation Xitami

Cette section contient les conseils d'installation spécifiques à Xitami.

Xitami pour Windows

Cette liste décrit comment installer PHP comme CGI exécutable ou module ISAPI avec Xitami sous Windows.

- Assurez-vous que le serveur web fonctionne, et allez dans la console d'administration du serveur (généralement http://127.0.0.1/admin), puis cliquez sur "Configuration".
- Naviguez dans les "Filters", et ajoutez l'extension que vous souhaitez (souvent ".php") dans le champs "File extensions".
- Dans la commande "Filter", ajoutez le nom et le chemin de votre exécutable PHP (souvent `c:\php\php.exe`).
- Cliquez sur le bouton "Save".

Autres serveurs web

PHP peut être compilé pour fonctionner avec de nombreux autres serveurs web. Reportez-vous à Options particulières aux serveurs web pour une liste complète des options de configuration. Les exécutables PHP CGI sont compatibles avec la majorité des serveurs supportant les interfaces CGI.

Des problèmes?

Lisez la FAQ

Certains problèmes sont récurrents : les plus communs sont listés dans la FAQ PHP, disponible à http://www.php.net/FAQ.php.

Rapports de Bug

Si vous pensez avoir trouvé un bug dans PHP, n'oubliez pas de le signaler. L'équipe de développement PHP ne le connaît peut être pas, et si vous ne le signaler pas, vos chances de voir le bug corrigé sont nulles. Vous pouvez rapporter des bugs grâce au système de suivi, accessible à http://bugs.php.net/.

Lisez les Bugs-Dos-And-Donts (http://bugs.php.net/bugs-dos-and-donts.php) (Les bugs : ce qu'il faut faire, et ce qu'il ne faut pas faire) avant d'envoyer n'importe quel rapport!

Autres problèmes

Si vous êtes complètement bloqués, quelqu'un sur la liste de diffusion PHP pourra probablement vous aider. Essayez de consulter les archives, au cas où quelqu'un aurait déjà rencontré votre problème. Les archives sont toujours accessibles à : http://www.php.net/. Pour souscrire à la liste de diffusion, envoyez un mail vide à php-install-subscribe@lists.php.net (mailto:php-install-subscribe@lists.php.net). L'adresse de la mailing liste : `php-install@lists.php.net`.

Si vous voulez obtenir de l'aide sur la liste de diffusion PHP, essayez d'être concis et clair, et pensez à donner tous les détails sur votre environnement (OS, version de PHP, serveur web, CGI ou module, safe mode...), et n'hésitez pas à envoyer suffisamment de code pour que nous puissions reproduire l'erreur.

Chapitre 3. Configuration

Le fichier de configuration

Le fichier de configuration (appelé `php3.ini` dans la version 3.0 du PHP, et simplement `php.ini` dans la version 4.0) est lu par le PHP au démarrage. Si vous avez compilé PHP en module, le fichier n'est lu qu'une seule fois, au lancement du démon HTTP. Pour la version CGI le fichier est lu à chaque invocation.

Lorsque vous utilisez le module Apache vous pouvez aussi changer les paramètres de configurations en utilisant les directives dans les fichiers de configuration d'Apache et dans les fichiers `.htaccess`.

Dans la version 3.0, à chaque directive de configuration présente dans le fichier de configuration d'Apache correspond une directive de configuration dans le fichier `php3.ini` à l'exception des directives préfixées par "php3_".

Dans la version 4.0, il n'y a seulement que quelques directives dans le fichier de configuration d'Apache qui vous permettent de modifier la configuration de PHP.

php_value *`name value`*

Cette directive affecte une valeur à la variable spécifiée.

php_flag *`name on|off`*

Cette directive est utilisée pour activer ou désactiver une option.

php_admin_value *`name value`*

Cette directive affecte une valeur à la variable spécifiée. La directive "Admin" ne peut être utilisée que dans le fichier de configuration d'Apache, et non dans un fichier `.htaccess`.

php_admin_flag *`name on|off`*

Cette directive est utilisée pour activer ou désactiver l'option précédente.

Vous pouvez voir l'état de votre configuration en utilisant la fonction **phpinfo()**. Vous pouvez aussi accéder aux valeurs de votre configuration de manière individuelle en utilisant la fonction **get_cfg_var()**.

Directives de configuration générale

`allow_url_fopen` boolean

Cette option autorise les accès au réseau des fonctions **fopen()**. Par défaut, l'accès est autorisé aux procédures d'accès distants, avec les protocoles FTP, NNTP, et certaines extensions telles ques zlib.

Note : Cette option a été introduite immédiatement après la version 4.0.3. Pour les versions jusqu'à la 4.0.3 inclus, vous pouvez désactiver cette fonctionnalité au moment de la compilation en utilisant la configuration *`--disable-url-fopen-wrapper`*.

`asp_tags` booléen

Active l'utilisation des balises de type ASP <% %>, en plus des traditionnelles balises <?php ?> . Cela inclus l'utilisation du raccourcis <%= $value %>. Pour plus d'informations, reportez vous à inclusion dans le HTML.

Note : Le support des balises ASP a été ajouté dans la version 3.0.4.

`auto_append_file` chaîne de caractères

Spécifie le nom d'un fichier qui sera automatiquement ajouté après le fichier principal. Le fichier est inclus comme si il avait été appelé avec la fonction **include()**, donc include_path est utilisé.

Le mot réservé NONE désactive l'auto-appending.

Note : Si le script s'arrête par la fonction **exit()**, auto-append *ne fonctionnera pas*.

`auto_prepend_file` chaîne de caractères

Spécifie le nom d'un fichier qui sera automatiquement ajouté avant le fichier principal. Le fichier est inclus comme si il avait été appelé avec la fonction **include()**, donc include_path est utilisé.

Le mot réservé NONE désactive l'auto-appending.

`cgi_ext` chaîne de caractères

(NDT : aucune documentation n'est fournie).

`display_errors` booléen

Cette directive détermine si les erreurs doivent être affichées à l'écran au format HTML ou non.

`doc_root` string

Le dossier racine de PHP. Utilisée uniquement si elle est définie. Si PHP est configuré en safe mode, aucun fichier en dehors de ce dossier ne sera accessible.

`engine` boolean

Cette directive ne sert vraiment que si PHP est un module Apache. Elle sert aux sites qui veulent activer ou désactiver l'analyse des fichiers par PHP, dossier par dossier. En mettant **`php3_engine off`** au bon endroit, dans le fichier `httpd.conf`, PHP peut être activé ou desactivé.

`error_log` string

Nom du fichier oú les erreurs seront enregistrées. Si la valeur spéciale `syslog` est utilisée, les erreurs sont envoyées au système standard d'historique. Sous UNIX, c'est syslog(3) et sous Windows NT c'est l'historique d'événements. L'historique système n'est pas supporté sous Windows 95.

`error_reporting` integer

Fixe le niveau d'erreur. Ce paramètre est un entier, représentant un champs de bits. Ajoutez les valeurs suivantes pour choisir le niveau que vous désirez :

Tableau 3-1. Niveau de rapport d'erreur

valeur du bit	niveau choisi
1	erreurs normales
2	alertes normales
4	erreurs d'analyseur (parser errors)
8	alertes non critiques

Par défaut, la valeur est de 7 (erreurs normales, alertes normales et erreurs d'analyseur sont affichées).

`open_basedir` string

Limite l'espace oú PHP peut ouvrir des fichiers.

Lorsqu'un script essaie d'ouvrir un fichier avec les fonctions fopen ou gzopen (par exemple), la localisation du fichier est vérifiée. Si ce fichier est hors du dossier cité dans cette directive, PHP refusera de l'ouvrir. Tous les liens symboliques sont résolus, et subissent aussi la restriction.

La valeurs spéciale . indique que le dossier courant du script est utilisé comme open_basedir.

Sous Windows, séparez les noms de dossiers par un point virgule (;). Sur les autres systèmes, séparez les noms de dossiers par des deux points (:). Lorsque PHP est un module Apache, la valeur de la directive open_basedir des dossiers parents sont automatiquement hérités par les fils.

Note : Le support pour les dossiers multiples a été ajouté dans 3.0.7.

La valeur par défaut est : libre accès à tous les fichiers.

gpc_order chaîne de caractères

Etablit l'ordre de préséance des méthodes GET/POST/COOKIE. Par défaut, cette directive est établie à "GPC". En affectant "GP" à cette directive, PHP ignorera les cookies, et écrasera toute méthode GET utilisée par une méthode POST avec des variables du même nom.

ignore_user_abort chaîne de caractères

Désactivée par défaut. Si cette directive est activée, alors tous les scripts lancés iront jusqu'à leur terme, même si le client se déconnecte en plein milieu. Voir aussi la fonction **ignore_user_abort()**.

include_path string

Spécifie une liste de dossier oú les fonctions **require()**, **include()** et fopen_with_path (NDtraducteur : cette fonction semble avoir disparue) iront chercher les fichiers. Le format est le même que celui de la variable d'environnement PATH : une liste de dossiers, séparés par des deux points (:) sous UNIX, et des points virgules (;), sous Windows.

Exemple 3-1. UNIX include_path

```
include_path=.:/home/httpd/php-lib
```

Exemple 3-2. Windows include_path

```
include_path=".;c:\www\phplib"
```

La valeur par défaut pour cette directive est `.`, c'est à dire le dossier courant.

isapi_ext string

(Aucune documentation n'est fournie)

log_errors boolean

Indique oú les messages d'erreur générés doivent être écrits. Cette fonction est spécifique aux serveurs.

magic_quotes_gpc boolean

Fixe le mode magic_quotes pour les opérations GPC (Get/Post/Cookie). Lorsque magic_quotes est activé, tous les caractères ' (guillemets simples), " (guillemets doubles), \ (antislash) et NUL sont échappés avec un antislash. Si magic_quotes_sybase fonctionne aussi, les guillemets simples seront échappés avec un autre guillemet simple, plutôt qu'un antislash.

magic_quotes_runtime boolean

Si *magic_quotes_runtime* est activé, toutes les fonctions qui retournent des données d'une source externe, y compris les bases de données et les fichiers texte, verront leur guillemets échappés avec un antislash. Si *magic_quotes_sybase* est aussi activé, les guillemets simples seront échappés avec un autre guillemet simple, plutôt qu'un antislash.

magic_quotes_sybase boolean

Si *magic_quotes_sybase* est activé, les guillemets simples seront échappés avec un autre guillemet simple, plutôt qu'un antislash, si *magic_quotes_gpc* ou *magic_quotes_runtime* est activé.

max_execution_time integer

Fixe le temps maximal d'éxécution d'un script, en secondes. Cela permet d'éviter que des scripts en boucles infinies ne saturent le serveur.

memory_limit entier

Grâce à cette option, vous pouvez donner la quantité maximale de mémoire qu'un script peut allouer. Cela permet de reserver toute la mémoire d'un serveur à un seul script.

nsapi_ext chaîne de caractères

Aucune documentation n'est fournie.

register_globals boolean

Cette option active l'enregistrement des variables EGPCS (Environement, GET, POST, Cookie, Serveur), en tant que variables globales. Vous pouvez désactiver cette fonction si vous ne voulez pas truffer vos scripts avec des valeurs utilisateurs. Cette option est surtout utile lorsqu'elle est utilisée conjointement avec track_vars - dans ce cas, vous pouvez accéder à toutes les variables EGPCS grâce aux tableaux `$HTTP_ENV_VARS`, `$HTTP_GET_VARS`, `$HTTP_POST_VARS`, `$HTTP_COOKIE_VARS`, et `$HTTP_SERVER_VARS`.

short_open_tag booléen

Active ou désactive l'utilisation des balises courtes, (**<? ?>**). Si vous voulez utiliser PHP et XML en même temps, vous devez désactiver cette option. Si cette option est désactivée, vous devez utiliser la forme longue des tags, (**<?php ?>**).

sql.safe_mode booléen

Aucune documentation n'est fournie.

track_errors booléen

Si cette option est activée, le dernier message d'erreur sera placé dans la variable globale $php_errormsg.

track_vars booléen

Si cette option est activée, lors de l'appel des méthodes GET, POST et l'utilisation des cookies, les variables sont disponibles dans des tableaux associatifs globaux appelés respectivement $HTTP_GET_VARS, $HTTP_POST_VARS et $HTTP_COOKIE_VARS.

upload_tmp_dir chaîne de caractères

Indique le répertoire utilisé lors du chargement d'un fichier sur un serveur. Ce répertoire doit être accessible en lecture pour l'utilisateur qui lance le script PHP.

user_dir chaîne de caractères

Répertoire oú sont stockés les fichiers PHP dans le répertoire d'un utilisateur. Par exemple, `public_html`.

warn_plus_overloading booléen

Si cette option est activée, PHP émet un warning lorsque l'opérateur plus (+) est utilisé sur une chaîne de caractères. Cela permet de réperer plus facilement les scripts qui doivent être réécrits en utilisant l'opérateur de concaténation (`.`) plutôt que l'opérateur plus.

Configuration des directives concernant le mail

SMTP chaîne de caractères

Sous Windows, adresse IP ou nom que PHP doit utiliser pour envoyer du mail avec la fonction **mail()**.

sendmail_from chaîne de caractères

Sous Windows, valeur du champs "From:" qui doit être utilisée lors de l'envoie de mail.

sendmail_path chaîne de caractères

Localisation du programme de **sendmail**, habituellement `/usr/sbin/sendmail` ou `/usr/lib/sendmail`. **configure** essaye de repérer la présence de sendmail par lui même, et affecte ce résultat par défaut. En cas de problème de localisation, vous pouvez établir une nouvelle valeur par défaut ici.

Tout système n'utilisant pas **sendmail** doit établir cette directive à la valeur chemin du programme de substitution qui remplace le serveur de mail, si celui-ci existe, par exemple, Qmail (http://www.qmail.org/). Dans ce cas la, vous devez mettre: `/var/qmail/bin/sendmail`.

Directives de configuration du "Safe Mode"

safe_mode booléen

Cette directive active ou désactive l'option "safe mode". Lisez le chapitre sécurité pour plus d'informations.

safe_mode_exec_dir chaîne de caractères

Si l'option "SAFE MODE" est activée, **system()** et les autres fonctions exécutant des programmes systèmes refusent de se lancer si ces programmes ne sont pas placés dans ce répertoire.

Directives de configuration de débbugage.

debugger.host chaîne de caractères

Adresse IP ou nom de l'hôte utilisé pour le déboggage.

debugger.port chaîne de caractères

Numéro du port utilisé pour le déboggage.

debugger.enabled booléen

Activation ou désactivation du debugger.

Directives de chargement des extensions

enable_dl booléen

Cette directive n'est réellement utile que dans le cas d'une compilation comme module Apache. Vous pouvez activer le chargement dynamique des extensions avec la fonction **dl()**, et cela de manière locale à chaque serveur virtuel ou à chaque répertoire.

La principale raison qui pousse à désactiver le chargement dynamique est un problème de sécurité. Lorsque le chargement dynamique est activé, il est possible d'ignorer les directives safe mode ou "open_basedir".

Par défaut, il est possible d'utiliser le chargement dynamique, sauf lorsque la directive safe mode est activée. En effet, il est alors impossible d'utiliser la fonction **dl()**.

extension_dir chaîne de caractères

Définit le répertoire dans lequel le PHP doit chercher les extensions lors du chargement dynamique.

extension chaîne de caractères

Définit les extensions qui doivent être chargées lors du démarrage du PHP.

MySQL Configuration Directives

mysql.allow_persistent booléen

Active ou désactive les connexions persistentes à la base de données MySQL.

mysql.default_host chaîne de caractères

Adresse par défaut du serveur, à utiliser lors de la connexion à un serveur MySQL, si aucun hôte n'est spécifié.

mysql.default_user chaîne de caractères

Utilisateur par défaut, à utiliser lors de la connexion à un serveur MySQL, si aucun utilisateur n'est spécifié.

mysql.default_password chaîne de caractères

Mot de passe par défaut, à utiliser lors de la connexion à un serveur MySQL, si aucun mot de passe n'est spécifié.

mysql.max_persistent entier

Nombre maximum de connexions persistantes à une base de donnée MySQL, par processus.

mysql.max_links entier

Nombre de connexion maximum à une base de donnée MySQL, par processus, incluant les connexions persistantes

Directives de configuration mSQL

msql.allow_persistent booléen

Active ou désactive les connexions persistentes à la base de données mSQL.

msql.max_persistent entier

Nombre maximum de connexions persistantes à une base de donnée mSQL, par processus.

msql.max_links entier

Nombre maximum de connexions à une base de donnée mSQL, par processus, incluant les connexions persistantes.

Directives de configuration Postgres

pgsql.allow_persistent booléen

Active ou désactive les connexions persistantes à la base de données Postgres.

pgsql.max_persistent entier

Nombre maximum de connexions persistantes à une base de données Postgres, par processus.

pgsql.max_links entier

Nombre maximal de connexions à une base de donnée Postgres, par processus, incluant les connexions persistantes.

Directives de configuration SESAM

sesam_oml string

Nom de la librairieBS2000 PLAM contenant les pilotes de connexion SESAM. Obligatoire pour les fonctions SESAM. La librairie BS2000 PLAM doit être configurée avec ACCESS=READ,SHARE=YES car elle doit être accessible à l'utilisateur Apache.

sesam_configfile string

Nom du fichier de configuration de l'application SESAM. Obligatoire pour les fonctions SESAM. Le fichier BS2000 doit être accessible être accessible à l'utilisateur Apache.

Le fichier de configuration de l'application va contenir la configuration sur le schéma suivant (voir le manuel SESAM) :

```
CNF=B
NAM=K
NOTYPE
```

sesam_messagecatalog string

Nom du catalogue de messages SESAM. Dans la plus part des cas, cette directive n'est pas nécessaire. Seulement, si le fichier de messages SESAM n'est pas installé dans la table de messages BS2000, il faut indiquer sa localisation avec cette directive.

Le catalogue de messages doit être paramétré avec ACCESS=READ,SHARE=YES car elle doit être accessible à l'utilisateur Apache.

Directives de configuration Sybase

sybase.allow_persistent booléen

Active ou désactive les connexions persistentes à la base de données Sybase.

sybase.max_persistent entier

Nombre maximum de connexions persistantes à une base de données Sybase par processus.

sybase.max_links entier

Nombre maximum de connexions à une base de données Sybase, par processus, incluant les connexions persistantes.

Sybase-CT Configuration Directives

sybct.allow_persistent booléen

Active ou désactive les connexions persistantes à la base de données Sybase-CT. Par défaut, cette option est activée.

sybct.max_persistent entier

Nombre maximum de connexions persistantes à une base de données Sybase-CT par processus. Par défaut, cette option est à -1, ce qui signifie que le nombre de connexion est illimité.

sybct.max_links entier

Nombre maximum de connexions à une base de données Sybase-CT, par processus, incluant les connexions persistantes. Par défaut, cette option est à -1, ce qui signifie que le nombre de connexions est illimitée.

sybct.min_server_severity entier

Les messages en provenance du serveur d'un niveau d'erreur égal à sybct.min_server_severity seront considérés comme des alertes (warnings). Cette valeur peut être modifiée à l'intérieur du script en appelant la fonction sybase_min_server_severity (NDtraducteur : cette fonction semble ne pas exister). Par défaut, cette valeur vaut 10.

sybct.min_client_severity entier

Les messages en provenance de la librairie client avec un niveau d'erreur égal ou supérieur à sybct.min_client_severity seront considérés comme des alertes. Cette valeur peut être modifiée à l'intérieur du script en appelant la fonction sybase_min_client_severity (NDtraducteur : cette fonction semble ne pas exister). Par défaut, cette valeur vaut 10, ce qui annule tout rapport d'erreur.

sybct.login_timeout entier

Délai de validité d'une tentative de connexion. Il est à noter que si max_execution_time est dépassé avant que la connexion n'éxpire, le script sera terminé avant le message d'erreur. Par défaut, cette valeur vaut 1 minute.

sybct.timeout entier

Temps maximum en secondes avant qu'une tentative de requête "select_db" ou "query" non aboutie renvoie une erreur. Il est à noter que si max_execution_time est dépassé avant que la requête n'éxpire, votre script sera terminé avant le message d'erreur. Par défaut, il n'y a pas de limite.

sybct.hostname chaîne de caractères

Nom de l'hôte à partir duquel vous vous connectez, afin d'être affiché par la fonction sp_who (NDtraducteur : cette fonction semble ne pas exister). Par défaut, cette valeur égale à 0.

Directives de configuration Informix

ifx.allow_persistent booléen

Active les connexions persistantes à une base de données Informix.

ifx.max_persistent entier

Nombre maximum de connexions persistantes à une base de données Informix, par processus.

ifx.max_links entier

Nombre maximum de connexions à une base de données Informix par processus, en incluant les connexions persistantes.

ifx.default_host chaîne de caractères

Hôte par défaut oú se connecter si aucun hôte n'est spécifié par les fonctions **ifx_connect()** ou **ifx_pconnect()**.

ifx.default_user chaîne de caractères

Utilisateur par défaut si aucun utilisateur n'est spécifié par les fonctions **ifx_connect()** ou **ifx_pconnect()**.

ifx.default_password chaîne de caractères

Mot de passe par défaut si aucun mot de passe n'est spécifié par les fonctions **ifx_connect()** ou **ifx_pconnect()**.

ifx.blobinfile booléen

Lorsque cette option est activée, les colonnes de type "blob" seront retournées dans un fichier. Par défaut, elles seront retournées en mémoire. Il est possible de modifier dynamiquement cette valeur grâce à la fonction **ifx_blobinfile_mode()**.

ifx.textasvarchar booléen

Lorsque cette option est activée, les colonnes de type "TEXT" seront retournées dans une chaîne de caractères. Par défaut, elles seront retournées en mémoire. Il est possible de modifier dynamiquement cette valeur grâce à la fonction **ifx_textasvarchar()**.

ifx.byteasvarchar booléen

Lorsque cette option est activée, les colonnes de type "BYTE" seront retournées dans une chaîne de caractères. Par défaut, elles seront retournées en mémoire. Il est possible de modifier dynamiquement cette valeur grâce à la fonction **ifx_textasvarchar()**.

ifx.charasvarchar booléen

Lorsque cette option est activée, les espaces en fin de chaîne de caractères seront conservés lors d'une commande FETCH.

ifx.nullformat booléen

Lorsque cette option est activée, les colonnes de valeur `NULL` seront retournées comme des chaînes de caractères vides. Il est possible de modifier dynamiquement cette valeur grâce à la fonction **ifx_nullformat()**.

Directives de configuration pour les calculs mathématiques.

bcmath.scale entier

Nombre de chiffres après la virgule pour toutes les fonctions de précision mathématique arbitraire.

Directives de configuration du navigateur.

browscap chaîne de caractères

Nom du fichier de descriptif des clients HTML. Voir aussi **get_browser()**.

Directives de configuration du driver ODBC unifié

uodbc.default_db chaîne de caractères

Source de données ODBC à utiliser par défaut avec les fonctions **odbc_connect()** ou **odbc_pconnect()**.

uodbc.default_user chaîne de caractères

Nom d'utilisateur défaut avec les fonctions **odbc_connect()** ou **odbc_pconnect()**.

uodbc.default_pw chaîne de caractères

Mot de passe par défaut dans les fonctions **odbc_connect()** ou **odbc_pconnect()**.

uodbc.allow_persistent booléen

Cette option active ou désactive les connexions persistantes à la base de données, via le canal ODBC.

uodbc.max_persistent entier

Nombre maximum de connexions persistantes autorisées à la base de données.

uodbc.max_links entier

Nombre maximum de connexions (persistantes ou non), par processus, à la base de données.

Chapitre 4. Sécurité

PHP est un langage puissant et l'interpréteur, qu'il soit inclus dans le serveur web ou bien compilé en version CGI, est capable d'accéder aux fichiers, d'exécuter des commandes et d'ouvrir des connexions réseaux. Toutes ces propriétés rendent fragile la sécurité d'un serveur web. Le langage PHP a été pensé afin d'être un langage beaucoup plus sécurisé pour écrire des CGI que le Perl ou le langage C. De plus, une sélection rigoureuse des options de compilation et d'exécution vous permettra d'obtenir un équilibre parfait entre liberté et sécurité.

Etant donné qu'il y a de nombreux moyens d'utiliser le langage PHP, il y a de nombreuses directives de configuration afin d'en contrôler le comportement. Un grand nombre d'options permettent d'utiliser le PHP dans de nombreuses situations, mais cela signifie aussi qu'il y a certaines combinaisons d'options de compilation et d'exécution qui fragilisent la sécurité du serveur. Ce chapitre explique comme les différentes options de configurations peuvent être combinées, tout en conservant une sécurité maximum.

La flexibilité de configuration de PHP est épaulée par la flexibilité du code. PHP peut être compilé pour constituer une application serveur complète, avec toutes les fonctionnalités d'un shell, ou il peut encore être utilisé comme simple SSI (server side include) avec peu de risque, dans un environnement à sécurité renforcée. Comment créer cet environnement et le sécuriser est largement à la charge du développeur PHP.

Ce chapitre commence par expliquer les différentes options de configuration et les situations dans lesquelles elles peuvent être utilisées en toute sécurité. Puis, viennent les considérations de niveaux de sécurité, et les conseils généraux.

Binaires CGI

Faiblesses connues

Utiliser le PHP comme un CGI exécutable vient la majorité du temps du fait que l'on ne veut pas l'utiliser comme un module du serveur web, (comme Apache), ou bien que l'on souhaite l'utiliser en combinaison d'un CGI complémentaire, afin de créer un environnement de script sécurisé (en utilisant des techniques de chroot ou setuid). Une telle décision signifie habituellement que vous installez votre exécutable dans le répertoire cgi-bin de votre serveur web. CERT CA-96.11 (http://www.cert.org/advisories/CA-96.11.interpreters_in_cgi_bin_dir.html) recommande effectivement de placer l'interpréteur à l'intérieur du répertoire cgi-bin. Même si le binaire PHP peut être utilisé comme interpréteur indépendant, PHP a été pensé afin de rendre impossible les attaques que ce type d'installation induit.

- Accès au système de fichier: `http://ma.machine/cgi-bin/php?/etc/passwd`

  Lorsque la requête est passée dans une url, après le point d'interrogation (?), elle est envoyée à l'interpréteur comme une ligne de commande par l'interface CGI. Habituellement, l'interpréteur ouvre le fichier spécifié et l'exécute.

  Lorsqu'il est invoqué comme exécutable CGI, le PHP refuse d'interpréter les arguments de la ligne de commande.

- Accès d'un document web sur le serveur : `http://my.host/cgi-bin/php/secret/doc.html`

  Le "path information" dans l'url, situé juste après le nom du binaire PHP, `/secret/doc.html` est utilisé par convention pour spécifier le nom du fichier qui doit être ouvert et interprété par le programe CGI. Habituellement, des directives de configuration du serveur web (pour le serveur Apache: Action) sont utilisées pour rediriger les requêtes afin d'obtenir un document `http://my.host/secret/script.php` par l'interpréteur PHP. Dans une telle configuration, le serveur web vérifie d'abord s'il a accès au répertoire `/secret`, et après cette vérification redirige la requête vers `http://my.host/cgi-bin/php/secret/script.php`. Malheureusement, si la requête est faite directement sous cette forme, aucune vérification d'accès n'est faite par le serveur web pour le fichier `/secret/script.php`, mais uniquement pour le fichier `/cgi-bin/php`. De cette manière, n'importe quel utilisateur qui peut accéder au fichier `/cgi-bin/php` peut aussi accéder aux documents protégés sur le serveur web.

  Avec le PHP, l'option de compilation --enable-force-cgi-redirect et les options d'exécution doc_root et user_dir peuvent être utilisées pour prévenir ce genre d'attaques, si des restrictions d'accès sont appliquées sur les documents du serveur. Voir ci-dessous pour des explications plus complètes sur les différentes combinaisons.

Cas 1: Tous les fichiers sont publics

Si votre serveur n'a aucun document dont l'accès est restreint par un mot de passe ou un système de vérification de l'adresse IP, vous n'avez aucun besoin de ce type de configuration. Si votre serveur web ne permet pas les redirections, ou si votre serveur web n'a aucun besoin de communiquer avec le binaire PHP de manière sécurisée, vous pouvez utiliser

l'option de compilation --disable-force-cgi-redirect. Vous devez quand même vérifier qu'aucun script ne fait appel au PHP, de manière directe, `http://my.host/cgi-bin/php/dir/script.php` ou bien de manière indirecte, par redirection, `http://my.host/dir/script.php`.

Les redirections peuvent être configurées dans les fichiers de configuration d'Apache en utilisant les directives "AddHandler" et "Action" (voir ci-dessous).

Cas 2: Utilisation de la directive de compilation --enable-force-cgi-redirect

Cette option de compilation prévient quiconque d'appeler directement un script avec l'url `http://my.host/cgi-bin/php/secretdir/script.php`. Dans ce cas là, PHP parsera le fichier uniquement s'il y a eu redirection.

Habituellement, le serveur web Apache réalise une redirection grâce aux directives suivantes :

```
Action.php-script /cgi-bin/php
AddHandler.php-script .php
```

Cette option a uniquement été testée avec Apache et compte sur Apache pour affecter la variable d'environnement non-standart REDIRECT_STATUS pour les requêtes redirigées. Dans le cas où votre serveur web ne supporte pas le renseignement du PHP, pour savoir si la requête a été redirigée ou non, vous ne pouvez pas utiliser cette option de compilation. Vous devez alors utiliser une des autres méthodes d'exploitation de la version binaire CGI du PHP, comme exposé ci-dessous.

Cas 3: Utilisation du "doc_root" ou du "user_dir"

Ajouter un contenu interactif dans votre serveur web, comme des scripts ou des exécutables, est souvent considéré comme une pratique non-sécurisée. Si, par erreur, le script n'est pas exécuté mais affiché comme une page HTML classique, il peut en résulter un vol de propriété intellectuelle ou des problèmes de sécurité à propos des mots de passe notamment. Donc, la majorité des administrateurs préfèrent mettre en place un répertoire spécial pour les scripts qui soit uniquement accessible par le biais du binaire CGI du PHP, et donc, tous les fichiers de ce répertoire seront interprétés et non affichés tels quels.

Aussi, si vous ne pouvez pas utiliser la méthode présentée ci-dessus, il est nécessaire de mettre en place un répertoire "doc_root" différent de votre répertoire "document root" de votre serveur web.

Vous pouvez utiliser la directive doc_root dans le fichier de configuration, ou vous pouvez affecter la variable d'environnement PHP_DOCUMENT_ROOT. Si cette variable d'environnement est affectée, le binaire CGI du PHP construira toujours le nom de fichier à ouvrir avec *`doc_root`* et le "path information" de la requête, et donc vous serez sÛr qu'aucun script n'est exécuté en dehors du répertoire prédéfini (à l'exception du répertoire désigné par la directive *`user_dir`* Voir ci-dessous).

Une autre option possible ici est la directive user_dir. Lorsque la directive n'est pas activée, seuls les fichiers contenus dans le répertoire *`doc_root`* peuvent être ouverts. Ouvrir un fichier possédant l'url `http://my.host/~user/doc.php` ne correspond pas à l'ouverture d'un fichier sous le répertoire racine de l'utilisateur mais à l'ouverture du fichier `~user/doc.php` sous le repertoire "doc_root" (oui, un répertoire commence par un tilde [~]).

Si la directive "user_dir" est activée à la valeur `public_php` par exemple, une requête du type `http://my.host/~user/doc.php` ouvrira un fichier appelé `doc.php` sous le répertoire appelé `public_php` sous le répertoire racine de l'utilisateur. Si le répertoire racine des utilisateurs est `/home/user`, le fichier exécuté sera `/home/user/public_php/doc.php`.

`user_dir` et *`doc_root`* sont deux directives totalement indépendantes et donc vous pouvez contrôler l'accès au répertoire "document root" séparément des répertoires "user directory".

Cas 4: L'exécutable PHP à l'extérieur de l'arborescence du serveur

Une solution extrêmement sécurisée consiste à mettre l'exécutable PHP à l'extérieur de l'arborescence du serveur web. Dans le répertoire `/usr/local/bin`, par exemple. Le problème de cette méthode est que vous aurez à rajouter la ligne suivante :

```
#!/usr/local/bin/php
```

dans tous les fichiers contenant des tags PHP. Vous devrez aussi rendre le binaire PHP exécutable. Dans ce cas-là, traitez le fichier exactement comme si vous aviez un autre script écrit en Perl ou en sh ou en un autre langage de script qui utilise `#!` comme mécanisme pour lancer l'interpréteur lui-même.

Pour que l'exécutable PHP prenne en compte les variables d'environnement PATH_INFO et PATH_TRANSLATED correctement avec cette configuration, vous devez utiliser l'option de compilation --enable-discard-path.

Module Apache

Lorsque le PHP est compilé en tant que module Apache, ce module hérite des permissions accordées à l'utilisateur faisant tourner Apache (par défaut, l'utilisateur "nobody"). Par exemple, si vous utilisez PHP pour accéder à une base de données, à moins que la base n'ait un système de droits d'accès interne, vous devrez rendre la base accessible à l'utilisateur "nobody". Cela signifie qu'un script mal intentionné peut accéder à la base, la modifier sans authentification. Il est aussi possible qu'un robot accède à la page d'administration, et détruise toutes les pages. Vous devez aussi protéger vos bases de données avec les autorisations Apache, ou définir votre propre modèle d'accès avec LDAP, .htaccess, etc... et inclure ce code dans tous vos scripts PHP.

Souvent, lorsqu'on a établi les droits de l'utilisateur PHP (ici, l'utilisateur Apache) pour minimiser les risques, on s'aperçoit que PHP ne peut plus écrire des virus dans les fichiers des utilisateurs. Ou encore, de modifier une base de données privée. Il est aussi incapable de modifier des fichiers qu'il devrait pouvoir modifier, ou effectuer certaines transactions.

Une erreur fréquente de sécurité est de donner à l'utilisateur Apache les droits de superadministrateur.

Donner de telles permissions à l'utilisateur Apache est extrêmement dangereux, et peut compromettre tout le système, telle que l'utilisation des `sudo` ou du `chroot`. Pour les professionnels de la sécurité, une telle utilisation est à exclure d'office.

Sécurité des fichiers

PHP est soumis aux règles de sécurité intrinsèques de la plupart des systèmes serveurs : il respecte notamment les droits des fichiers et des dossiers. Une attention particulière doit être portée aux fichiers ou dossiers qui sont accessibles à tout le monde, afin de s'assurer qu'ils ne divulguent pas d'informations critiques.

Puisque PHP a été fait pour permettre aux utilisateurs d'accéder aux fichiers, il est possible de créer un script qui vous permet de lire des fichiers tels que /etc/password, de modifier les connexions ethernet, lancer des impressions de documents, etc... Cela implique notamment que vous devez-vous assurer que les fichiers accédés par les scripts sont bien ceux qu'il faut.

Considérez le script suivant, où l'utilisateur indique qu'il souhaite effacer un fichier dans son dossier racine. Nous supposons que PHP est utilisé comme interface web pour gérer les fichiers, et que l'utilisateur Apache est autorisé à effacer les fichiers dans le dossier racine des utilisateurs.

Exemple 4-1. Une erreur de vérification de variable conduit à ...

```
<?php
// efface un fichier dans un dossier racine
$username = $user_submitted_name;
$homedir = "/home/$username";
$file_to_delete = "$userfile";
unlink ($homedir/$userfile);
echo "$file_to_delete a été effacé!";
?>
```

Etant donné que le nom de l'utilisateur est à fournir, des intrus peuvent envoyer un nom d'utilisateur autre que le leur, et effacer des documents dans les comptes des autres utilisateurs. Dans ce cas, vous souhaiterez utiliser une autre forme d'authentification. Considérez ce qui pourrait se passer si les utilisateurs passent "../etc/" et "passwd" comme arguments! Le code serait éxécuté tel que :

Exemple 4-2. Une attaque du système de fichiers!

```
<?php
// efface un fichier n'importe où sur le disque dur,
// où l'utilisateur PHP a accès. Si PHP a un accès root :
$username = "../etc/";
$homedir = "/home/../etc/";
$file_to_delete = "passwd";
unlink ("/home/../etc/passwd");
echo "/home/../etc/passwd" has been deleted!";
?>
```

Il y a deux mesures primordiales à prendre pour éviter ces manoeuvres :

- Limiter les permissions du l'utilisateur web PHP.
- Vérifier toutes les variables liées aux chemins et aux fichiers qui sont fournie.

Voici le script renforcé :

Exemple 4-3. Une vérification renforcée

```
<?php
// Efface un fichier sur le disque où l'utilisateur à le droit d'aller
$username = get_env("REMOTE_USER");
// utilise un mécanisme d'authentification
$homedir = "/home/$username";
$file_to_delete = basename("$userfile");
// supprime le chemin excédentaire
unlink ($homedir/$file_to_delete);
$fp = fopen("/home/logging/filedelete.log","+a"); //note l'effacement
$logstring = "$HTTP_REMOTE_USER $homedir $file_to_delete";
fputs ($fp, $logstring);
fclose($fp);
echo "$file_to_delete a été éffacé!";
?>
```

Vous pouvez-vous protéger avec une vérification telle que :

Exemple 4-4. Vérification de noms de fichier sécurisée

```
<?php
$username = $HTTP_REMOTE_USER;
$homedir = "/home/$username";
if (!ereg('^[^./][^/]*$', $userfile))
    die('Erreur de nom de fichier'); //meurt, ne pas traiter!
//etc...
?>
```

Suivant votre système d'exploitation, vous devrez protéger un grand nombre de fichiers, notamment les entrées de périphériques, (/dev/ ou COM1), les fichiers de configuration (fichiers /etc/ et .ini), les lieux de stockages d'informations (/home/, My Documents), etc. Pour cette raison, il est généralement plus sÛr d'établir une politique qui interdit TOUT sauf ce que vous autorisez.

Rapport d'erreur

Une tactique d'attaque standard consiste à faire faire des erreurs au système, et lire les variables d'environnement et de contexte qui sont retournées. Cela permet au pirate de lire de nombreuses informations sur le serveur, et de détecter des faiblesses du serveur.

Les erreurs PHP qui sont normalement retournées peuvent être très pratiques pour un développeur qui essaie de débugger un script, car elles donnent de précieux renseignements tels que quelle fonction a échoué, quel fichier n'a pas été trouvé,

quel script PHP a buggé, et quelle ligne est en faute. Toutes ces informations sont exploitables. Il est commun aux développeurs PHP d'utiliser les fonctions **show_source()**, **highlight_string()**, ou **highlight_file()** comme outils de débuggage, mais sur un site en production, cela expose des variables cachées, des syntaxes non vérifiées ou d'autres informations dangereuses.

Par exemple, le style d'erreur indique sur quel système PHP fonctionne. Si un pirate affiche une page html, et essaye de la tester (pour rechercher des faiblesses du système), il peut déterminer sur quel système PHP a été compilé.

Une erreur de fonction indique si un système supporte une base de données spécifique, ou bien indique comment la page a été générée. Cela peut orienter l'intrus vers les ports de cette base de données ou bien vers une attaque liée à cette application. En envoyant des données erronées, par exemple, un pirate peut déterminer l'ordre d'authentification dans un script (à partir des lignes d'erreurs), et d'essayer de les exploiter ailleurs, dans le script.

Une erreur de fichier, ou une erreur générale PHP peut indiquer quelles sont les permissions du serveur web, ainsi que la structure et l'organisation des fichiers. Les gestionnaires d'erreurs utilisateurs peuvent aussi aggraver ce problème, en permettant l'exploitation facile d'informations préalablement cachées.

Il y a trois solutions majeures à ces problèmes : la première est de scruter toutes les fonctions, et essayer de traiter toutes les erreurs. La deuxième est d'inactiver le rapport d'erreur, dès que le script est en production. La troisième est d'utiliser les fonctions de gestions des erreurs. Suivant votre politique de sécurité, vous pouvez utiliser un panachage savant des trois méthodes.

Données transmises par les internautes

Les plus grandes faiblesses de nombreux programmes PHP ne viennent pas du langage lui-même, mais de son utilisation en omettant les caractéristiques de sécurité. Pour cette raison, vous devez toujours prendre le temps de prendre en compte les implications d'une fonction, et de cerner toutes les applications d'une utilisation exotiques des paramètres.

Exemple 4-5. Utilisation dangereuse de variables

```
<?php
// efface un fichier à la racine d'un utilisateur... ou peut être
// de quelqu'un d'autre?
unlink ($evil_var);
// Note l'accès de ce fichier ... ou pas?
fputs ($fp, $evil_var);
// Exécute une commande triviale... ou pas?
system ($evil_var);
exec ($evil_var);
?>
```

Il est vivement recommandé d'examiner minutieusement votre code pour vous assurer qu'il n'y a pas de variables envoyées par le client web, et qui ne sont pas suffisamment vérifiées avant utilisation.

- Est-ce que ce script n'affectera que les fichiers prévus?
- Est-il possible que des valeurs incohérentes soient exploitées ici?
- Est-ce que ce script peut être utilisé dans un but différent?
- Est-ce que ce script peut être utilisé malicieusement, en conjonction avec d'autres?
- Est-ce que toutes les actions seront notées?

En répondant de manière adéquate à ces questions lors de l'écriture de vos scripts (plutôt qu'après), vous éviterez une réécriture inopportune pour raison de sécurité. En commençant vos projets avec ces recommandations en tête, vous ne garantirez pas la sécurité de votre système, mais vous contribuerez à l'améliorer.

Vous pouvez aussi envisager de supprimer l'acquisition automatique des variables d'environnement, les guillemets magiques (magic_quotes), ou encore toute option qui pourrait vous conduire à mésévaluer la validité, la source ou la valeur d'une variable. En travaillant avec error_reporting(E_ALL), vous pouvez être averti que certaines variables sont utilisées avant d'être exploitées, ou vérifiées (et donc, vous pourrez traiter des valeurs exotiques à la source).

Considérations générales

Un système complètement sécurisé est virtuellement impossible. De ce fait, une approche qui équilibre les risques et l'ergonomie est souvent retenue. Si toutes les variables envoyées par l'utilisateur nécessitaient deux formes d'identification biométriques (comme par exemple un scanner de la rétine, et une empreinte digitale), vous pourriez avoir un niveau de sécurité élevé. Cela prendrait aussi une demi-heure pour remplir les conditions de sécurité, ce qui pousserait les utilisateurs à éviter d'utiliser votre système.

La meilleure sécurité est celle qui protège votre système, sans empêcher les utilisateurs de faire leur travail. En général, les attaques exploitent des trous de sécurité entre diverses couches d'identification : cet empilement devient trop complexe, et finalement, peu fiable.

Une phrase qui vaut la peine d'être retenue : un système est aussi fiable que son maillon le plus faible. Si toutes les transactions sont exhaustivement notées (heure, lieu, type, etc...) mais que l'utilisateur n'est authentifié que par un cookie, la validité de votre système de surveillance est intimement liée à la validité du cookie (et donc, sévèrement réduite).

Lors de vos tests, gardez à l'esprit que vous ne pourrez pas tester toutes les configurations, mais probablement les plus simples. Les données en entrées auxquelles vous pouvez vous attendre ne seront rien comparées aux données incohérentes d'un employé négligent, un pirate disposant d'autant de temps qu'il veut, ou du chat de la maison marchant sur le clavier. Il est donc bon de regarder le code logiquement, de chercher où des données incohérentes peuvent être introduites, modifiées, réduites ou amplifiées.

L'Internet est rempli de personnes qui tentent de se faire une renommée en piratant vos programmes, en bloquant votre site, en envoyant des contenus inappropriés, qui rendent vos journées si "spéciales". Peut importe que vous ayez un grand portail ou un petit web, vous pouvez être la cible pour tout quidam avec une connexion. Vous êtes une cible potentielle dès que vous êtes connecté vous-même. Certains programmes de piratage ne font pas dans la demi-mesure, et testent systématiquement des millions d'IP, à la recherche de victimes : ne soyez pas la prochaine.

Etre à jour

PHP, comme de nombreux systèmes de grande taille, est constamment testé et amélioré. Chaque nouvelle version rassemble des modifications majeures et mineures, aussi bien pour renforcer la sécurité, que pour réparer les problèmes de conceptions de configuration, et d'autres points qui peuvent affecter la sécurité globale et la stabilité de votre système.

Comme les autres langages de scripts systèmes, la meilleure approche est de mettre à jour souvent PHP, et de rester à l'écoute des dernières versions et des modifications qu'elles apportent.

Partie II. Référence

Chapitre 5. La syntaxe de base

Le passage du HTML au PHP

Lorsque PHP commence à traiter un fichier, il ne fait qu'afficher le texte HTML qu'il rencontre. Si vous renommez un fichier .html en .php, il fonctionnera exactement comme avant.

Si vous voulez insérer des commandes PHP dans votre fichier, vous devez indiquer à PHP le début d'une telle séquence, en passant en mode PHP. Il y a quatre moyens pour passer du mode HTML au mode PHP :

Exemple 5-1. Le passage du HTML au PHP

```
1.  <? echo ("Ceci est un exemple d'affichage à l'écran en PHP, sous forme d'expres-
sion SGML.\n"); ?>
    <?= expression ?>  Raccourci de "<? echo expression ?>"
2.  <?php echo("Si vous voulez afficher du XML ou du XHTML, faites comme ceci.\n"); ?>
3.  <script language="php">
        echo ("Certains éditeurs HTML (comme FrontPage)
         n'acceptent pas les expressions telles que celle-ci.");
    </script>
4.  <% echo ("Vous pouvez aussi utiliser le style ASP comme délimiteur."); %>
    <%= $variable; # ceci est un raccourci pour  "<%%echo .." %>
```

La deuxième méthode est généralement utilisée, car elle permet une implémentation aisée de PHP avec la prochaine génération de XHTML.

La première possibilité n'est valable que si vous l'avez activée. Soit en faisant appel à la fonction short_tags() (NdT : semble avoir disparu), soit en utilisant l'option d'exécution short_open_tag dans le fichier de configuration, soit en utilisant l'option de compilation --enable-short-tags.

La quatrième possibilité est seulement disponible si vous l'avez activée en utilisant soit l'option d'exécution asp_tags, soit en utilisant l'option de compilation --enable-asp-tags.

> **Note :** Le support de la quatrième possibilité, ASP-style, a été ajouté dans la version 3.0.4.

La marque de fermeture d'un bloc (?>) comprend la nouvelle ligne suivante, s'il y en a une.

PHP vous permet d'utiliser des structures telles que :

Exemple 5-2. Méthode avancée

```
<?php
  if ( boolean-expression ) {
?>
  <strong>Ceci est vrai.</strong>
<?php
  }
  else
  {
?>
<strong>Ceci est faux.</strong>
<?php
  }
?>
```

Cela fonctionne comme on peut s'y attendre, car PHP traite le texte entre ?> et <?php comme une fonction **echo()**, sans remplacer les variables éventuelles par leur valeur.

Le séparateur d’instruction

Les instructions sont séparées comme en C ou en Perl par un point virgule à chaque fin d’instruction.

La balise de fin (?>) implique la fin d’un instruction, et donc ajoute implicitement un point virgule. Les deux exemples suivants sont équivalents.

```
<?php
    echo "Ceci est un test";
?>
<?php echo "Ceci est un test" ?>
```

Commentaires

Le PHP supporte les commentaires comme en C, C++ et Shell Unix. Par exemple:

```
<?php
    echo "Ceci est un test"; // Ceci est un commentaire sur une ligne comme en C++
    /* Ceci est un commentaire sur plusieurs lignes,
       comme en C et C++  */
    echo "Ceci est encore un test";
    echo "Enfin, le test final"; # Ceci est un commentaire comme en Shell Unix
?>
```

Le premier type de commentaire ne commente que jusqu’à la fin de la ligne ou bien jusqu’à la fin du bloc, cela dépend du premier rencontré.

```
<h1>Ceci est un <?php echo "simple";?> exemple.</h1>
<p>
```

La ligne du dessus affichera ’Ceci est un simple exemple’.

Faites attention à ne pas emboîter les commentaires de type ’C’, ce qui arrive de temps en temps lorsque vous voulez commenter une grande partie de code.

```
<?php
 /*
    echo "Ceci est un test"; /* Ce commentaire va poser un problème */
 */
?>
```

Chapitre 6. Les types

Introduction

PHP supporte les 8 types basiques suivants :

PHP supporte quatres types scalaires :

- booléen
- entier
- nombre à virgule flottante
- chaîne de caractères

PHP supporte deux types composés :

- tableau
- objet

PHP supporte deux types spéciaux :

- ressource
- null

> **Note :** Dans ce manuel, vous rencontrerez souvant le type `mixed`. C'est un pseudo-type, qui indique que le paramètre peut indifféremment prendre plusieurs types.

Habituellement, le type d'une variable n'est pas déclaré par le programmeur. Il est décidé au moment de l'exécution par le PHP, en fonction du contexte dans lequel la variable est utilisée.

Si vous voulez forcer une variable à être convertie en un certain type, vous devez transtyper (cast) la variable ou utiliser la fonction **settype()**.

Il est à noter qu'une variable peut se comporter de manière différente suivant les situations, en fonction du type qui lui est affecté. Pour plus d'informations, voir le paragraphe transtypage.

Booléens

C'est le type le plus simple. Un booléen exprime les valeurs de vrai `TRUE` ou `FALSE`.

Vous pouvez utiliser les constantes '`TRUE`' et '`FALSE`' pour spécifier une valeur de type boolean. Généralement, vous les créerez avec un des operateurs qui retourne une valeur boolean, pour le passer à une structure de contrôle.

```
<?php
  if ($action == "show_version"){
// == is an operator
// qui retourne un booléen
    echo "La version est 1.23";
  }
// ceci n'est pas nécessaire
  if ($show_separators == true){
    echo "<hr>\n";
  }
// ceci est plus pratique
  if ($show_separators){
    echo "<hr>\n";
  }
?>
```

Note : Le type booéen a été introduit en PHP 4.

Conversion en booléen

Reportez-vous au chapitre "Définition du type" pour plus d'informations sur les conversions.

Lors des conversions de valeurs de type boolean, les valeurs suivantes sont considérées comme fausse (`FALSE`) :

- Le booléen FALSE
- L'entier 0 (zéro)
- Le nombre à virgule flottante 0.0 (zéro)
- La chaîne vide, et la chaîne "0"
- Le tableau vide (aucun élément)
- L'objet vide (aucun élément)
- La constante spéciale NULL

Toutes les autres valeurs sont considérées comme vraies (`TRUE` (y compris les ressources).

Avertissement

-1 est considéré comme vrai!

Entiers

Un entier est un nombre de l'ensemble des entiers naturels Z : Z = {...., -2, -1, 0, 1, 2, ...}. Il est possible de spécifier les nombres entiers (integers) de toutes les manières suivantes : décimale (base 10), hexadécimale (base 16), octale (base 8) éventuellement précédé du signe moins (-).

Pour utiliser la notation octale, vous devez préfixer le nombre avec un zéro; pour utiliser la notation hexadécimale, vous devez préfixer le nombre avec `0x`.

```
<?php
  $a = 1234; # nombre entier en base 10
  $a = -123; # nombre entier négatif
  $a = 0123; # nombre entier en base 8, octale (équivalent à 83 en base 10)
  $a = 0x12; # nombre entier en base 16, hexadécimale
             # (équivalent à 18 en base 10)
?>
```

La taille des entiers dépend de la plate-forme de support, mais la valeur maximale est généralement de 2 milliards et des poussières (c'est un entier signé de 32 bits). PHP ne supporte pas les entiers non signés.

Note : En PHP, il n'existe pas de type fraction. `1/2` se transforme en nombre à virgule flottante `0.5`.

Dépassement de capacité des entiers

Si un nombre est hors de l'intervalle de validité des entiers, il sera interprété comme un float.

```
<?php
  $large_number =  2147483647;
  var_dump($large_number);
// affiche : int(2147483647)
```

```
  $large_number =  2147483648;
  var_dump($large_number);
// affiche : float(2147483648)
?>
```

De même, si une fonction ou un opérateur retourne un entier qui est hors des limites de validité des entiers, il sera aussi automatiquement converti en float.

```
<?php
  $million = 1000000;
  $large_number =  50000 * $million;
  var_dump($large_number);
// affiche : float(50000000000)
?>
```

Avertissement

Malheureusement, il y a un bug dans le moteur (toujours présent en 4.0.6 et probablement résolu en 4.0.7) ce qui fait que PHP que fonctionne pas toujours bien lorsque des nombres négatifs sont utilisés. Lorsque les deux opérandes sont positifs, il n'y a pas de problèmes. Par exemple : `-50000 * $million`, conduit à `-429496728`.

Conversion en entiers

Pour explicitement convertir une valeur en integer, utilisez les opérateurs de transtypage `(int)` ou `(integer)`. Cependant, dans la plupart des situations, vous n'en aurez pas besoin, car une valeur sera automatiquement convertie si un opérateur, un fonction ou tout autre élément du langage requiert un entier.

Reportez-vous à la section définition de type pour plus d'informations sur les conversions.

Depuis un booléen

FALSE devient 0 (zéro), et TRUE devient 1 (un).

Depuis un nombre à virgule flottante

Lors de conversion entre nombre à virgule flottante et un entier, le nombre sera arrondi à la valeur inférieure s'il est positif, et supérieure s'il est négatif (conversion dite 'vers zéro').

Si le nombre est hors de l'intervalle de validité des entiers, (généralement `+/- 2.15e+9 = 2^31`), le résultat est indéfinit, car les nombres à virgules flottante n'ont pas assez de précision pour fournir une valeur exacte pour un entier.

Avertissement

Aucune alerte, même pas le plus petit message ne sera affiché dans ce cas.

Avertissement

Ne transformez jamais une fraction inconnue en entier, car cela peut conduire à des résultats irrationnels.

```
<?php
  echo (int) ( (0.1+0.7) * 10 );
// affiche 7!
?>
```

Pour plus d'informations, reportez-vous aux alertes liées aux nombres à virgule flottante.

From strings

Reportez-vous à la section des conversions de chaînes.

Conversion d'autres types

La conversion d'autres types en entier est indéfinie. Actuellement, PHP convertit d'abord la valeur en booléen.

> **Attention**
>
> Mais, ne vous fiez pas à ce comportement, car il est susceptible de changer sans préavis!

Voir aussi : Nombres de grande taille et Nombres à virgules flottantes.

Les nombres à virgule flottante

Les nombres à virgule flottante (connus aussi sous le vocable de "`double`" ou "`float`" "`nombre rééls`") peuvent être spécifiés en utilisant la syntaxe suivante:

```
<?php
  $a = 1.234;
  $a = 1.2e3;
?>
```

> **Précision des nombres à virgule flottante**
>
> Il est fréquent que de simples fractions décimales telles que `0.1` ou `0.7` ne puissent être converties au format interne binaire sans une légère perte de précision. Cela peut conduire à des résultats étonnants : par exemple, `floor((0.1+0.7)*10)` retournera `7` au lieu de `8` car le résultat de la représentation interne est `7.9999999999...`.
>
> Tout ceci est lié au fait qu'il est impossible d'exprimer certaines fractions en un nombre fini de chiffres. Par exemple `1/3` s'écrira `0.3333333...` en mode décimal.
>
> Ne faites donc jamais confiance aux nombres à virgule jusqu'à leur dernière décimale, et ne comparez jamais ces nombres avec l'opérateur d'égalité. Si vous avez besoin d'une précision particulière, reportez-vous au traitement des nombres de grande taille avec les librairies BC ou GMP.

Les chaînes de caractères

Les chaînes de caractères sont des séquences de caractères. En PHP, un caractère est un octet, et il y en 256 de possibles. PHP n'a pas (encore?) de support natif d'Unicode.

> **Note :** La taille n'est pas un problème majeur pour une chaîne. Elle peut devenir très grande sans problème. Il n'y a a pas à s'en faire pour cela.

Syntax

Une chaîne peut être spécifiée de trois manières différentes :

- guillemets simples
- guillemets doubles
- syntaxe HereDoc

guillemets simples

Le moyen le plus simple de spécifier une chaîne de caractères est d'utiliser les guillemets simples : `'`.

Pour spécifier un guillemet simple littéral, vous devez l'échapper avec un anti-slash (\), comme dans de nombreux langages. Si un anti-slash doit apparaître dans votre chaîne ou bien en fin de chaîne, il faudra le doubler. Notez que si vous essayez d'échapper n'importe quel autre caractère, l'anti-slash sera conservé! Il n'y a pas besoin d'échapper d'autres caractères que le guillemets lui-même.

> **Note :** En PHP 3, une alerte sera affichée si cela arrive avec un niveau de rapport d'erreur de `E_NOTICE`.

> **Note :** Contrairement aux autres syntaxes, les variables présentes dans la chaîne ne seront *PAS* remplacées par leurs valeurs.

```
<?php
  echo 'Ceci est une chaîne simple';
  echo 'Vous pouvez inclure des nouvelles lignes dans une chaîne,
  comme ceci.';
  echo 'Arnold a coutume de dire : "I\'ll be back"';
// affiche : ... "I'll be back"
  echo 'Etes vous sÛr de vouloir effacer le dossier C:\\*.*?';
// affiche : Etes vous sÛr de vouloir effacer le dossier C:\*.*?
  echo 'Etes vous sÛr de vouloir effacer le dossier C:\*.*?';
// affiche : Etes vous sÛr de vouloir effacer le dossier C:\*.*?
  echo 'Je suis en train de mettre une nouvelle ligne comme ceci : \n';
// affiche : Je suis en train de mettre une nouvelle ligne comme ceci : \n
?>
```

Guillemets doubles

Si la chaîne est entourés de guillemets doubles, PHP va comprendre certaines séquences de caractères :

Tableau 6-1. Les caractères spéciaux

Séquence	Valeur
`\n`	Nouvelle ligne (linefeed, LF ou 0x0A (10) en ASCII)
`\r`	Retour à la ligne(carriage return, CR ou 0x0D (13) en ASCII)
`\t`	Tabulation horizontale (HT ou 0x09 (9) en ASCII)
`\\`	Antislash
`\$`	Caractère $
`\"`	Guillemets doubles
`\[0-7]{1,3}`	Une séquence de caractères qui permet de rechercher un nombre en notation octale.
`\x[0-9A-Fa-f]{1,2}`	Une séquence de caractères qui permet de rechercher un nombre en notation hexadécimale.

Si vous essayez d'utiliser l'anti-slash sur n'importe quelle autre séquence, l'anti-slash sera affiché dans votre chaîne.

Le plus important des chaînes à guillemets doubles est le fait que les variables qui s'y trouvent seront remplacées par leurs valeurs. Voir la section sur le traitement des variables dans les chaînes pour plus de détails.

Syntaxe Heredoc

Un autre moyen de délimiter les chaînes est d'utiliser la syntaxe de "Here doc" (en français, documentation ici): <«, suivi d'un identifiant arbitraire, puis de la chaîne. Cette séquence se termine par l'identifiant initial, placé en premier sur une

nouvelle ligne.

L'identifiant utilisé doit suivre les mêmes règles que les étiquettes PHP : il ne doit contenir uniquement que des caractères alpha-numériques, et des soulignés ("_"), et enfin, commencer par un caractère alphabétique ou un souligné.

> **Avertissement**
>
> Il est très important de noter que la ligne qui contient l'identifiant de fermeture ne doit contenir aucun autre caractère, hormis, éventuellement, un point-virgule `;`. Cela signifie notamment que l'identifiant ne doit pas être indenté, et qu'il n'y a aucun caractère blanc entre le retour à la ligne et l'identifiant, ou bien entre l'identifiant et le `;`.
>
> Le plus dur est peut être qu'il ne faut pas qu'il y ait un retour à la ligne ((`\r`) à la fin de cette ligne, mais seulement un form-feed (`\n`). Puisque Microsoft Windows utilise la séquence `\r\n` comme terminaison de ligne, la syntaxe heredoc risque de ne pas fonctionner là. Cependant, la plupart des éditeurs PHP fournissent une sauvegarde au format UNIX.

La syntaxe Here doc se comporte exactement comme une chaîne à guillemets doubles, sans les guillemets doubles. Cela signifie que vous n'avez pas à échapper les guillemets (simples ou doubles) dans cette syntaxe. Les variables sont remplacées par leur valeur, et le même soin doit leur être apporté que dans les chaînes à guillemets doubles.

Exemple 6-1. Exemple de chaîne HereDoc

```
<?php
  $str = <«EOD
  Exemple de chaîne
  s'étalant sur
  plusieurs lignes
  avec la syntaxe heredoc
EOD;
/* Exemple plus complexe, avec des variables. */
  class foo {
    var $foo;
    var $bar;
    function foo() {
        $this->foo = 'Foo';
        $this->bar = array('Bar1', 'Bar2', 'Bar3');
    }
  }
  $foo = new foo();
  $name = 'MonNom';
  echo <«EOT
  Mon nom est "$name". J'affiche des $foo->foo.
  Maintenant, j'affiche un {$foo->bar[1]}.
  Ceci se traduit par un 'A' majuscule: \x41
EOT;
?>
```

Note : Le support Here doc a été ajouté en PHP 4.

Traitement des variables dans les chaînes

Lorsqu'une chaîne est spécifiée avec des guillemets doubles, ou en utilisant la syntaxe heredoc, les variables qu'elle contient sont remplacées par leur valeur.

Il y a deux types de syntaxe, une simple et une complexe. La syntaxe simple est la plus courante, et la plus pratique : elle fournit un moyen d'utiliser les variables, que ce soit des chaînes, des tableaux ou des membres d'objets.

La syntaxe complexe a été introduite en PHP 4 et peut être reconnue grâce aux accolades entourant les expressions.

Syntaxe simple

Dès qu'un signe dollar $ est rencontré, l'analyseur PHP va lire autant de caractère qu'il peut pour former un nom de variable valide. Entourez le nom de la variable avec des accolades pour indiquer explicitement son nom.

```
<?php
  $boisson = 'vin';
  echo "Du $boisson, du pain et du fromage!';
 // OK, car "," n'est pas autorisé dans les noms de variables
  echo "Il a goÛté plusieurs $vins";
 // Pas OK, car 's' peut entrer dans un nom de variable, et PHP recherche $boissons
  echo "Il a goÛté plusieurs ${vin}s";
 // OK
?>
```

Similairement, vous pouvez aussi utiliser un élément de tableau, ou un membre d'objet. Pour les éléments de tableau, le crochet fermant ']' marquera la fin du nom de la variable. Pour les membres d'objets, les mêmes règles que ci-dessus s'appliquent. Cependant, il n'existe pas de truc comme pour les variables simples.

```
 $fruits = array( 'fraise' => 'rouge' , 'banane' => 'jaune' );
 echo "Une banane est $fruits[banane].";
 // OK
 echo "Ce carré est large de $carre->largeur mètres.";
 // OK
 echo "Ce carré est large de $carre->largeur00 mètres..";
 // pas OK
 // pour résoudre ce problème, voyez syntaxe complexe.
```

Pour tout ce qui sera plus compliqué, voyez la syntaxe complexe.

Syntaxe complexe

La syntaxe est dite "complexe" car elle permet l'utilisation d' expressions complexes, et non pas parcequ'elle serait obscure. Nuance.

En fait, vous pouvez inclure n'importe quelle valeur qui est dans l'espace de nom avec cette syntaxe. Il suffit d'écrire une expression exactement comme si elle était hors de la chaîne, puis de l'entourer d'accolades {}. Puisque vous ne pouvez pas échapper les accolades, cette syntaxe ne commence qu'à partir du signe dollar, qui suit immédiatement l'accolade ouvrante. Par exemple, vous pouvez utiliser "{\$" pour obtenir un "{$" littéral. Voici quelques exemples :

```
<?php
  $super = 'fantastique';
  echo "C'est { $super}";
// ne fonctionne pas,
// affiche "C'est { fantastique}"
  echo "C'est {$super}";
// fonctionne,
// affiche "C'est fantastique"
  echo "Ce carré a {$square->width}00 centimètres de large.";
  echo "Ceci fonctionne : {$tableau[4][3]}";
  echo "Ceci échoue : {$tableau[foo][3]}";
// pour la même raison que $tableau[bar] ne fonctionne pas hors d'une chaîne

  echo "Essayez plutôt comme ceci : {$tableau['foo'][3]}";
  echo "Vous pouvez même écrire {$objet->valeurs[3]->nom}";
  echo "Voici la valeur de la variable nommée $name: {${$name}}";

// cela fonctionne, mais c'est vivement déconseillé.
// Et pour finir, on peut combiner avec des fonctions
  $inv = 'Bordeaux';
  echo "Je reprendrai bien un verre de cet excellent {${ strrev('niv') }}, hips";

?>
```

Accès aux caractères d'une chaîne

Les caractères d'une chaîne sont accessibles en spécifiant leur offset (le premier caractère est d'offset 0) entre accolade, après le nom de la variable.

> **Note :** Pour assurer la compatibilité ascendante, vous pouvez toujours accéder aux caractères avec des crochets. Mais cette syntaxe est obsolète en PHP 4.

Exemple 6-2. Exemples de chaînes

```
<?php
/* Affectation d'une chaîne. */
  $str = "Ceci est une chaîne";
/* Concaténation. */
  $str = $str . " avec un peu plus de texte";
/* Une autre méthode de concaténation. */
  $str .= " et une nouvelle ligne à la fin.\n";
/*  Cette chaîne finira comme : '<p>Nombre: 9</p>' */
  $num = 9;
  $str = "<p>Nombre: $num</p>";
/* Celle-ci sera '<p>Nombre: $num</p>' */
  $num = 9;
  $str = '<p>Nombre: $num</p>';
/* Premier caractère d'une chaîne */
  $str = 'Ceci est un test.';
  $first = $str{0};
/* Dernier caractère d'une chaîne. */
  $str = 'This is still a test.';
  $last = $str{strlen($str)-1};
?>
```

Fonctions et opérateurs pratiques

Les chaînes peuvent être concaténées avec l'opérateur '.' (point). Notez que l'opérateur d'addition '+' (plus) ne fonctionnera pas. Reportez-vous à la section opérateurs de chaînes.

Il y a une grande quantité de fonctions pratiques pour modifier les chaînes.

Reportez-vous à la section chaînes de caractères pour les fonctions les plus générales, à Expressions régulières Perl et Expressions régulières POSIX extendues pour les recherches et remplacements.

Il y a aussi les fonctions sur les URL, ainsi que des fonctions de chiffrage (mcrypt et mhash).

Finalement, si vous ne trouvez toujours pas votre bonheure, il y a les fonctions de types de caractères.

Conversion de type

Lorsqu'une chaîne de caractère est évaluée comme une valeur numérique, le résultat et le type de la variable sont déterminés comme suit.

La chaîne de caractères est de type "double" si elle contient un des caractère '.', 'e' ou 'E'. Sinon, elle est de type entier ("integer").

La valeur est définie par la première partie de la chaîne. Si la chaîne de caractères débute par une valeur numérique cette valeur sera celle utilisée. Sinon, la valeur sera égale à 0 (zéro).

Lorsque la première expression est une chaîne de caractères, le type de la variable dépend de la seconde expression.

```
<?php
  $foo = 1 + "10.5";              // $foo est du type  float (11.5)
```

```
  $foo = 1 + "-1.3e3";            // $foo est du type  float (-1299)
  $foo = 1 + "bob-1.3e3";         // $foo est du type  integer (1)
  $foo = 1 + "bob3";              // $foo est du type  integer (1)
  $foo = 1 + "10 Small Pigs";     // $foo est du type  integer (11)
  $foo = 1 + "10 Little Piggies"; // $foo est du type  integer (11)
  $foo = "10.0 pigs " + 1;        // $foo est du type  integer (11)
  $foo = "10.0 pigs " + 1.0;      // $foo est du type  float (11)
?>
```

Pour plus d'informations sur les conversions de type, voir les pages de man à propos de la fonction strtod(3).

Si vous voulez testez l'un des exemples de cette section, vous pouvez faire un copier/coller et l'insérer dans un script pour voir comment il se comporte.

```
<?php
  echo "\$foo==$foo; type is " . gettype( $foo ) . "<br>\n";
?>
```

Les tableaux

Un tableau PHP est en fait une association ordonnée (map). Une association est un type qui fait correspondre des valeurs à des *clés*. Ce type est optimisé de diverses façons, qui font que vous pouvez le manipuler comme un tableau à indices réels, une liste (vecteur), ou un table de hachage (qui est une implémentation d'association), dictionnaire, collection, pile, queue et encore d'autres. Comme une valeur peut elle-même être un tableau, vous pouvez simuler facilement un arbre.

Les détails d'implémentation de ces structures sont hors du champs de ce manuel, mais vous trouverez ici un exemple de toutes ces structures.

Syntaxe

Créer un tableau array()

Un tableau array peut être créé avec la fonction **array()**. Cette fonction prend en argument des structures `key => value`, séparées par des virgules.

Une clé `key` est soit un entier positif ou bien une chaîne. Si une clé est la représentation standard d'un entier positif, elle sera interprété comme tel. (i.e. `'8'` sera interprété comme `8`, tandis que `'08'` sera interprété comme `'08'`).

Une valeur peut être n'importe quoi.

Omettre des clés

Si vous omettez une clé lors de la spécification d'un tableau, l'indice maximum + 1 sera utilisé comme clé par défaut. Si aucun indice numérique n'a été généré, ce sera 0. Si vous spécifiez une qui a déjà été assigné, la nouvelle valeur écrasera la précédente.

```
array( [key =>]
value
     , ...
     )
// key is either string
ou entier integer positif
// value peut être n'importe quoi
```

La syntaxe à crochets

Vous pouvez aussi modifier un tableau existant en lui assignant simplement des valeurs.

L'assignement de valeurs de tableau se fait en spécifiant la clé entre crochets. Si vous omettez la clé ("`$tableau[]`"), la valeur sera ajoutée à la fin du tableau.

```
$arr[key] = value;
$arr[] = value;
// key est soit une chaîne, soit un entier
// value peut être n'importe quoi
```

Si `$arr` n'existe pas, il sera créé. Cela en fait une alternative pour créer un tableau. Pour modifier une valeur, assignez lui une nouvelle valeur. Pour supprimer une valeur, utilisez la fonction **unset()**.

Fonctions pratiques

Il y a toute une panoplie de fonctions pratiques pour travailler avec les tableau : array-functions.

L'élément de langage foreach est spécifiquement dédiée aux tableau : elle permet de passer en revue simplement les valeurs d'un tableau.

Exemples

Le type tableau de PHP est très souple. Voici quelques exemples d'utilisation :

```
<?php
// ceci
  $a = array( 'couleur' => 'rouge'
            , 'gout' => 'sucre'
            , 'forme' => 'rond'
            , 'nom'  => 'pomme'
            ,            4        // cette clé sera 0
          );
// est complétement équivalent à
  $a['couleur'] = 'rouge';
  $a['gout'] =    'sucre';
  $a['forme'] =   'rond';
  $a['nom'] =     'pomme';
  $a[]        = 4;        // cette clé sera 0
  $b[] = 'a';
  $b[] = 'b';
  $b[] = 'c';
// va créer le tableau array( 0 => 'a' , 1 => 'b' , 2 => 'c' )
// ou plus simplement array('a' , 'b' , 'c' )
?<
```

Exemple 6-3. Utilisation de array()

```
<?php
// Array comme correspondance
  $map = array( 'version'    => 4
              , 'OS'         => 'Linux'
              , 'langue'       => 'français'
              , 'short_tags' => TRUE             );
// valeur strictement numériques
  $array = array( 7
                , 8
                , 0
                , 156
                , -10
```

```
                );
// ceci est la même chose que array( 0 => 7, 1 => 8, ...)
  $switching = array(         10 // clé = 0
                    , 5    =>  6
                    , 3    =>  7
                    , 'a'  =>  4
                    ,          11 // clé = 6 (index maximum : 5)
                    , '8'  =>  2 // clé = 8 (entier!)
                    , '02' => 77 // clé = '02'
                    , 0    => 12 // la valeur de la clé 10 sera remplacée par 12
                    );

// empty array
$empty = array();
?<
```

Exemple 6-4. Collection

```
<?php
  $couleurs = array('rouge','bleu','vert','jaune');
  foreach ( $couleurs as $couleur ){
    echo "Aimez-vous la couleur $couleur?\n";
  }
/* Affiche :
Aimez-vous la couleur rouge?
Aimez-vous la couleur bleu?
Aimez-vous la couleur vert?
Aimez-vous la couleur jaune?
*/
?>
```

Notez qu'il n'est pas possible actuellement de modifier les valeurs d'un tableau avec une telle boucle. Une solution pour cela est :

Exemple 6-5. Collection

```
<?php
  foreach( $couleurs as $cle => $couleur ){
// ne marche pas
//$couleur = strtoupper($couleur);
//marche :
    $couleurs[$cle] = strtoupper($couleur);
  }
  print_r($couleur);
/* Affiche :
Array
(
    [0] => ROUGE
    [1] => BLEU
    [2] => VERT
    [3] => JAUNE
)
*/
?>
```

Cet exemple crée un tableau d'index minimal 1.

Exemple 6-6. Tableau en 1

```
<?php
  $firstquarter  = array(1 => 'Janvier', 'Février', 'Mars');
  print_r($firstquarter);
/* Affiche:
Array
(
    [1] => 'Janvier'
    [2] => 'Février'
    [3] => 'Mars'
)
*/
?>
```

Exemple 6-7. Remplissage d'un tableau

```
<?php
// remplis un tableau avec les noms de fichiers d'un dossier
  $handle = opendir('.');
  while ( $file = readdir($handle) ){
    $files[] = $file;
  }
  closedir($handle);
?>
```

Les tableaux sont ordonnés. Vous pouvez modifier l'ordre des valeurs avec de nombreuses fonctions de classement. Voyez les fonctions de tableaux.

Exemple 6-8. Tri de tableaux

```
<?php
  sort($files);
  print_r($files);
?>
```

Comme une valeur de tableau peut être n'importe quoi, elle peut aussi être un autre tableau. Comme cela, vous pouvez avoir des tableaux multi-dimensionnels, ou récursifs.

Exemple 6-9. Tableaux multi-dimensionnels, et récursifs

```
<?php
  $fruits = array ( "fruits"  => array ( "a" => "orange"
                                       , "b" => "banane"
                                       , "c" => "pomme"
                                       )
                  , "nombres" => array ( 1
                                       , 2
                                       , 3
                                       , 4
                                       , 5
                                       , 6
                                       )
                  , "trous"   => array ( "premier"
                                       , 5 => "second"
                                       , "troisième"
                                       )
                  );

?>
```

Attention aux tableaux

Pourquoi est ce que `$foo[bar]` est invalide?

Dans vos vieux scripts, vous pouvez avoir utilisé la syntaxe suivante :

```
<?php
  $foo[bar] = 'ennemi';
  echo $foo[bar];
?>
```

Cela est mauvais, mais ca marche. Pourquoi est-ce mauvais? La raison est que PHP réclame une expression entre les crochets (comme indiqué dans la section sur la syntaxe des tableaux). Cela signifie que vous pouvez écrire quelque chose comme :

```
<?php
  echo $arr[ foo(true) ];
?>
```

Ceci est un exemple d'utilisation de retour de fonction dans un index de tableau. PHP reconnait aussi les constantes, et vous pouvez avoir déjà rencontré `E_*`.

```
<?php
  $error_descriptions[E_ERROR] = "Une erreur fatale est survenue";
  $error_descriptions[E_WARNING] = "PHP a généré une alerte";
  $error_descriptions[E_NOTICE] = "Ceci est juste une note grâcieuse";
?>
```

Notez que `E_ERROR` est aussi un identifiant vlaide, tout comme `bar` dans le premier exemple. Mais le dernier exemple revient à écrire :

```
<?php
  $error_descriptions[1] = "Une erreur fatale est survenue";
  $error_descriptions[2] = "PHP a généré une alerte";
  $error_descriptions[8] = "Ceci est juste une note grâcieuse";
?>
```

car `E_ERROR` égale `1`, etc.

Alors, comment se fait-tile que `$foo[bar]` fonctionne? C'est parce que `bar` est attendu comme une constante. Cependant, dans ce cas, aucune constante n'a pour nom `bar`. PHP suppose alors que vous voulez dire `bar` littéralement, c'est-à-dire la chaîne `"bar"`, mais que vous avez oublié les guillements.

Alors, pourquoi est-ce mal?

Dans le futur, l'équipe de développement peut vouloir ajouter une nouvelle constante et vous vous retrouverez dans la panade. Par exemple, vous ne pouvez déjà pas utiliser les constantes `empty` et `default`, car ce sont des mots reservés.

Et, pour en mettre une autre couche, cette syntaxe est tout simplement obsolète, et risque de ne plus fonctionner du tout un jour ou l'autre.

> **Note :** Lorsque vous activez le rapport d'erreur avec **error_reporting()** avec `E_ALL`, PHP générera des notes à chaque fois que cette syntaxe est utilisée. Essayez donc `error_reporting(E_ALL);` dans vos scripts, pour voir).

> **Note :** Dans une chaîne à guillements doubles, une autre syntaxe est valide. Voyez la section sur Traitement des variables dans les chaînes pour plus de détails.

Les objets

Initialisation d'un objet

Pour initialiser un objet, vous devez utiliser la commande "new" afin de créer linstance de l'objet.

```
<?php
  class foo {
    function faire_foo () {
        echo "Faisant foo.";
    }
  }
  $bar = new foo;
  $bar->faire_foo();
?>
```

Ressources

Une ressource (resource en anglais), est un type spécial, qui représente une référence sur une ressource externe. Les ressources sont créées par des fonctions dédiées. Reportez vous à l'annexe ressource pour une liste exhaustive des fonctions créant et utilisant ces ressources.

Note : Le type ressource a été introduit en PHP 4.

Libérer des ressources

Grâce au système de comptabilisation des références introduit en PHP 4 (avec le moteur Zend), PHP détecte automatiquement qu'une ressource n'est plus utilisée (comme Java). Dans ce cas, toutes les ressources systèmes utilisées par cette ressource sont libérées automatiquement.

Note : Les connexions persistantes représentent un cas particulier, elles ne seront *PAS* détruites. Voyez connexions persistantes.

La valeur NULL

La valeur spéciale `NULL` représente l'absence de valeur. Une variable avec la valeur NULL n'a pas de valeur.

Syntaxe

Il y a seulement une valeur de type NULL, et c'est la constante `NULL`, insensible à la casse.

```
<?php
  $var = Null;
?>
```

Note : La valeur NULL a été introduite en PHP 4.

Définition du type

PHP ne nécessite pas de déclaration explicite du type d'une variable. Le type d'une variable est déterminé par le contexte d'utilisation. Par exemple, si vous assignez une chaîne de caractères à la variable *var*, var devient une chaîne de caractère. Si vous assignez un nombre entier à *var*, elle devient un entier.

Un exemple de convertisseur automatique de type est l'opérateur '+'. Si un des opérandes est de type double, alors tous les opérandes sont évalués comme des variables de type double et le résultat est de type double. Sinon, tous les opérandes sont évalués comme des variables de type entier et le résultat sera du type entier. Il est à noter que cela NE CHANGE PAS le type des opérandes. Le seul changement est la manière dont les opérandes sont évaluées.

```
<?php
  $foo = "0";  // $foo est une chaîne de caractères (ASCII 48)
  $foo += 2;   // $foo est maintenant du type entier (2)
  $foo = $foo + 1.3;  // $foo est maintenant du type double (3.3)
  $foo = 5 + "10 Petits cochons"; // $foo est du type entier (15)
  $foo = 5 + "10 cochonnets";     // $foo est du type entier (15)
?>
```

Si les deux derniers exemples vous semblent obscurs ou si vous voulez forcer une variable a être évaluée avec un certain type, reportez-vous au paragraphe Conversion de type ci-dessous. Si vous voulez changer le type d'une variable, intéressez-vous à aux fonctions de conversion de chaînes.

Si vous voulez forcer le type d'une variable, vous pouvez vous reporter à la section transtypage. Si vous voulez changer le type d'une variable, utilisez **settype()**.

Pour tester les exemples de cette section, il suffit d'en faire un copier/coller, et d'insérer les lignes dans un script PHP.

```
<?php
  echo "\$foo==$foo; le type est " . gettype( $foo ) . "<br>\n";
?>
```

Note : Le comportement de la conversion automatique est actuellement indéfinie.

```
<?php
  $a = 1;       // $a est un entier
  $a[0] = "f";  // $a devient un tableau, et $a[0] contient "f"
?>
```

Bien que dans l'exemple ci-dessus, il semble évident que $a va devenir un tableau, dont le premier élément est 'f', considérez l'exemple suivant :

```
<?php
  $a = "1";     // $a est une chaîne
  $a[0] = "f";  // Qu'est ce qu'une position dans une chaîne ? que se passe t il?
?>
```

Etant donné que PHP supporte l'indexation de chaîne avec des offsets identiques à celles des tableaux, l'exemple ci-dessus conduit à un problème : est ce que $a est un tableau, dont le premier élément est "f", ou bien est ce que "f" est le premier élément de la chaîne de caractères $a?

Pour cette raison, aussi bien pour PHP 3.0.12 que PHP 4.0b3-RC4, le résultat de la conversion automatique est considéré comme indéfinie. Des solutions sont en cours de discussion.

Transtypage

La conversion de type en PHP fonctionne de la même manière qu'en C: le nom du type désiré est écrit entre parenthèses devant la variable à transtyper ("cast").

```
<?php
  $foo = 10;   // $foo est un entier
  $bar = (double) $foo;   // $bar est un double
?>
```

Les conversions autorisées sont:

- (int), (integer) - type entier
- (bool), (boolean) - booléen
- (real), (double), (float) - type double
- (string) - ctype chaîne
- (array) - type tableau
- (object) - type objet

Il est à noter que les tabulations et les espaces sont autorisés à l'intérieur des parenthèses, donc les lignes suivantes sont équivalentes:

```
<?php
  $foo = (int) $bar;
  $foo = ( int ) $bar;
?>
```

Le transtypage n'a pas toujours de résultat prévisible. Pour plus d'informations, voyez :

- Conversion en booléen
- Conversion en entier

Avertissement

Pour transformer facilement une variable en chaîne, entourez la simplement de guillements doubles.

Lors de la conversion d'un tableau en chaîne, le résultat sera le mot `Array` (tableau, en anglais). Lors de la conversion d'un objet en chaîne, le résultat sera le mot `Object` (objet, en anglais). Dans les deux cas, une alerte sera affichée.

Lorsque vous transtypez un scalaire ou une chaîne en tableau, la variable verra son contenu affecté au premier élément du tableau.

```
<?php
  $var = 'ciao';
  $arr = (array) $var;
  echo $arr[0];  // affiche 'ciao'
?>
```

Lorsque vous transtypez un scalaire ou une chaîne en objet, la valeur de la variable sera transformée en attribut de l'objet. L'attribut s'appellera 'scalar':

```
<?php
  $var = 'ciao';
  $obj = (object) $var;
```

```
  echo $obj->scalar;  // affiche 'ciao'
?>
```

Chapitre 7. Les variables

Essentiel

En PHP, les variables sont représentées par un signe dollar "$" suivi du nom de la variable. Le nom est sensible à la casse (ie : $x != $X).

Les noms de variables suivent les mêmes règles de nommage que les autres entités PHP. Un nom de variable valide doit commencer par une lettre ou un souligné (_), suivi de lettres, chiffres ou soulignés. Exprimé sous la forme d'une expression régulière, cela donne : '[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*'

> **Note :** Dans nos propos, une lettre peut être une des lettres minuscules (a à z) ou majuscules (A à Z), et les caractères ASCII de 127 à 255 (0x7f-0xff).

```
<?php
$var = "Jean";
$Var = "Paul";
echo "$var, $Var";         // affiche "Jean, Paul"
$4site = 'pas encore';     // invalide : commence par un nombre
$_4site = 'pas encore';    // valide : commence par un souligné
$maïs = 'jaune';      // valide; 'ï' est ASCII 239.
?>
```

En PHP 3, les variables sont toujours assignées par valeur. C'est-à-dire, lorsque vous assignez une expression à une variable, la valeur de l'expression est recopiée dans la variable. Cela signifie, par exemple, qu'après avoir assigné la valeur d'une variable à une autre, modifier l'une des variables n'aura pas d'effet sur l'autre. Pour plus de détails sur ce genre d'assignation, reportez-vous à Expressions.

PHP 4 permet aussi d'assigner les valeurs aux variables *par référence*. Cela signifie que la nouvelle variable ne fait que référencer (en d'autres terme, "devient un alias de", ou encore "pointe sur") la variable originale. Les modifications de la nouvelle variable affecteront l'ancienne, et vice versa. Cela signifie aussi qu'aucune copie n'est faite : l'assignation est donc beaucoup plus rapide. Cela se fera notamment sentir dans des boucles, ou lors d'assignation de grands objets (tableaux).

Pour assigner par référence, ajoutez simplement un & (ET commercial) au début de la variable qui est assignée (la variable source). Dans l'exemple suivant, "`Mon nom est Pierre`" s'affichera deux fois :

```
<?php
$foo = 'Pierre';              // Assigne la valeur 'Pierre' à $foo
$bar = &$foo;              // Référence $foo avec $bar.
$bar = "Mon nom est Pierre";  // Modifie $bar...
echo $foo;                 // $foo est aussi modifiée
echo $bar;
?>
```

Une chose importante à noter est que seules les variables nommées peuvent être assignées par référence.

```
<?php
$foo = 25;
$bar = &$foo;      // assignation valide .
$bar = &(24 * 7);  // assignation invalide : référence une expression sans nom
function test() {
   return 25;
}
$bar = &test();    // assignation invalide.
?>
```

Variables prédéfinies

PHP fourni un grand nombre de variables prédéfinies. Cependant, beaucoup de ces variables ne peuvent pas être présentées ici, car elles dépendent du serveur sur lequel elles tournent, de la version du serveur, et de la configuration du serveur, ou encore d'autres facteurs. Certaines de ces variables ne seront pas accessibles lorsque PHP fonctionne en exécutable.

Malgré ces données, voici une liste de variables prédéfinies, qui seront accessibles avec une installation ad hoc de PHP3, fonctionnant en module, sous Apache (http://www.apache.org/) 1.3.6.

Pour la liste complète des variables prédéfinies (et d'autres informations pratiques) reportez-vous (et usez) de **phpinfo()**.

> **Note :** Cette liste n'est pas exhaustive et ne le sera pas. C'est simplement un aperçu des variables prédéfinies qui peuvent être accessibles dans les scripts.

Variables Apache

Ces variables sont créées par le serveur Apache (http://www.apache.org/). Si vous utilisez un autre serveur web, il n'est pas sur que celui-ci vous fournira les mêmes variables. Il peut ne pas les fournir, en fournir d'autres. Cependant, un bon nombre de ces variables font partie de l'interface CGI 1.1 (http://hoohoo.ncsa.uiuc.edu/cgi/env.html), et on peut s'attendre à les retrouver.

Notez que peu d'entre elles seront accessibles lorsque PHP est appelé en ligne de commande, (et elles n'auront alors peut être pas de sens)

$GATEWAY_INTERFACE

Numéro de révision de l'interface CGI du serveur : i.e. 'CGI/1.1'.

$SERVER_NAME

Le nom du serveur hôte qui éxécute le script suivant. Si le script est exécuté sur un hôte virtuel, ce sera la valeur définie pour cet hôte virtuel.

$SERVER_SOFTWARE

Chaîne d'identification du serveur, qui est donnée dans les en-têtes lors de la réponse aux requêtes.

$SERVER_PROTOCOL

Nom et révision du protocole de communication : i.e. 'HTTP/1.0';

$REQUEST_METHOD

Méthode de requête utilisée pour accéder à la page; i.e. 'GET', 'HEAD', 'POST', 'PUT'.

$QUERY_STRING

La chaîne de requête, si elle existe, qui est utilisée pour accéder à la page.

$DOCUMENT_ROOT

La racine sous laquelle le script courant est exécuté, comme défini dans la configuration du serveur.

$HTTP_ACCEPT

Contenu de l'en-tête `Accept:` de la requête courante, s'il y en a une.

$HTTP_ACCEPT_CHARSET

Contenu de l'en-tête `Accept-Charset:` de la requête courante, s'elle existe. Par exemple : '`iso-8859-1,*,utf-8`'.

$HTTP_ACCEPT_ENCODING

Contenu de l'en-tête `Accept-Encoding:` de la requête courante, si elle existe. Par exemple : 'gzip'.

$HTTP_ACCEPT_LANGUAGE

Contenu de l'en-tête `Accept-Language:` de la requête courante, si elle existe. Par exemple : 'en'.

$HTTP_CONNECTION

Contenu de l'en-tête `Connection:` de la requête courante, si elle existe. Par exemple : 'Keep-Alive'.

$HTTP_HOST

Contenu de l'en-tête `Host:` de la requête courante, si elle existe.

$HTTP_REFERER

L'adresse de la page (si elle existe) qui a conduit le client à la page courante. Cette valeur est affectée par le client, et tous les clients ne le font pas.

$HTTP_USER_AGENT

Contenu de l'en-tête `User_Agent:` de la requête courante, si elle existe. C'est une chaîne qui décrit le client HTML utilisé pour voir la page courante. Par exemple : `Mozilla/4.5 [en] (X11; U; Linux 2.2.9 i586)`. Entre autres choses, vous pouvez utiliser cette valeur avec **get_browser()** pour optimiser votre page en fonction des capacités du client.

$REMOTE_ADDR

L'adresse IP du client qui demande la page courante.

$REMOTE_PORT

Le port utilisé par la machine cliente pour communiquer avec le serveur web.

$SCRIPT_FILENAME

Le chemin absolu jusqu'au script courant.

$SERVER_ADMIN

La valeur donnée à la directive SERVER_ADMIN (pour Apache), dans le fichier de configuration. Si le script est exécuté par un hôte virtuel, ce sera la valeur définie par l'hôte virtuel.

$SERVER_PORT

Le port de la machine serveur utilisé pour les communications. Par défaut, c'est '80'. En utilisant SSL, par exemple, il sera remplacé par le numéro de port HTTP sécurisé.

$SERVER_SIGNATURE

Chaîne contenant le numéro de version du serveur et le nom d'hôte virtuel, qui sont ajoutés aux pages générées par le serveur, si cette option est activée.

$PATH_TRANSLATED

Chemin dans le système de fichier (pas le document root-) jusqu'au script courant, une fois que le serveur a fait une chemin traduction virtuel->réel.

$SCRIPT_NAME

Contient le nom du script courant. Cela sert lorsque les pages doivent s'appeler elles-mêmes.

$REQUEST_URI

L'URI qui a été fourni pour accéder à cette page. Par exemple : '/index.html'.

Variables d'environnement

Ces variables sont importées dans l'espace de nom global de PHP, depuis l'environnement sous lequel PHP fonctionne. Beaucoup d'entre elles sont fournies par le shell qui exécute PHP et différents systèmes étant susceptibles de disposer de différents shells, une liste définitive est impossible à établir. Reportez-vous à la documentation de votre shell, pour connaître la liste des variables d'environnement prédéfinies.

Les autres variables d'environment incluent les variables CGI, placées ici, quelquefois la méthode d'éxécution de PHP (CGI ou module).

Variables PHP

Ces variables sont créées par PHP lui_même. Les variables `$HTTP_*_VARS` ne sont disponibles que si l'option de configuration track_vars a été activée. Lorsque c'est le cas, ces variables existent toujours, même si ce sont des tableaux vides. Cela évite les usurpations mal intentionnées de ces variables.

> **Note :** Depuis PHP 4.0.3, track_vars est toujours activé, quelle que soit la configuration.

Si la directive register_globals est activée, alors ces variables seront aussi disponibles comme variables globales du script : c'est-à-dire, indépendamment des tableaux `$HTTP_*_VARS`. Cette fonctionnalité doit être utilisée avec précautions, et de préférence, désactivée. Si `$HTTP_*_VARS` est sécurisé, les équivalents globaux peuvent être écrasés par les données d'entrée de l'utilisateur, avec des intrusions possibles. Si vous ne pouvez pas désactiver register_globals, vous devez prendre toutes les dispositions possibles pour vous assurer que les données utilisées sont sÛres.

$argv

Tableau des arguments passées au script. Lorsque le script est appelé en ligne de commande, cela donne accès aux arguments, comme en langage C. Lorsque le script est appelé avec la méthode GET, ce tableau contiendra la chaîne de requête.

$argc

Contient le nombre de paramètres de la ligne de commande passés au script (si le script fonctionne en ligne de commande).

$PHP_SELF

Le nom du fichier du script en cour d'éxécution, par rapport au document root. Si PHP fonctionne en ligne de commande, cette variable n'est pas disponible.

$HTTP_COOKIE_VARS

Un tableau associatif des variables passées au script courant via les HTTP cookies. Uniquement possible si le suivi des variables a été activé avec la directive générale track_vars ou avec la directive locale `<? php_track_vars ?>`.

$HTTP_GET_VARS

Un tableau associatif des variables passées au script courant via les HTTP GET. Uniquement possible si le suivi des variables a été activé avec la directive générale track_vars ou avec la directive locale `<? php_track_vars ?>`.

$HTTP_POST_VARS

Un tableau associatif des variables passées au script courant via les HTTP POST. Uniquement possible si le suivi des variables a été activé avec la directive générale track_vars ou avec la directive locale `<? php_track_vars ?>`.

$HTTP_POST_FILES

Un tableau associatif contenant les informations sur les fichiers téléchargés avec la méthode HTTP POST. Reportez-vous au chapitre Téléchargement par méthode POST pour plus de détails sur le contenu de `$HTTP_POST_FILES`.

`$HTTP_POST_FILES` n'est disponible que dans les versions 4.0.0 et plus récentes de PHP.

$HTTP_ENV_VARS

Un tableau associatif des variables passées au script par l'environnement parent.

$HTTP_SERVER_VARS

Un tableau associatif des variables passées au script par le serveur HTTP. Ces variables sont analogues aux variables décrites ci-dessus.

Portée des variables

La portée d'une variable dépend du contexte dans lequel la variable est définie. Pour la majorité des variables, la portée concerne la totalité d'un script PHP. Mais, lorsque vous définissez une fonction, la portée d'une variable définie dans cette fonction est locale à la fonction. Par exemple:

```
<?php
$a = 1;
include "b.inc";
?>
```

Ici, la variable $a sera accessible dans le script inclus `b.inc`. Cependant, dans les fonctions définies par l'utilisateur, une nouvelle définition de cette variable sera donnée, limitée à la fonction. Toute variable utilisée dans une fonction est par définition, locale. Par exemple :

```
<?php
$a = 1; /* portée globale */
function test() {
    echo $a; /* portée locale */
}
test();
?>
```

Le script n'affichera rien à l'écran car la fonction **echo()** utilise la variable locale $a, et celle-ci n'a pas été assignée préalablement dans la fonction. Vous pouvez noter que ce concept diffère un petit peu du langage C dans lequel une variable globale est automatiquement accessible dans les fonctions, à moins d'être redéfinie localement dans la fonction. Cela peut poser des problèmes si vous redéfinissez des variables globales localement. En PHP, une variable globale doit être déclarée à l'intérieur de chaque fonction afin de pouvoir être utilisée dans cette fonction. Par exemple:

```
<?php
$a = 1;
$b = 2;
function somme() {
    global $a, $b;
    $b = $a + $b;
}
somme();
echo $b;
```

Le script ci-dessus va afficher la valeur "3". En déclarant globales les variables $a et $b locales de la fonction somme(), toutes les références à ces variables concerneront les variables globales. Il n'y a aucune limite au nombre de variables globales qui peuvent être manipulées par une fonction.

Une deuxième méthode pour accéder aux variables globales est d'utiliser le tableau associatif prédéfini $GLOBALS. Le précédent exemple peut être réécrit de la manière suivante:

```
<?php
$a = 1;
$b = 2;
function somme() {
    $GLOBALS["b"] = $GLOBALS["a"] + $GLOBALS["b"];
}
somme();
echo $b;
?>
```

Le tableau $GLOBALS est un tableau associatif avec le nom des variables globales comme clef et les valeurs des éléments du tableau comme valeur des variables.

Une autre caractéristique importante de la portée des variables est la notion de variable *static*. Une variable statique a une portée locale uniquement, mais elle ne perd pas sa valeur lorsque le script appelle la fonction. Prenons l'exemple suivant:

```
<?php
function test() {
    $a = 0;
```

```
    echo $a;
    $a++;
}
?>
```

Cette fonction est un peu inutile car à chaque fois qu'elle est appelée, elle initialise $a à 0 et affiche "0". L'incrémentation de la variable ($a++) ne sert pas à grand chose, car dès que la fonction est terminée la variable disparaît. Pour faire une fonction de comptage utile, c'est-à-dire qui ne perdra pas la trace du compteur, la variable $a est déclarée comme une variable statique:

```
<?php
function test() {
    static $a = 0;
    echo $a;
    $a++;
}
?>
```

Maintenant, à chaque fois que la fonction Test() est appelée, elle affichera une valeur de $a incrémentée de 1.

Les variables statiques sont essentielles lorsque vous faites des appels récursifs à une fonction. Une fonction récursive est une fonction qui s'appelle elle-même. Il faut faire attention lorsque vous écrivez une fonction récursive car il est facile de faire une boucle infinie. Vous devez vérifier que vous avez bien une condition qui permet de terminer votre récursivité. La fonction suivante compte récursivement jusqu'à 10:

```
<?php
function test() {
    static $count = 0;
    $count++;
    echo $count;
    if ($count < 10) {
        test();
    }
    $count--;
}
?>
```

Les variables dynamiques

Il est pratique d'avoir parfois des noms de variables qui sont variables. C'est-à-dire un nom de variable qui est affectée et utilisée dynamiquement. Une variable classique est affecté avec l'instruction suivante:

```
<?php
$a = "bonjour";
?>
```

Une variable dynamique prend la valeur d'une variable et l'utilise comme nom d'une autre variable. Dans l'exemple ci-dessous, *bonjour* peut être utilisé comme le nom d'une variable en utilisant le "$$" précédent la variable. C'est-à-dire

```
<?php
$$a = "monde";
?>
```

A ce niveau, deux variables ont été définies et stockées dans l'arbre des symboles PHP: $a avec comme valeur "bonjour" et $bonjour avec comme valeur "monde". Alors, l'instruction

```
<?php
echo "$a ${$a}";
?>
```

produira le même affichage que :

```
<?php
echo "$a $bonjour";
?>
```

c'est-à-dire : *bonjour monde*.

Afin de pouvoir utiliser les variables dynamiques avec les tableaux, vous avez à résoudre un problème ambigu. Si vous écrivez $$a[1], le parseur a besoin de savoir si vous parler de la variable qui a pour nom $a[1] ou bien si vous voulez l'index [1] de la variable $$a. La syntaxe pour résoudre cette ambiguïté est la suivante: ${$a[1]} pour le premier cas, et ${$a}[1] pour le deuxième.

Variables externes à PHP

Formulaires HTML (GET et POST)

Lorsqu'un formulaire est envoyé à un script PHP, toutes les variables du formulaire seront automatiquement disponibles dans le script. Par exemple, considérons le formulaire suivant:

Exemple 7-1. Exemple avec un formulaire simple

```
<form action="foo.php3" method="post">
    Name: <input type="text" name="name"><br>
    <input type="submit">
</form>
```

Lorsque ce formulaire est envoyé, le PHP va créer la variable `$name`, qui contiendra la valeur que vous avez entrée dans le champs *Name:* du formulaire.

Le PHP permet aussi l'utilisation des tableaux dans le contexte de formulaire, mais seulement des tableaux à une seule dimension. Comme cela, vous pouvez rassembler des variables ou utiliser cette fonctionnalité pour récupérer les valeurs d'un choix multiple :

Exemple 7-2. Variables complexes de formulaire

```
<form action="array.php" method="post">
    Name: <input type="text" name="personal[name]"><br>
    Email: <input type="text" name="personal[email]"><br>
    Beer: <br>
    <select multiple name="vin[]">
        <option value="medoc">Médoc
        <option value="chablis">Chablis
        <option value="riesling">Riesling
        </select>
    <input type="submit">
</form>
```

Si l'option "track_vars" est activée, soit par l'option de compilation track_vars, soit par la directive de configuration `<? php_track_vars ?>`, les variables transmises par les méthodes POST et GET pourront aussi être trouvées dans le tableau associatif global $HTTP_POST_VARS ou $HTTP_GET_VARS suivant la méthode utlisée.

Bouton "submit" sous forme d’image

Lorsque vous envoyez le résultat d’un formulaire, vous pouvez utiliser une image au lieu du bouton "submit" standard en utilisant un tag :

```
<input type=image src="image.gif" name="sub">
```

Lorsqu’un utilisateur clique sur l’image, le formulaire sera transmis au serveur avec deux variables de plus, sub_x et sub_y. Ces deux variables contiennent les coordonnées de l’endroit où l’utilisateur à cliqué. Les utilisateurs expérimentés remarqueront que les noms de variables sont transmis avec une virgule à la place du caractère "_", mais le PHP fait la conversion automatiquement.

HTTP Cookies

Le PHP supporte les cookies HTTP de manière totalement transparente, comme défini dans les Netscape’s Spec (http://www.netscape.com/newsref/std/cookie_spec.html). Les cookies sont un mécanisme permettant de stocker des données sur la machine cliente à des fins d’authentification de l’utilisateur. Vous pouvez établir un cookie grâce à la fonction **setcookie()**. Les cookies font partie intégrante du "header" HTTP, et donc la fonction **setcookie()** doit être appelée avant que le moindre affichage ne soit envoyé au navigateur. C’est la même restriction que pour la fonction **header()**. Tout cookie envoyé depuis le client sur le serveur sera automatiquement stocké sous forme de variable, comme pour la méthode POST ou GET.

Si vous souhaitez assigner plusieurs valeurs à un seul cookie, il vous faut ajouter les caractères *[]* au nom de votre cookie. Par exemple :

```
<?php
setcookie ("MonCookie[]", "test", time()+3600);
?>
```

Il est à noter qu’un cookie remplace le cookie précédent par un cookie de même nom tant que le "path" ou le domaine sont identiques. Donc, pour une application de caddie, vous devez implémenter un compteur et l’incrémenter au fur et à mesure. C’est-à-dire:

Exemple 7-3. Exemple avec setcookie()

```
<?php
$compte++;
SetCookie ("Compte", $compte, time()+3600);
SetCookie ("Caddie[$compte]", $item, time()+3600);
?>
```

Variables d’environnement

Le PHP fait en sorte que les variables d’environnement soient accessibles directement comme des variables PHP normales.

```
<?php
echo $HOME;  /* Affiche la valeur de la variable d’environnement HOME,
                si celle-ci est affectée. */
?>
```

Même si le PHP crée les variables lors de l’utilisation des méthodes GET, POST et cookie, il est de temps en temps préférable de transmettre explicitement la valeur de la variable afin d’être sÛr de la valeur. La fonction **getenv()** peut être utilisée pour récupérer la valeur des variables d’environnement. Vous pouvez aussi affecter une variable d’environnement grâce à la fonction **putenv()**.

Cas des points dans les noms de variables

Typiquement, PHP ne modifie pas les noms des variables lorsqu'elles sont passées à un script. Cependant, il faut noter que les points (.) ne sont pas autorisés dans les noms de variables PHP. Pour cette raison, jetez un oeil sur :

```
<?php
$varname.ext;  /* nom de variable invalide */
?>
```

Dans ce cas, l'analyseur croit voir la variable nommée $varname, suivie par l'opérateur de concaténation, et suivi encore par la chaîne non-guillemetée (une chaîne sans guillemets, et qui n'a pas de signification particulière). Visiblement, ce n'est pas ce qu'on attendait...

Pour cette raison, il est important de noter que PHP remplacera automatiquement les points des noms de variables entrantes par des soulignés (underscore).

Détermination du type des variables

Parceque le PHP détermine le type des variables et les convertit (généralement) comme il faut, ce n'est pas toujours le type de variable que vous souhaitez. PHP inclut des fonctions permettant de déterminer le type d'une variable : **gettype()**, **is_long()**, **is_double()**, **is_string()**, **is_array()** et **is_object()**.

Chapitre 8. Les constantes

Une constante est un identifiant (un nom) qui représente une valeur simple. Comme leur nom le suggère, cette valeur ne peut jamais être modifiée durant l'exécution du script (les constantes magiques `__FILE__` et `__LINE__` sont les seules exception). Le nom d'une constante est sensible à la casse, apr défaut. Par convention, les constantes sont toujours en majuscules.

Les noms de constantes suivent les mêmes règles que n'importe quel nom en PHP. Un nom de constante valide commence par une lettre ou un souligné (_), suivi d'un nombre quelconque de lettre, chiffres ou soulignés. Sous forme d'expression régulière, cela peut s'exprimer comme ceci : `[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*`

> **Note :** Dans cette documentation, une lettre peut être un des caractères suivants : de a à z, de A à Z et tous les caractères ASCII de 127 à 255 (0x7f-0xff).

Les constantes sont accessibles de manière globale.

Syntaxe

Vous pouvez définir une constante en utilisant la fonction **define()**. Une fois qu'une constante est définie, elle ne peut jamais être modifiée, ou détruite.

Seuls les types de données scalaires peuvent être placés dans une constante : c'est à dire les types booléen, entier, double et chaîne de caractères (soit boolean, integer, double et string).

Vous povuez accéder à la valeur d'une constante en spécifiant simplement son nom. Contrairement aux variables, vous ne devez *PAS* préfixer le nom de la constante avec $. Vous pouvez aussi utiliser la fonction **constant()**, pour lire dynamiquement la valeur d'une constante, si vous obtenez le nom de cette constante dynamiquement (retour de fonction, par exemple). Utilisez la fonction **get_defined_constants()** pour connaître la liste de toutes les fonctions définies.

> **Note :** Les constantes et les variables globales utilisent deux espaces de noms différents. Ce qui implique que `TRUE` et `$TRUE` sont généralement différents (en tous cas, ils peuvent avoir des valeurs différentes).

Lorsque vous utilisez une constante non définie, PHP suppose que vous utilisez le nom de la constante. Une note sera générée. Utilisez la fonction **defined()** pour savoir si une constante existe ou pas.

Il y a des différences entre les constantes et les variables :

- Les constantes ne commencent pas par le signe ($);
- Les constantes sont définies et accessibles à tout endroit du code, globalement.
- Les constantes ne peuvent pas être redéfinies ou indéfinies une fois qu'elles ont été définies.
- Les constantes ne peuvent contenir que des scalaires.

Exemple 8-1. Definir une constante

```
<?php
  define("CONSTANTE", "Bonjour le monde.");
  echo CONSTANTE; // affiche "Bonjour le monde."
  echo Constante; // affiche "Constante" et une note.
?>
```

Constantes prédéfinies

Les constantes prédéfinies sont toujours disponibles. En voici la liste :

__FILE__ (insensible à la casse)

Le nom du fichier qui est actuellement exécuté. Si cette constante est utilisée dans le cadre d'un fichier "inclus" (après utilisation de **require()**), alors le nom du fichier inclus est renvoyée, et non le nom du fichier parent.

__LINE__ (insensible à la casse)

Le numéro de la ligne qui est actuellement exécutée. Si cette constante est utilisée dans le cadre d'un fichier "inclus" (après utilisation de **require()**), c'est la position dans le fichier inclus qui est renvoyé.

PHP_VERSION

La chaîne de caractères de présentation de la version du PHP qui est actuellement utilisée. Par exemple '4.0.0'.

PHP_OS

Nom du système d'exploitation qui est utilisé par la machine qui fait tourner le PHP. Parmi les valeurs possibles : "AIX", "Darwin" (MacOS), "Linux", "SunOS", "WIN32", "WINNT". Note : cette liste n'est pas exhaustive.

TRUE

La valeur vraie booléenne, TRUE.

FALSE

La valeur faux booléenne, FALSE.

E_ERROR

Dénote une erreur autre qu'une erreur d'analyse ("parse error") qu'il n'est pas possible de corriger.

E_WARNING

Dénote un contexte dans lequel le PHP trouve que quelque chose ne va pas. Mais l'exécution se poursuit tout de même. Ces alertes-là peuvent être récupérées par le script lui-même. Un exemple serait une expression régulière (regexp) invalide dans la fonction **ereg()**.

E_PARSE

L'analyseur a rencontré une forme syntaxique invalide dans le script. Correction de l'erreur impossible.

E_NOTICE

Quelque chose s'est produit, qui peut être ou non une erreur. L'exécution continue. Par exemple, le cas de guillemets doubles (") non refermés, ou bien la tentative d'accéder à une variable qui n'est pas encore affectée.

E_ALL

Toutes les constantes E_* rassemblées en une seule. Si vous l'utilisez avec **error_reporting()**, toutes les erreurs et les problèmes que PHP rencontrera seront notifiés.

Les constantes E_* sont généralement utilisées avec la fonction **error_reporting()**.

Vous pouvez définir d'autres constantes en utilisant la fonction **define()**.

Il est à noter que ce sont des constantes, et non pas des macros comme en C. Seulement les données scalaires peuvent être représentées par des constantes.

Exemple 8-2. Définition de constantes

```
<?php
define("CONSTANTE", "Bonjour le monde.");
echo CONSTANTE;
// affiche "Bonjour le monde."
?>
```

Exemple 8-3. Utilisation des constantes __FILE__ et __LINE__

```
<?php
function rapport_erreur($file, $line, $message) {
    echo "Une erreur est survenue dans le fichier $file à la ligne $line: $message.";
}
rapport_erreur(__FILE__,__LINE__, "Y a un problème!");
```

```
?>
```

Chapitre 9. Les expressions

Les expressions sont la partie la plus importante du PHP. En PHP, presque tout ce que vous écrivez est une expression. La manière la plus simple de définir une expression est : "tout ce qui a une valeur".

Les formes les plus simples d'expressions sont les constantes et les variables. Lorsque vous écrivez "$a = 5", vous assignez la valeur '5' à la variable $a. Bien évidemment, '5' vaut 5 ou, en d'autres termes, '5' est une expresssion avec pour valeur 5 (dans ce cas, '5' est un intier constant).

Après cette assignation, vous pouvez considérer que $a a pour valeur 5 et donc, écrire $b = $a, revient à écrire $b = 5. En d'autres termes, $a est une expression avec de valeur 5. Si tout fonctionne correctement, c'est exactement ce qui arrive.

Un exemple plus complexe concerne les fonctions. Par exemple, considérons la fonction suivante :

```
<?php
function foo () {
    return 5;
}
?>
```

Considérant que vous êtes familier avec le concept de fonction, (si ce n'est pas le cas, jetez un oeil au chapitre concernant les fonctions), vous serez d'accord que $c = foo() est équivalent à $c = 5, et vous aurez tout à fait raison. Les fonctions sont des expressions qui ont la valeur de leur "valeur de retour". Si foo() renvoie 5, la valeur de l'expression 'foo()' est 5. Habituellement, les fonctions ne font pas que renvoyer une valeur constante mais réalisent des traitements.

Bien sur, les valeurs en PHP n'ont pas à être des valeurs numériques, comme c'est souvent le cas. PHP supporte 3 types de variables scalaires : les valeurs entières, les nombres à virgule flottante et les chaînes de caractères. (une variable scalaire est une variable que vous ne pouvez pas scinder en morceau, au contraire des tableaux par exemple). PHP supporte aussi deux types composés : les tableaux et les objets. Chacun de ces types de variables peuvent être affectés ou renvoyés par une fonction.

Les utilisateurs de PHP/FI 2 ne verront aucun changement. Malgré tout, PHP va plus loin dans la gestion des expressions, comme le font d'autres langages. PHP est un langage orienté expression, dans le sens oú presque tout est une expression. Considérons l'exemple dont nous avons déjà parlé, '$a = 5'. Il est facile de voir que il y a deux valeurs qui entrent en jeux ici, la valeur numérique constante '5' et la valeur de la variable $a qui est mis à jour à la valeur 5. Mais, la vérité est qu'il y a une autre valeur qui entre en jeu ici et c'est la valeur de l'assignement elle-même. L'assignement lui-même est assigné à une valeur, dans ce cas-là 5. En pratique, cela signifie que '$a = 5' est une expression qui a pour valeur 5. Donc, en écrire '$b = ($a = 5)' revient à écrire '$a = 5; $b = 5;' (un point virgule marque la fin d'une instruction). Comme les assignements sont analysés de droite à gauche, vous pouvez aussi bien écrire '$b = $a = 5'.

Un autre bon exemple du langage orienté expression est la pré-incrémentation et la post-incrémentation, (ainsi que la décrémentation). Les utilisateurs de PHP/FI 2 et ceux de nombreux autres langages sont habitués à la notation "variable++" et "variable--". Ce sont les opérateurs d'incrémentation et de décrémentation. En PHP/FI 2, l'instruction '$a++' n'a aucune valeur (c'est-à-dire que ce n'est pas une expression) et vous ne pouvez donc pas l'utiliser. PHP ajoute les possibilités d'incrémentation et de décrémentation comme c'est le cas dans le langage C. En PHP, comme en C, il y a deux types d'opérateurs d'incrémentation (pré-incrémentation et post-incrémentation). Les deux types d'opérateur d'incrémentation jouent le même rôle (c'est-à-dire qu'il incrémente la variable). La différence vient de la valeur de l'opérateur d'incrémentation. L'opérateur de pré-incrémentation, qui s'écrit '++$variable', évalue la valeur incrémentée (PHP incrémente la variable avant de lire la valeur de cette variable, d'oú le nom de 'pré-incrémentation'). L'opérateur de post-incrémentation, qui s'écrit '$variable++', évalue la valeur de la variable avant de l'incrémenter. (PHP incrémente la variable après avoir lu sa valeur, d'oú le nom de 'post-incrémentation').

Un type d'expression très commun est l'expression de comparaison. Ces expressions sont évaluées à 0 ou 1, autrement dit `FALSE` ou `TRUE` (respectivement). PHP supporte les opérateurs de comparaison > (plus grand que), => (plus grand ou égal), == (égal à), < (plus petit que), <= (plus petit ou égal). Ces expressions sont utilisées de manière courante dans les instructions conditionnelles, comme l'instruction `if`.

Pour le dernier exemple d'expression, nous allons parler des combinaisons d'opérateurs/assignement. Vous savez que si vous voulez incrémenter la variable $a d'une unité, vous devez simplement écrire '$a++'. Mais si vous voulez ajouter la valeur '3' à votre variable ? Vous pouvez écrire plusieurs fois '$a++', mais ce n'est pas la meilleure des méthodes. Un pratique plus courante est d'écrire '$a = $a + 3'. L'expression '$a + 3' correspond à la valeur $a plus 3, et est de nouveau assignée à la variable $a. Donc le résultat est l'incrémentation de 3 unités. En PHP, comme dans de nombreux autres langages comme le C, vous pouvez écrire cela de manière plus concise, manière qui avec le temps se révélera plus claire et plus rapide à comprendre. Ajouter 3 à la valeur de la variable $a peut s'écrire '$a += 3'. Cela signifie précisement : "on prend la valeur de la variable $a, on ajoute la valeur 3 et on assigne cette valeur à la variable $a". Et pour être plus concis et

plus clair, cette expression est plus rapide. La valeur de l'expression '$a += 3', comme l'assignement d'une valeur quelconque, est la valeur assignée. Il est à noter que ce n'est pas 3 mais la combinaison de la valeur de la variable $a plus la valeur 3. (c'est la valeur qui est assignée à la variable $a). N'importe quel opérateur binaire peu utiliser ce type d'assignement, par exemple '$a -= 5' (soustraction de 5 de la valeur de la variable $a), '$b *= 7' (multiplication de la valeur de la variable $b par 7).

Il y a une autre expression qui peut paraître complexe si vous ne l'avez pas vu dans d'autre langage, l'opérateur conditionnel ternaire:

```
<?php
$first ? $second : $third
?>
```

Si la valeur de la première sous-expression est vraie, (différente de 0), alors la deuxième sous-expression est évaluée et constitue le résultat de l'expression conditonnelle. Sinon, c'est la troisème sous-expression qui est évaluée et qui constitue le résultat de l'expression.

Les exemples suivants devraient vous permettre de mieux comprendre la pré- et post- incrémentation et le concept des expressions en général:

```
<?php
function double($i) {
    return $i*2;
}
$b = $a = 5;        /* assigne la valeur 5 aux variables $a et $b  */
$c = $a++;          /* post-incrémentation de la variable $a et assignation de
                       la valeur à la variable $c */
$e = $d = ++$b;     /* Pré-incrémentation, et assignation de la valeur aux
                       variables $d et $e  */
/* à ce niveau, les variables $d et $e sont égales à 6 */
$f = double($d++);  /* assignation du double de la valeur de $d à la va-
riable $f ($f vaut 12),
                       puis incrémentation de la valeur de $d  */
$g = double(++$e);  /* assigne deux fois la valeur de $e après
                       incrémentation, 2*7 = 14 to $g */
$h = $g += 10;      /* Tout d'abord, $g est incrémentée de 10, et donc $g vaut 24.
                       Ensuite, la valeur de $g, (24) est assignée à la variable $h,
                       qui vaut donc elle aussi 24. */
?>
```

Au début de ce chapitre, nous avons dit que nous allions décrire les différents types d'instructions, et donc, comme promis, nous allons voir que les expressions peuvent être des instructions. Mais, attention, toutes les expressions ne sont pas des instructions. Dans ce cas-là, une instruction est de la forme 'expr' ';', c'est-à-dire, une expression suivie par un point-virgule. L'expression '$b = $a = 5;', '$a = 5' est valide, mais ce n'est pas une instruction en elle-même. '$b = $a = 5' est une instruction valide.

La dernière chose qui mérite d'être mentionnée est la véritable valeur des expressions. Lorsque vous faîtes des tests sur une variable, dans une boucle conditionnelle par exemple, cela ne vous intéresse pas de savoir qu'elle est la valeur exacte de l'expression. Mais vous voulez seulement savoir si le résultat signifie `TRUE` ou `FALSE` (PHP n'a pas de type booléen). La véritable valeur d'une expression en PHP est calculée de la même manière qu'en Perl. Toute valeur numérique différente de 0 est considérée comme étant `TRUE`. Une chaîne de caractères vide et la chaîne de caractère 0 sont considérées comme `FALSE`. Toutes les autres valeurs sont vraies. Avec les types de variables non-scalaires (les tableaux et les objets), s'ils ne contiennent aucun élément, renvoient `FALSE`, sinon, ils renvoient `TRUE`.

PHP propose une implémentation complète et détaillée des expressions. PHP documente toutes ses expressions dans le manuel que vous êtes en train de lire. Les exemples qui vont suivre devraient vous donner une bonne idée de ce qu'est une expression et comment construire vos propres expressions. Dans tout ce qui va suivre, nous écrirons *`expr`* pour indiquer toute expression PHP valide.

Chapitre 10. Les opérateurs

Les opérateurs arithmétiques

Vous rappelez-vous des opérations élémentaires apprises à l'école ?

Tableau 10-1. Opérations élémentaires

Exemple	Nom	Résultat
$a + $b	Addition	Somme de $a et $b.
$a - $b	Soustraction	Différence de $a et $b.
$a * $b	Multiplication	Produit de $a et $b.
$a / $b	Division	Quotient de $a et $b.
$a % $b	Modulo	Reste de $a divisé par $b.

L'opérateur de division ("/") retourne une valeur entière (le résultat d'une division entière) si les deux opérandes sont entiers (ou bien des chaînes converties en entier. Si l'un des opérandes est un nombre à virgule flottante, ou bien le résultat d'une opération qui retourne une valeur non entière, un nombre à virgule flottante sera retourné.

Les opérateurs d'assignation

L'opérateur d'assignation le plus simple est le signe "=". Le premier réflexe est de penser que ce signe veut dire "égal à". Ce n'est pas le cas. Il signifie que l'opérande de gauche se voit affecter la valeur de l'expression qui est à droite du signe égal.

La valeur d'une expression d'assignation est la valeur assignée. Par exemple, la valeur de l'expression '$a = 3' est la valeur 3. Cela permet d'utiliser des astuces telles que :

```
<?php
  $a = ($b = 4) + 5;
// $a est maintenant égal à 9, et $b vaut 4.
?>
```

En plus du simple opérateur d'assignation, il existe des "opérateurs combinés" pour tous les opérateurs arithmétiques et pour les opérateurs sur les chaînes de caractères. Cela permet d'utiliser la valeur d'une variable dans une expression et d'affecter le résultat de cette expression à cette variable. Par exemple:

```
<?php
  $a = 3;
  $a += 5;
// affecte la valeur 8 à la variable $a.
// correspond à l'instruction '$a = $a + 5');
  $b = "Bonjour ";
  $b .= " tout le monde!";
// affecte la valeur "Bonjour tout le monde!" à
//  la variable $b
//  identique à $b = $b." tout le monde!";
?>
```

On peut noter que l'assignation copie le contenu de la variable originale dans la nouvelle variable (assignation par valeur), ce qui fait que les changements de valeur d'une variable ne modifieront pas la valeur de l'autre. Cela peut se revéler important lors de la copie d'un grand tableau durant une boucle. PHP 4 supporte aussi l'assignation par référence, en utilisant la syntaxe `$var = &$othervar;`, mais ce n'était pas possible en PHP 3. 'L'assignation par référence' signifie que les deux variables contiennent les mêmes données, et que la modification de l'une affecte l'autre. D'un autre côté, la recopie est très rapide.

Opérateurs sur les bits

Les opérateurs sur les bits vous permettent de manipuler les bits dans un entier.

Tableau 10-2. Les opérateurs sur les bits

Exemple	Nom	Résultat
$a & $b	ET (AND)	Les bits positionnés à 1 dans $a ET dans $b sont positionnés à 1.
$a | $b	OU (OR)	Les bits positionnés à 1 dans $a OU $b sont sont positionnés à 1.
$a ^ $b	Xor	Les bits positionnés à 1 dans $a OU dans $b sont positionnés à 1.
~ $a	NON (Not)	Les bits qui sont positionnés à 1 dans $a sont positionnés à 0, et vice versa.
$a << $b	Décalage à gauche	Décale les bits de $a dans $b par la gauche (chaque décalage équivaut à une multiplication par 2).
$a >> $b	Décalage à droite	Décalage des bits de $a dans $b par la droite (chaque décalage équivaut à une division par 2).

Opérateurs de comparaison

Les opérateurs de comparaison, comme leur nom l'indique, vous permettent de comparer deux valeurs.

Tableau 10-3. Opérateurs de comparaison

Exemple	Nom	Résultat
$a == $b	Egal	Vrai si $a est égal à $b.
$a === $b	Identique	Vrai si $a est égal à $b et qu'ils sont de même type (PHP 4 seulement).
$a != $b	Différent	Vrai si $a est différent de $b.
$a <> $b	Différent	Vrai si $a est différent de $b.
$a < $b	Plus petit que	Vrai si $a est plus petit strictement que $b.
$a > $b	Plus grand	Vrai si $a est plus grand strictement que $b.
$a <= $b	Inférieur ou égal	Vrai si $a est plus petit ou égal à $b.
$a >= $b	Supérieur ou égal	Vrai si $a est plus grand ou égal à $b.

Un autre opérateur conditionnel est l'opérateur ternaire (":?"), qui fonctionne comme en langage C.

```
<?php
  (expr1) ? (expr2) : (expr3);
?>
```

Cette expression renvoie la valeur de l'expression *`expr2`* si l'expression *`expr1`* est vraie, et l'expression *`expr3`* si l'expression *`expr1`* est fausse.

Opérateur de contrôle d'erreur

PHP supporte un opérateur de contrôle d'erreur : c'est @. Lorsque cet opérateur est ajouté en préfixe d'une expression PHP, les messages d'erreur qui peuvent être générés par cette expression seront ignorés.

Si l'option track_errors est activée, les messages d'erreurs générés par une expression seront sauvés dans la variable globale $php_errormsg. Cette variable sera écrasée à chaque erreur. Il faut alors la surveiller souvent pour pouvoir l'utiliser.

```
<?php
/* Erreur intentionnelle (le fichier n'existe pas): */
  $my_file = @file ('non_persistent_file') or
    die ("Impossible d'ouvrir le fichier : L'erreur est : '$php_errormsg'");
// Cela fonctionne avec n'importe quelle expression, pas seulement les fonctions
  $value = @$cache[$key];
// la ligne ci-dessus n'affichera pas d'alerte si la clé $key du tableau n'existe pas
?>
```

Note : L'opérateur @ ne fonctionne qu'avec les expressions. La règle générale de fonctionnement est la suivante : si vous pouvez prendre la valeur de quelquechose, vous pouvez le préfixer avec @. Par exemple, vous pouvez ajouter @ aux variables, fonctions, à **include()**, aux constantes, etc... Vous ne pourrez pas le faire avec des éléments de langage tels que les classes, `if` et `foreach`, etc...

Note : La plupart des fonctions d'accès aux bases de données ne retournent pas d'erreur PHP. Il faut y accéder avec une fonction du type base_de_donnees_get_error().

Voir aussi **error_reporting()**.

Opérateur d'exécutions

PHP supporte un opérateur d'exécution : guillemets obliques ("“"). Notez bien la différence entre les guillemets simples (sur la touche 4), et ceux-ci (sur la touche de la livre anglaise). PHP essaiera d'exécuter le contenu de ces guillemets obliques comme une commande shell. Le résultat sera retourné (i.e. : il ne sera pas simplement envoyé à la sortie standard, il peut être assigné à une variable).

```
<?php
  $output = `ls -al`;
  echo "<pre>$output</pre>";
?>
```

Note : Cet opérateur est désactivé lorsque le safe mode est activé.

Voir aussi **system()**, **passthru()**, **exec()**, **popen()** et **escapeshellcmd()**.

Opérateurs d'incrémentation/Décrémentation

PHP supporte les opérateurs de pré et post incrémentation et décrémentation, comme en C.

Tableau 10-4. Opérateurs d'incrémentation/Décrémentation

Exemple	Nom	Résultat
++$a	Pré-incrémente	Incrémente $a de 1, puis retourne $a.
$a++	Post-incrémente	Retourne $a, puis l'incrémente de 1.
--$a	Pré-décrémente	Décrémente $a de 1, puis retourne $a.
$a--	Post-décrémente	Retourne $a, puis décrémente $a de 1.

Voici un exempla simple

```
<?php
  echo "<h3>Post-incrémentation</h3>";
  $a = 5;
  echo "Devrait valoir  5: " . $a++ . "<br>\n";
  echo "Devrait valoir  6: " . $a . "<br>\n";
  echo "<h3>Pré-incrémentation</h3>";
  $a = 5;
  echo "Devrait valoir  6: " . ++$a . "<br>\n";
  echo "Devrait valoir  6: " . $a . "<br>\n";
  echo "<h3>Post-décrémentation</h3>";
  $a = 5;
  echo "Devrait valoir  5: " . $a-- . "<br>\n";
  echo "Devrait valoir  4: " . $a . "<br>\n";
  echo "<h3>Pré-décrémentation</h3>";
  $a = 5;
  echo "Devrait valoir  4: " . --$a . "<br>\n";
  echo "Devrait valoir  4: " . $a . "<br>\n";
?>
```

Les opérateurs logiques

Tableau 10-5. Les opérateurs logiques

Exemple	Nom	Résultat
$a and $b	ET (And)	Vrai si $a ET $b sont vrais.
$a or $b	OU (Or)	Vrai si $a OU $b est vrai
$a xor $b	XOR (Xor)	Vrai si $a OU $b est vrai, mais pas les deux en même temps.
! $a	NON (Not)	Vrai si $a est faux.
$a && $b	ET (And)	Vrai si $a ET $b sont vrais.
$a || $b	OU (Or)	Vrai si $a OU $b est vrai.

La raison pour laquelle il existe deux types de "ET" et de "OU" est qu'ils ont des priorités différentes. Voir le paragraphe précédence d'opérateurs.

La précédence des opérateurs

La priorité des opérateurs spécifie l'ordre dans lequel les valeurs doivent être analysées. Par exemple, dans l'expression 1 + 5 * 3, le résultat est 16 et non 18, car la multiplication ("*") à une priorité supérieure par rapport à à l'addition ("+").

Le tableau suivant dresse une liste de la priorité des différents opérateurs dans un ordre croissant de priorité.

Tableau 10-6. Précédence des opérateurs

Associativité	Opérateurs
gauche	,
gauche	or
gauche	xor
gauche	and
droite	print
gauche	= += -= *= /= .= %= &= |= ^= ~= <<=>>=
gauche	? :
gauche	||

Associativité	Opérateurs
gauche	&&
gauche	|
gauche	^
gauche	&
non-associative	== != ===
non-associative	< <= > >=
gauche	<< >>
gauche	+ - .
gauche	* / %
droite	! ~ ++ -- (int) (double) (string) (array) (object) @
droite	[
non-associative	new

Opérateurs de chaînes

Il y a deux opérateurs de chaînes. Le premier est l'opérateur de concaténation ('.'), qui retourne la concaténation de ses deux arguments. Le second est l'opérateur d'assignation concaténant ('.='). Reportez-vous à Opérateurs d'assignations pour plus de détails.

```
<?php
  $a = "Bonjour ";
  $b = $a . "Monde!";
// $b contient "Bonjour Monde!"
  $a = "Bonjour ";
  $a = $a . "Monde!";
// $a contient "Bonjour Monde!"
?>
```

Chapitre 11. Les structures de contrôle

Tous les scripts PHP sont une suite d'instructions. Une instruction peut être une assignation, un appel de fonction, une instruction conditionnelle ou bien une instruction qui ne fait rien (une instruction vide). Une instruction se termine habituellement par un point virgule (";"). De plus, plusieurs instructions peuvent être regroupées en bloc, délimité par des accolades ("{}"). Un bloc est considéré comme une instruction. Les différents types d'instruction sont décrits dans ce chapitre.

if

L'instruction `if` est une des plus importantes instructions de tous les langages, PHP inclus. Elle permet l'exécution conditionnelle d'une partie de code. Les fonctionnalités de l'instruction `if` sont les mêmes en PHP qu'en C :

```
<?php
  if (expression)
    commandes
?>
```

Comme nous l'avons vu dans le paragraphe consacré aux expressions, *`expr`* est évaluée à sa vraie valeur. Si l'expression *`expr`* est `TRUE`, PHP exécutera l'instruction et si elle est `FALSE`, l'instruction sera ignorée.

L'exemple suivant affiche la phrase `a est plus grand que b` si *`$a`* est plus grand que *`$b`*:

```
<?php
  if ($a > $b)
    print "a est plus grand que b";
?>
```

Souvent, vous voulez que plusieurs instructions soient exécutées après un branchement conditionnel. Bien évidemment, il n'est pas obligatoire de répéter l'instruction conditionnelle autant de fois que vous avez d'instructions à exécuter. A la place, vous pouvez rassembler toutes les instructions dans un bloc. L'exemple suivant affiche `a est plus grand que b`, et assigne la valeur de la variable *`$a`* à la variable *`$b`*:

```
<?php
if ($a > $b) {
    print "a est plus grand que b";
    $b = $a;
}
?>
```

Vous pouvez imbriquer indéfiniment des instructions `if` les unes dans les autres, ce qui permet une grande flexibilité dans l'exécution d'une partie de code suivant un grand nombre de conditions.

else

Souvent, vous voulez exécuter une instruction si une condition est remplie, et une autre instruction si cette condition n'est pas remplie. C'est à cela que sert `else`. `else` fonctionne après un `if` et exécute les instructions correspondantes au cas oú l'expression du `if` est `FALSE`. Dans l'exemple suivant, ce bout de code affiche `a est plus grand que b` si la variable *`$a`* est plus grande que la variable *`$a`*, et `a est plus petit que b` sinon:

```
<?php
if ($a > $b) {
    print "a est plus grand que b";
} else {
    print "a est plus petit que b";
}
?>
```

Les instructions après le `else` ne sont exécutées que si l'expression du `if` est `FALSE`, et si elle n'est pas suivi par l'expression `elseif`.

`elseif`

`elseif`, comme son nom l'indique, est une combinaison de `if` et de `else`. Comme l'expression `else`, il permet d'exécuter une instruction aprè un `if` dans le cas oú le "premier" if est évalué comme `FALSE`. Mais, à la différence de l'expression `else`, il n'exécutera l'instruction que si l'expression condionnelle `elseif` est évaluée comme `TRUE`. L'exemple suivant affichera `a est plus grand que b`, `a est égal à b` ou `a est plus petit que b`:

```
<?php
if ($a > $b) {
    print "a est plus grand que b";
} elseif ($a == $b) {
    print "a est égal à b";
} else {
    print "a est plus petit que b";
}
?>
```

Vous pouvez avoir plusieurs `elseif` qui s'imbriquent les uns dans les autres, après un `if` initial. Le premier `elseif` qui sera évalué à `TRUE` sera exécuté. En PHP, vous pouvez aussi écrire "else if" en deux mots et son comportement sera identique à la version en un seul mot.

L'expression `elseif` est exécutée seulement si le `if` précédent et tout autre `elseif` précédent est évalués comme `FALSE`, et que votre `elseif` est évalué à `TRUE`.

Syntaxe alternative

> **Avertissement**
>
> Cette syntaxe alternative est obsolète depuis PHP 4. Elle génère un code qui est tout simplement illisible, et il est très difficile de la combiner avec la syntaxe normale. Bien que cela ne soit pas à l'ordre du jour aujourd'hui, cette syntaxe risque de disrparaitre à terme. Soyez prévenus.

Le PHP propose une autre manière de rassembler des instructions à l'intérieur d'un bloc, pour les fonctions de contrôle `if`, `while`, `for`, `foreach` et `switch`. Dans chaque cas, le principe est de remplacer l'accolade d'ouverture par deux points (:) et l'accolade de fermeture par, respectivement, `endif;`, `endwhile;`, `endfor;`, ou `endswitch;`.

```
<?php if ($a == 5): ?>
 A vaut 5
<?php endif; ?>
```

Dans l'exemple ci-desssus, le block HTML "A = 5" est inclus à l'intérieur d'un `if` en utilisant cette nouvelle syntaxe. Ce code HTML ne sera affiché que si la variable $a est égale à 5.

Cette autre syntaxe fonctionne aussi avec le `else` et `elseif`. L'exemple suivant montre une structure avec un `if`, un `elsif` et un `else` utilisant cette autre syntaxe:

```
<?php
if ($a == 5):
    print "a égale 5";
    print "...";
elseif ($a == 6):
    print "a égale 6";
```

```
    print "!!!";
else:
    print "a ne vaut ni 5 ni 6";
endif;
?>
```

Allez voir while, for, et if pour d'autres exemples.

while

La boucle `while` est le moyen le plus simple d'implémenter une boucle en PHP. Cette boucle se comporte de la même manière qu'en C. L'exemple le plus simple d'une boucle `while` est le suivant :

```
<?php
while (expression) commandes
?>
```

La signification d'une boucle `while` est très simple. Le PHP exécute l'instruction tant que l'expression de la boucle `while` est évaluée comme `TRUE`. La valeur de l'expression est vérifiée à chaque début de boucle, et, si la valeur change durant l'exécution de l'instruction, l'exécution ne s'arrêtera qu'à la fin de l'itération (chaque fois que le PHP exécute l'instruction, on appelle cela une itération). De temps en temps, si l'expression du `while` est `FALSE` avant la première itération, l'instruction ne sera jamais exécutée.

Comme avec le `if`, vous pouvez regrouper plusieurs instructions dans la même boucle `while` en les regroupant à l'intérieur de parenthèses ou en utilisant la syntaxe suivante:

```
<?php
while (expression): commandes ... endwhile;
?>
```

Les exemples suivants sont identiques, et affichent tous les nombres de 1 à 10:

```
<?php
/* exemple 1 */
$i = 1;
while ($i <= 10) {
    print $i++;  /* La valeur affiche est $i avant l'incrémentation
                    (post-incrémentation)  */
}
/* exemple 2 */
$i = 1;
while ($i <= 10):
    print $i;
    $i++;
endwhile;
?>
```

do..while

Les boucles `do..while` ressemblent beaucoup aux boucles `while`, mais l'expression est testée à la fin de chaque itération plutôt qu'au début. La principale différence par rapport à la boucle `while` est que la première itération de la boucle `do..while` est toujours exécutée (l'expression n'est testée qu'à la fin de l'itération), ce qui n'est pas le cas lorsque vous utilisez une boucle `while` (l'expression est vérifiée au début de chaque itération).

Il n'y a qu'une syntaxe possible pour les boucles `do..while`:

```
<?php
$i = 0;
do {
   print $i;
} while ($i>0);
?>
```

La boucle ci-dessus ne va être exécutée qu'une seule fois, car lorsque l'expression est évaluée, elle vaut `FALSE` (car la variable $i n'est pas plus grande que 0) et l'exécution de la boucle s'arrête.

Les utilisateurs familiers du C sont habitués à une utilisation différente des boucles `do..while` , qui permet de stopper l'exécution de la boucle au milieu des instructions, en l'encapsulant dans un `do..while`(0) la fonction `break`. Le code suivant montre une utilisation possible:

```
<?php
do {
    if ($i < 5) {
        print "i n'est pas suffisamment grand";
        break;
    }
    $i *= $factor;
    if ($i < $minimum_limit) {
        break;
    }
    print "i est bon";
     ...process i...
} while(0);
?>
```

Ne vous inquiétez pas si vous ne comprenez pas tout correctement. Vous pouvez écrire des scripts très très puissants sans utiliser cette fonctionnalité.

for

Les boucles `for` sont les boucles les plus complexes en PHP. Elles fonctionnent comme les boucles `for` du langage C. La syntaxe des boucles for est la suivante:

```
<?php
for (expr1; expr2; expr3) statement
?>
```

La première expression (*`expr1`*) est évaluée (exécutée), quoi qu'il arrive au début de la boucle.

Au début de chaque itération, l'expression *`expr2`* est évaluée. Si l'évaluation vaut `TRUE`, la boucle continue et l'instruction est exécutée. Si l'évaluation vaut `FALSE`, l'exécution de la boucle s'arrête.

A la fin de chaque itération, l'expression *`expr3`* est évaluée (exécutée).

Les expressions peuvent éventuellement être laissées vides. Si l'expression *`expr2`* est laissée vide, cela signifie que c'est une boucle infinie (PHP considère implicitement qu'elle vaut `TRUE`, comme en C). Cela n'est pas vraiment très utile, à moins que vous souhaitiez terminer votre boucle par l'instruction conditionnelle `break`.

Considérons les exemples suivants. Tous affichent les chiffres de 1 à 10:

```
<?php
/* exemple 1 */
for ($i = 1; $i <= 10; $i++) {
```

```
    print $i;
}
/* exemple 2 */
for ($i = 1;;$i++) {
    if ($i > 10) {
        break;
    }
    print $i;
}
/* exemple 3 */
$i = 1;
for (;;) {
    if ($i > 10) {
        break;
    }
    print $i;
    $i++;
}
/* exemple 4 */
for ($i = 1; $i <= 10; print $i, $i++) ;
?>
```

Bien évidemment, le premier exemple est le plus simple de tous (ou peut être le quatrième), mais vous pouvez aussi pensez qu'utiliser une expression vide dans une boucle `for` peut être utile parfois.

PHP supporte aussi la syntaxe alternative suivante pour les boucles `for` :

```
<?php
for (expr1; expr2; expr3): statement; ...; endfor;
?>
```

Les autres langages ont l'instruction `foreach` pour accéder aux éléments d'un tableau. PHP 3 ne dispose pas d'une telle fonction; PHP 4 en dispose (voir foreach). En PHP 3, vous pouvez combiner while avec **list()** et **each()** pour obtenir le même résultat. Reportez-vous aux exemples de la documentation.

`foreach`

PHP 4 (mais pas PHP 3) inclut une commande `foreach`, comme en Perl ou d'autres langages. C'est un moyen simple de passer en revue un tableau. Il y a deux syntaxes possibles : la seconde est une extension mineure mais pratique de la première:

```
<?php
  foreach(array_expression as $value) commandes
  foreach(array_expression as $key => $value) commandes
?>
```

La première forme passe en revue le tableau `array_expression`. A chaque itération, la valeur de l'élément courant est assignée à `$value` et le pointeur interne de tableau est avancé d'un élément (ce qui fait qu'à la prochaine itération, on accédera à l'élément suivant).

La deuxième forme fait exactement la même chose, mais c'est la clé de l'élément courant qui est assigné à la variable `$key`.

Lorsque `foreach` démarre, le pointeur interne de fichier est automatiquement ramené au premier élément du tableau. Cela signifie que vous n'aurez pas à faire appel à **reset()** avant `foreach`.

> **Note :** De plus, notez que `foreach` travaille sur une copie du tableau spécifié, et pas sur le tableau lui-même. Par conséquent, le pointeur de tableau n'est pas modifié, comme il le serait avec le fonction **each()**, et les modifications faites dans le tableau ne seront pas prises en compte dans le tableau original.

Note : `foreach` n'accepte pas l'opérateur de supppression des erreurs @.

Vous pouvez remarquer que les exemples suivants fonctionnent de manière identique :

```
<?php
  reset($arr);
  while (list(, $value) = each ($arr)) {
    echo "Valeur: $value<br>\n";
  }
  foreach ($arr as $value) {
    echo "Valeur: $value<br>\n";
  }
?>
```

Les exemples suivants sont aussi fonctionnellement identiques :

```
<?php
  reset($arr);
  while (list($key, $value) = each ($arr)) {
    echo "Clé: $key; Valeur: $value<br>\n";
  }
  foreach ($arr as $key => $value) {
    echo "Clé: $key; Valeur: $value<br>\n";
  }
?>
```

Voici quelques exemples de plus :

```
<?php
/* exemple 1: valeurs seules */
$a = array (1, 2, 3, 17);
foreach ($a as $v) {
   print "Valeur courante de \$a: $v.\n";
}
/* exemple 2: valeurs (avec la clé correspondante) */
$a = array (1, 2, 3, 17);
$i = 0; /* pour l'illustration uniquement */
foreach($a as $v) {
    print "\$a[$i] => $v.\n";
}
/* exemple 3: clé et valeur */
$a = array (
    "un" => 1,
    "deux" => 2,
    "trois" => 3,
    "dix-sept" => 17
);
foreach($a as $k => $v) {
    print "\$a[$k] => $v.\n";
}
/* exemple 4: tableaux multi-dimensionnels */
$a[0][0] = "a";
$a[0][1] = "b";
$a[1][0] = "y";
$a[1][1] = "z";
foreach($a as $v1) {
    foreach ($v1 as $v2) {
        print "$v2\n";
    }
}
```

```
/* exemple 5: tableaux dynamique */
foreach(array(1, 2, 3, 4, 5) as $v) {
    print "$v\n";
}
?>
```

break

L'instruction `break` permet de sortir d'une structure `for`, `while`, `foreach` ou `switch`.

`break` accepte un argument numérique optionnel qui vous indiquera combien de structures emboitées ont été interrompues.

```
<?php
$i = 0;
while ($i < 10) {
    if ($arr[$i] == "stop") {
        break;  /* Vous pouvez aussi écrire 'break 1;' ici. */
    }
    $i++;
}
/* Utilisation de l'argument optionnel. */
$i = 0;
while ( ++$i ) {
    switch ( $i ) {
    case 5:
        echo "à 5<br>\n";
        break 1;  /* Ne sort que du switch. */
    case 10:
        echo "à 10; quitting<br>\n";
        break 2;  /* Sort du switch et du while. */
    default:
        break;
    }
}
?>
```

continue

L'instruction `continue` est utilisée dans une boucle afin d'éluder les instructions de l'itération courante afin de passer directement à l'itération suivante.

`continue` accepte un argument numérique optionnel qui vous indiquera combien de structures emboitées ont été ignorées.

```
<?php
while (list ($cle, $valeur) = each ($arr)) {
   if (!($cle % 2)) { // évite les membres impairs
       continue;
   }
   fonction_quelconque($valeur);
}
$i = 0;
while ($i++ < 5) {
    echo "Dehors<br>\n";
    while (1) {
        echo "  Milieu<br>\n";
        while (1) {
             echo "  Intérieur<br>\n";
             continue 3;
```

```
        }
        echo "Ceci n'est jamais atteint.<br>\n";
    }
    echo "Ceci non plus.<br>\n";
}
?>
```

switch

L'instruction `switch` équivaut à une série d'instructions `if`. En de nombreuses occasions, vous aurez besoin de comparer la même variable (ou expression) avec un grand nombre de valeurs différentes, et d'exécuter différentes parties de code suivant la valeur à laquelle elle est égale. C'est exactement à cela que sert l'instruction `switch`.

Les deux exemples suivants sont deux manières différentes d'écrire la même chose, l'une en utilisant une séries de `if`, et l'autre en utilisant l'instruction `switch`:

```
<?php
if ($i == 0) {
    print "i égale 0";
}
if ($i == 1) {
    print "i égale 1";
}
if ($i == 2) {
    print "i égale 2";
}
switch ($i) {
    case 0:
        print "i égale 0";
        break;
    case 1:
        print "i égale 1";
        break;
    case 2:
        print "i égale 2";
        break;
}
?>
```

Il est important de comprendre que l'instruction `switch` exécute chacune des clauses dans l'ordre. L'instruction `switch` est exécutée ligne par ligne. Au début, aucun code n'est exécuté. Seulement lorsqu'un `case` est vérifié, PHP exécute alors les instructions correspondantes. PHP continue d'exécuter les instructions jusqu'à la fin du bloc d'instructions du `switch`, ou bien dès qu'il trouve l'instruction `break`. Si vous ne pouvez pas utiliser l'instruction `break` à la fin de l'instruction `case`, PHP continuera à exécuter toutes les instructions qui suivent. Par exemple :

```
<?php
switch ($i) {
    case 0:
        print "i égale 0";
    case 1:
        print "i égale 1";
    case 2:
        print "i égale 2";
}
?>
```

Dans cet exemple, si $i est égal à 0, PHP va exécuter quand même toutes les instructions qui suivent. Si $i est égal à 1, PHP exécutera les deux dernières instructions. Et seulement si $i est égal à 2, vous obtiendrez le résultat escompté, c'est-à-dire, l'affiche de "i égal 2". Donc, l'important est de ne pas oublier l'instruction `break` (même s'il est possible que vous l'omettiez dans certaines circonstances).

Dans une commande `switch`, une condition n'est évaluée qu'une fois, et le résultat est comparé à chaque `case`. Dans une structure `elseif`, les conditions sont évaluées à chaque comparaison. Si votre condition est plus compliquée qu'une simple comparaison, ou bien fait partie d'une boucle, `switch` sera plus rapide.

La liste de commandes d'un `case` peut être vide, auquel cas PHP utilisera la liste de commandes du cas suivant.

```
<?php
switch ($i) {
    case 0:
    case 1:
    case 2:
        print "i est plus petit que 3 mais n'est pas négatif";
        break;
    case 3:
        print "i égale 3";
}
?>
```

Un `case` spécial est `default`. Ce cas est utilisé lorsque tous les `case` ont échoués. Il doit être le dernier cas listé. Par exemple :

```
<?php
switch ($i) {
    case 0:
        print "i égale 0";
        break;
    case 1:
        print "i égale 1";
        break;
    case 2:
        print "i égale  2";
        break;
    default:
        print "i n'est ni égal à 2, ni à 1, ni à 0.";
}
?>
```

Une autre chose à mentionner est que l'instruction `case` peut être une expression à de type scalaire, c'est-à-dire nombre entier, nombre à virgule flottante et chaîne de caractères. Les tableaux sont sans intérêt dans ce contexte-là.

La syntaxe alternative pour cette structure de contrôle est la suivante : pour plus d'informations, voir syntaxes alternatives).

```
<?php
  switch ($i):
    case 0:
        print "i égale 0";
        break;
    case 1:
        print "i égale 1";
        break;
    case 2:
        print "i égale 2";
        break;
    default:
        print "i n'est ni égal à 2, ni à 1, ni à 0";
  endswitch;
```

```
?>
```

declare

L'élément de langage `declare` sert à ajouter des directives d'éxécutions dans un bloc de code. La syntaxe de `declare` est similaire à la syntaxe des autres fonctions de contrôle :

```
<?php
  declare (directive) statement
?>
```

L'expression `directive` permet de contrôler l'intervention du bloc `declare`. Actuellement, une seule directive est reconnue : la directive `ticks` (Voir plus bas pour plus de détails) sur les ticks).

L'expression `statement` du bloc de `declare` sera exécutée. Comment elle sera exécutée, et quels effets cela aura dépend de la directive utilisée dans le bloc `directive`.

Ticks

Un tick est un événement qui intervient toutes les *N* commandes bas niveau, exécutées par l'analyseur dans le bloc de `declare`. La valeur de *N* est spécifiée avec la syntaxe `ticks=`*N* dans le bloc de directive `declare`.

Un événement qui intervient à chaque tick est spécifié avec la fonction **register_tick_function()**. Reportez vous à l'exemple ci-dessous pour plus de détails. Notez que plus d'un événement peut intervenir par tick.

```
<PRE>
<?php
// Un fonction qui enregistre l'heure à laquelle elle est appelée
  function profile($dump = FALSE){
    static $profile;
    // Retourne les horaires stockés dans le profile, et l'efface
    if ($dump) {
        $temp = $profile;
        unset ($profile);
        return ($temp);
    }
    $profile[] = microtime ();
  }
// Enregistre un gestionnaire de tick
  register_tick_function("profile");
// Initialise la fonction avant le bloc de déclaration
  profile();
// Exécute un bloc de code, et appele un tick toutes les deux secondes
  declare (ticks=2) {
    for ($x = 1; $x < 50; ++$x) {
        echo similar_text(md5($x), md5($x*$x)), "<br>";
    }
  }
?>
```

Pour voir le résultat :

```
<?php
// Affiche les données de la variable $profile
  print_r(profile(TRUE));
?>
</pre>
```

Cet exemple profile le code PHP dans le bloc de déclaration, et enregistre l'heure de chaque commande bas niveau. Cette information peut être réutilisée pour débusquer les segments de code lents. Vous pouvez implémenter d'autres méthodes, mais les ticks sont plus rapides et plus efficaces.

Les ticks sont bien pratiques pour débugger, implémenter un multi-tâches simple, des entrées sorties en tâche de fond, ou bien d'autres choses, avec PHP.

Voir aussi **register_tick_function()** et **unregister_tick_function()**.

require()

La commande **require()** se remplace elle-même par le contenu du fichier spécifié, comme les préprocesseurs C le font avec la commande `#include`.

Il est important de noter que lorsqu'un fichier est **include()** ou **require()**, les erreurs d'analyse apparaîtront en HTML tout au début du fichier, et l'analyse du fichier parent ne sera pas interrompue. Pour cette raison, le code qui est dans le fichier doit être placé entre les balises habituelles de PHP.

require() n'est pas vraiment une fonction PHP : c'est plus une instruction du langage. Elle ne fonctionne pas comme les fonctions standards. Par exemple, **require()** est indépendante des structures de contrôle (cela ne sert à rien de la placer dans une condition, elle sera toujours exécutée). De plus, elle ne retourne aucune valeur. Lire une valeur retournée par un **require()** retourne une erreur d'analyse.

Contrairement à **include()**, **require()** va *toujours* lire dans le fichier cible, même si la ligne n'est jamais exécutée. Si vous souhaitez une inclusion conditionnelle, utilisez **include()**. La condition ne va jamais affecter **require()**. Cependant, si la ligne de **require()** n'est jamais exécutée, le code du fichier ne le sera jamais non plus.

Les boucles n'affectent pas le comportement de **require()**. Même si le code contenu dans le fichier source est appelé dans la boucle, **require()** n'est exécuté qu'une fois.

Cela signifie qu'on ne peut pas mettre un **require()** dans une boucle, et s'attendre à ce qu'il inclue du code à chaque itération. Pour cela, il faut utiliser **include()**.

```
<?php
  require('header.inc');
?>
```

Attention : **include()** et **require()** ajoutent le contenu du fichier cible dans le script lui-même. Elles n'utilisent pas le protocole HTTP ou tout autre protocole. Toute variable qui est dans le champs du script sera accessible dans le fichier d'inclusion, et vice-versa.

```
<?php
  require ("file.inc?varone=1&vartwo=2");
/* Ne fonctionne pas. */
  $varone = 1;
  $vartwo = 2;
  require ("file.inc");
/* $varone et $vartwo seront accessibles à file.inc */
?>
```

Ne vous laissez pas abuser par le fait que vous pouvez requérir ou inclure des fichiers via HTTP en utilisant la fonctionnalité de gestion des fichiers distants ce qui est au dessus reste vrai.

En PHP 3, il est possible d'exécuter une commande `return` depuis un fichier inclut, tant que cette commande intervient au niveau global du fichier inclus. Elle ne doit intervenir dans aucun bloc (entre accolade {}). En PHP 4, cette possibilité a été supprimée. Si vous en avez besoin, utilisez plutôt **include()**.

include()

La fonction **include()** inclus et évalue le fichier spécifié en argument.

Il est important de noter que lorsqu'un fichier est **include()** ou **require()**, les erreurs d'analyse apparaîtront en HTML tout au début du fichier, et l'analyse du fichier parent ne sera pas interrompue. Pour cette raison, le code qui est dans le fichier doit être placé entre les balises habituelles de PHP.

Cela se produit à chaque fois que la fonction **include()** est rencontrée. Donc, vous pouvez utiliser la fonction **include()** dans une boucle pour inclure un nombre infini de fois un fichier, ou même des fichiers différents.

```
<?php
$files = array ('premier.inc', 'second.inc', 'troisieme.inc');
for ($i = 0; $i < count($files); $i++) {
    include $files[$i];
}
?>
```

include() diffère de **require()** car le fichier inclus est ré-évaluée à chaque fois que la commande est exécutée, tandis que **require()** est remplacée par le fichier cible lors de la première exécution, que son contenu soit utilisé ou non. De plus, cela se fait même s'il est placé dans une structure conditionnelle, comme dans un if).

Parceque la fonction **include()** nécessite une construction particulière, vous devez l'inclure dans un bloc si elle est incluse dans une structure conditionnelle.

```
<?php
/*  Ceci est faux, et ne fonctionnera pas comme on l'attend. */
if ($condition)
    include($file);
else
    include($other);
/* Ceci est CORRECT. */
if ($condition) {
    include($file);
} else {
    include($other);
}
?>
```

En PHP 3, il est possible d'exécuter une commande `return` depuis un fichier inclus, tant que cette commande intervient au niveau global du fichier inclus. Elle ne doit intervenir dans aucun bloc (entre accolade {}). En PHP 4, cette possibilité a été supprimée. Cependant, PHP 4 vous autorise à retourner des valeurs d'un fichier inclus. Vous pouvez traiter **include()** comme une fonction normale, qui retourne une valeur. Mais cela génère une erreur d'analyse en PHP 3.

Exemple 11-1. include() en PHP 3 et PHP 4

On suppose que le fichier `test.inc` existe, et est placé dans le même dossier que le fichier principal :

```
<?php
echo "Avant le retour<br>\n";
if (1) {
    return 27;
}
echo "Après le retour <br>\n";
?>
```

On suppose que le fichier `main.html` contient ceci :

```
<?php
$retval = include ('test.inc');
echo "Fichier inclus: '$retval'<br>\n";
?>
```

Lorsque `main.html` est appelé en PHP 3, il va générer une erreur d'analyse (parse error) à la ligne 2; vous ne pouvez pas vous attendre à un retour sur une fonction **include()** en PHP 3. En PHP 4, cependant,le résultat sera :

```
Avant le retour
Ficher inclus : '27'
```

Supposons maintenant que `main.html` a été modifié et contient maintenant le code suivant :

```
<?php
include ('test.inc');
echo "Retour dans le main.html<br>\n";
?>
```

En PHP 4, l'affichage sera :

```
Avant le retour
Retour dans le main.html
```

Au contraire, PHP 3 affichera :

```
Avant le retour
27Retour dans le main.html
Parse error: parse error in /home/torben/public_html/phptest/main.html on line 5
```

L'erreur d'analyse ci-dessus est le résultat du fait que la commande `return` est dans un bloc qui n'est pas une fonction, dans `test.inc`. Lorsque le return est sorti du bloc, l'affichage devient :

```
Avant le retour
27Retour dans le main.html
```

Le '27' est du au fait que PHP 3 ne supporte pas le `return` dans ces fichiers.

Il est important de noter que lorsqu'un fichier est **include()** ou **require()**, les erreurs d'analyse apparaîtront en HTML tout au début du fichier, et l'analyse du fichier parent ne sera pas interrompue. Pour cette raison, le code qui est dans le fichier doit être placé entre les balises habituelles de PHP.

```
<?php
include ("file.inc?varone=1&vartwo=2"); /* ne fonctionne pas. */
$varone = 1;
$vartwo = 2;
include ("file.inc");  /* $varone et $vartwo sont accessibles dans file.inc */
?>
```

Ne vous laissez pas abuser par le fait que vous pouvez requérir ou inclure des fichiers via HTTP en utilisant la fonctionnalité de gestion des fichiers distants ce qui est au dessus reste vrai.

Voir aussi **readfile()**, **require()** et **virtual()**.

require_once()

La commande **require_once()** se remplace elle-même par le fichier spécifié, un peu comme les commandes de préprocesseur C `#include`, et ressemble sur ce point à **require()**. La principale différence est qu'avec **require_once()**, vous êtes assurés que ce code ne sera ajouté qu'une seule fois, évitant de ce fait les redéfinitions de variables ou de fonctions, génératrices d'alertes.

Par exemple, si vous créez les deux fichiers d'inclusion `utils.inc` et `foolib.inc`

Exemple 11-2. utils.inc

```
<?php
define(PHPVERSION, floor(phpversion()));
echo "LES GLOBALES SONT SYMPAS\n";
function goodTea() {
return "Le Earl Grey est délicieux!";
}
?>
```

Exemple 11-3. foolib.inc

```
<?php
require ("utils.inc");
function showVar($var) {
if (PHPVERSION == 4) {
print_r($var);
} else {
dump_var($var);
}
}
// Une série de fonctions
?>
```

Puis, vous écrivez un script `cause_error_require.php`

Exemple 11-4. cause_error_require.php

```
<?php
require("foolib.inc");
/* Ceci génère une erreur*/
require("utils.inc");
$foo = array("1",array("complex","quaternion"));
echo "Ce code requiert utils.inc une deuxième fois, car il est requis \n";
echo "dans foolib.inc\n";
echo "Utilisation de GoodTea: ".goodTea()."\n";
echo "Affichage de foo: \n";
showVar($foo);
?>
```

Lorsque vous exécutez le script ci-dessus, le résultat sera (sous PHP 4.01pl2):

```
GLOBALS ARE NICE
GLOBALS ARE NICE
Fatal error:  Cannot redeclare causeerror() in utils.inc on line 5
```

En modifiant `foolib.inc` et `cause_errror_require.php` pour qu'elles utilisent **require_once()** au lieu de **require()** et ne renommant pas le fichier en `avoid_error_require_once.php`, on obtient :

Exemple 11-5. foolib.inc (corrigé)

```
<?php
require_once("utils.inc");
function showVar($var) {
?>
```

Exemple 11-6. avoid_error_require_once.php

```
<?php
require_once("foolib.inc");
require_once("utils.inc");
$foo = array("1",array("complexe","quaternion"));
?>
```

L'exécution de ce script, sous PHP 4.0.1pl2, donne :

```
LES GLOBALES SONT SYMPA
Ce code requiert utils.inc une deuxième fois, car il est requis
dans foolib.inc
Utilisation de GoodTea: Le Earl Grey est délicieux!
Affichage de foo:
Array
(
    [0] => 1
    [1] => Array
        (
            [0] => complexe
            [1] => quaternion
        )
)
```

Notez aussi que, de la même manière que les préprocesseurs traitent les `#include`, cette commande est exécutée au moment de la compilation, c'est-à-dire lorsque le script est analysée, et avant qu'il soit exécuté, et ne doit pas être utilisée pour insérer des données dynamiques liées à l'éxécution. Il vaut alors mieux utiliser **include_once()** ou **include()**.

Pour plus d'exemples avec **require_once()** et **include_once()**, jetez un oeil dans le code de PEAR inclus dans la dernière distribution de PHP.

Voir aussi : **require()**, **include()**, **include_once()**, **get_required_files()**, **get_included_files()**, **readfile()**, et **virtual()**.

include_once()

La commande **include_once()** inclut et évalue le fichier spécifié durant l'exécution du script. Le comportement est similaire à **include()**, mais la différence est que si le code a déjà été inclus, il ne le sera pas une seconde fois.

Comme précisé dans la section **require_once()**, la fonction **include_once()** est utilisée de préférence lorsque le fichier doit être inclus ou évalué plusieurs fois dans un script, ou bien lorsque vous voulez être sÛr qu'il ne sera inclus qu'une seule fois, pour éviter des redéfinitions de fonction.

Pour plus d'exemples avec **require_once()** et **include_once()**, jetez un oeil dans le code de PEAR inclus dans la dernière distribution de PHP.

Voir aussi: **require()**, **include()**, **require_once()**, **get_required_files()**, **get_included_files()**, **readfile()**, et **virtual()**.

Chapitre 12. Les fonctions

Les fonctions utilisateur

Une fonction peut être définie en utilisant la syntaxe suivante :

```
<?php
function foo ($arg_1, $arg_2, ..., $arg_n) {
    echo "Exemple de fonction.\n";
    return $retval;
}
?>
```

Tout code PHP, correct syntaxiquement, peut apparaître dans une fonction et dans une définition de classe.

En PHP 3, les fonctions doivent être définies avant qu'elles ne soient utilisées. Ce n'est plus le cas en PHP 4.

PHP ne supporte pas le surchargement de fonction, ni la destruction ou la redéfinition de fonctions déjà déclarées.

PHP 3 ne supporte pas un nombre variable d'arguments (voir valeurs par défault d'arguments pour plus d'informations). PHP 4 supporte les deux : voir liste variable d'arguments de fonction et les fonctions de références que sont **func_num_args()**, **func_get_arg()**, et **func_get_args()** pour plus d'informations.

Les arguments de fonction

Des informations peuvent être passées à une fonction en utilisant un tableau d'arguments, dont chaque élément est séparé par une virgule. Un élément peut être une variable ou une constante.

PHP supporte le passage d'arguments par valeur (méthode par défaut), par référence. Les listes variables d'arguments sont supportées par PHP 4 et les versions plus récentes. Voir liste variable d'arguments de fonction et les fonctions utiles que sont **func_num_args()**, **func_get_arg()**, et **func_get_args()**. Fonctionnellement, on peut arriver au même résultat en passant un tableau comme argument :

```
function takes_array($input) {
    echo "$input[0] + $input[1] = ", $input[0]+$input[1];
}
```

Passage d'arguments par référence

Par défaut, les arguments sont passés à la fonction par valeur (donc vous pouvez changer la valeur d'un argument dans la fonction, cela ne change pas sa valeur à l'extérieur de la fonction). Si vous voulez que vos fonctions puissent changer la valeur des arguments, vous devez passer ces arguments par référence.

Si vous voulez qu'un argument soit toujours passé par référence, vous pouvez ajouter un '&' devant l'argument dans la déclaration de la fonction :

```
function add_some_extra(&$string) {
    $string .= ', et un peu plus.';
}
$str = 'Ceci est une chaîne';
add_some_extra($str);
echo $str;    // affiche 'Ceci est une chaîne, et un peu plus.'
```

Si vous souhaitez passer une variable par référence à une fonction mais de manière ponctuelle, vous pouvez ajouter un '&' devant l'argument dans l'appel de la fonction:

```
function foo ($bar) {
    $bar .= ', et un peu plus.';
}
$str = Ceci est une chaîne';
```

```
foo ($str);
echo $str;    // affiche 'Ceci est une chaîne'
foo (&$str);
echo $str;    // affiche 'Ceci est une chaîne, et un peu plus.'
```

Valeur par défaut des arguments

Vous pouvez définir comme en C++ des valeurs par défaut pour les arguments de type scalaire :

```
function servir_apero ($type = "ricard") {
    return "Servir un verre de $type.\n";
}
echo servir_apero();
echo servir_apero("whisky");
```

La fonction ci-dessus affichera :

```
Servir un verre de ricard.
Servir un verre de whisky.
```

La valeur par défaut d'un argument doit obligatoirement être une constante, et ne peut être ni une variable, ni un membre de classe.

Il est à noter que si vous utilisez des arguments avec valeur par défaut avec d'autres sans valeur par défaut, les premiers doivent être placés à la suite de tous les paramètres sans valeur par défaut. Sinon, cela ne fonctionnera pas. Considérons le code suivant :

```
<?php
function faireunyaourt ($type = "acidophilus", $flavour) {
    return "Préparer un bol de $type $flavour.\n";
}
echo faireunyaourt ("framboise");   // ne fonctionne pas comme voulu
?>
```

L'affiche du code ci-dessus est le suivant :

```
Warning: Missing argument 2 in call to faireunyaourt() in
/usr/local/etc/httpd/htdocs/PHP 3test/functest.html on line 41
Préparer un bol de framboise.
```

Maintenant comparons l'exemple précédent avec l'exemple suivant :

```
<?php
function faireunyaourt ($flavour, $type = "acidophilus") {
    return "Préparer un bol de $type $flavour.\n";
}
echo faireunyaourt ("framboise");   // fonctionne comme voulu
?>
```

L'affichage de cet exemple est le suivant :

```
Préparer un bol de acidophilus framboise.
```

Nombre d’arguments variable

PHP 4 supporte les fonctions à nombre d’arguments variable. C’est très simple à utiliser, avec les fonctions **func_num_args()**, **func_get_arg()**, et **func_get_args()**.

Aucune syntaxe particulière n’est nécessaire, et la liste d’argument doit toujours être fournie explicitement avec la définition de la fonction, et se comportera normalement.

Les valeurs de retour

Les valeurs sont renvoyées en utilisant une instruction de retour optionnelle. Tous les types de variables peuvent être renvoyés, tableaux et objets compris.

```
<?php
function carre ($num) {
    return $num * $num;
}
echo carre (4);   // affiche ’16’.
?>
```

Vous ne pouvez pas renvoyer plusieurs valeurs en même temps, mais vous pouvez obtenir le même résultat en renvoyant un tableau.

```
<?php
function petit_nombre() {
    return array (0, 1, 2);
}
list ($zero, $one, $two) = petit_nombre();
?>
```

Pour retourner une référence d’une fonction, utilisez l’opérateur & aussi bien dans la déclaration de la fonction que dans l’assignation de la valeur de retour.

```
<?php
function &retourne_reference() {
    return $uneref;
}
$newref =&retourne_reference();
?>
```

`old_function`

L’instruction `old_function` vous permet de déclarer une fonction en utilisant une syntaxe du type PHP/FI2 (au détail près que vous devez remplacer l’instruction ’function’ par ’old_function’.)

C’est une fonctionnalité obsolète et elle ne devrait être utilisée que dans le cadre de conversion de PHP/FI2 vers PHP 3

> **Avertissement**
>
> Les fonctions déclarées comme `old_function` ne peuvent pas être appelées à partir du code interne du PHP. Cela signifie, par exemple, que vous ne pouvez pas les utiliser avec des fonctions comme **usort()**, **array_walk()**, et **register_shutdown_function()**. Vous pouvez contourner ce problème en écrivant une fonction d'encapsulation qui appellera la fonction `old_function`.

Fonctions-variable

PHP supporte le concept de fonctions variables. Cela signifie que si le nom d'une variable est suivi de parenthèses, PHP recherchera une fonction de même nom, et essaiera de l'exécuter. Cela peut servir, entre autre, pour faire des fonctions call-back, des tables de fonctions...

Les fonctions-variables ne peuvent pas fonctionner avec les éléments de langage comme les **echo()**, **unset()**, **isset()** et **empty()**. C'est une des différences majeures entre les fonctions PHP et les éléments de langage.

Exemple 12-1. Exemple de fonction variable

```
<?php
function foo() {
    echo "dans foo()<br>\n";
}
function bar( $arg = ” ) {
    echo "Dans bar(); l’argument était ’$arg’.<br>\n";
}
$func = ’foo’;
$func();
$func = ’bar’;
$func( ’test’ );
?>
```

Chapitre 13. Les classes et les objets

Les classes : `class`

Une classe est une collection de variables et de fonctions qui fonctionnent avec ces variables. Une classe est définie en utilisant la syntaxe suivante :

```
<?php
class Caddie {
    var $items;  // Eléments de notre panier
    // Ajout de $num articles de type $artnr au panier
    function add_item ($artnr, $num) {
        $this->items[$artnr] += $num;
    }
    // Suppression de $num articles du type $artnr du panier
    function remove_item ($artnr, $num) {
        if ($this->items[$artnr] > $num) {
            $this->items[$artnr] -= $num;
            return TRUE;
        } else {
            return FALSE;
        }
    }
}
?>
```

L'exemple ci-dessus définit la classe Caddie qui est composée d'un tableau associatif contenant les articles du panier et de deux fonctions, une pour ajouter et une pour enlever des éléments au panier.

Attention

Les notes suivantes ne sont valable que pour PHP 4.

Le nom `stdClass` est utilisé en interne par Zend et ne doit pas être utilisé. Vous ne pouvez pas nommer une classe `stdClass` en PHP.

Les noms de fonctions `__sleep` et `__wakeup` sont magiques en PHP. Vous ne pouvez pas utiliser ces noms de fonctions dans vos classes, à moins que vous ne souhaitiez utiliser la magie qui y est associée.

PHP se réserve l'usage de tous les noms de fonctions commençants par __, pour sa propre magie. Il est vivement recommandé de ne pas utilser des noms de fonctions commençants par __, à moins que vous ne souhaitiez utiliser la magie qui y est associée.

Note : En PHP 4, seuls les initialiseurs constants pour les variables `var` sont autorisés. Utilisez les constructeurs pour les initialisations variables, ou utilisant des expressions.

```
<?php
/* Aucune de ces syntaxes ne fonctionnera en PHP 4 */
class Caddie {
    var $date_du_jour = date("d/m/Y");
    var $name = $firstname;
    var $owner = 'Fred ' . 'Jones';
    var $items = array("DVD", "Télé","Magnétoscope");
}
/* Voici comment cela doit se faire désormais. */
class Caddie {
    var $date_du_jour;
    var $name;
    var $owner;
    var $items;
    function Caddie() {
        $this->date_du_jour = date("d/m/Y");
        $this->name = $GLOBALS['firstname'];
        /* etc... */
    }
}
?>
```

Les classes forment un type de variable. Pour créer une variable du type désiré, vous devez utiliser l'opérateur new.

```
<?php
 $cart = new Caddie;
 $cart->add_item("10", 1);
 $another_cart = new Cart;
 $another_cart->add_item("0815", 3);
?>
```

L'instruction ci-dessus crée l'objet $cart de la class Caddie. La fonction add_idem() est appelée afin d'ajouter l'article numéro 10 dans le panier. 3 articles numéro 0815 sont ajoutés au cart $another_cart.

$cart et $another_cart disposent des fonctions add_item(), remove_item() et de la variable items. Ce sont des fonctions et variables distinctes. Vous pouvez vous représenter les objets comme des dossiers sur votre disque dur. Vous pouvez avoir deux fichiers "lisez-moi.txt" sur votre disque dur, tant qu'ils ne sont pas dans le même répertoire. De même que vous devez alors taper le chemin complet jusqu'au fichier, vous devez spécifier le nom complet de la méthode avant de l'employer : en termes PHP, le dossier racine est l'espace de nom global, et le séparateur de dossier est `->`. Par exemple, les noms $cart->items et $another_cart->items représentent deux variables distinctes. Notez que le nom de la variable est alors $cart->items, et non pas $cart->$items : il n'y a qu'un seul signe $ dans un nom de variable.

```
<?php
// correct, $ unique
$cart->items  = array("10" => 1);
// incorrect, car $cart->$items devient $cart->""
$cart->$items = array("10" => 1);
// correct, mais risque de ne pas se comporter comme prévu
// $cart->$myvar devient $ncart->items
$myvar = 'items';
$cart->$myvar = array("10" => 1);
?>
```

A l'intérieur d'une définition de classe, vous ne savez pas le nom de la variable à partir duquel l'objet sera accessible dans le script. On ne peut prévoir que l'objet créé sera affecté à la variable $cart ou $another_cart. Donc, vous ne pouvez pas utiliser la syntaxe $cart->items. Mais pour pouvoir accéder à aux méthodes et membres d'un objet, vous pouvez utiliser la variable spéciale $this, qui peut s'interpréter comme 'moi-même', ou bien 'l'objet courant'. Par exemple, `'$this->items[$artnr] += $num;'` peut se lire comme 'ajouter $num au compteur $artnr de mon propre tableau de compteur' ou bien 'ajouter $num au compteur $artnr du tableau de compteurs de l'objet courant'.

`extends` : héritage

Souvent, vous aurez besoin d'une classe avec des méthodes et fonctions similaires à une autre classe. En fait, il est bon de définir des classes génériques, qui pourront être réutilisées et adaptées à tous vos projets. Pour faciliter cela, une classe peut être une extension d'une autre classe. La classe dérivée hérite alors de toutes les méthodes et variables de la classe de base (cet héritage a ça de bien que personne ne meurt pour en profiter), mais peut définir ses propres fonctions et variables, qui s'ajouteront. Une classe ne peut hériter que d'une seule autre classe, et l'héritage multiple n'est pas supporté. Les héritages se font avec le mot clé 'extends'.

```
<?php
class Caddie_nomme extends Caddie {
    var $owner;
    function set_owner ($name) {
        $this->owner = $name;
    }
}
?>
```

L'exemple ci-desssus définit la classe Caddie_nomme qui possède les même variables que la classe Caddie et la variable $owner en plus, ainsi que la fonction set_owner(). Vous créez un panier nominatif de la même manière que précédemment,

et vous pouvez alors affecter un nom au panier ou en connaître le nom. Vous pouvez de toutes les façons utiliser les même fonctions que sur un panier classique.

```
<?php
$ncart = new Caddie_nomme;    // Création d'un panier nominatif
$ncart->set_owner ("kris"); // Affectation du nom du panier
print $ncart->owner;        // Affichage du nom du panier
$ncart->add_item ("10", 1); // (héritage des fonctions de la classe père)
?>
```

`Constructor` : constructeur

> **Attention**
>
> En PHP 3 et PHP 4, les constructeurs se comportent différemment. La sémantique de PHP 4 est fortement recommandée.

Le constructeur est la fonction qui est appelée automatiquement par la classe lorsque vous créez une nouvelle instance d'une classe a l'aide de l'opérateur `new`. La fonction constructeur a le même nom que la classe. En PHP 3, une fonction devient le constructeur si elle porte le même nom que la classe. En PHP 4, une fonction devient un constructeur si elle porte le même nom que la classe dans laquelle elle est définit. La différence est subtile, mais cruciale.

```
<?php
class Auto_Caddie extends Caddie {
    function Auto_Caddie () {
        $this->add_item ("10", 1);
    }
}
// Cette syntaxe est valable en PHP 3 et 4
?>
```

L'exemple ci-dessus définit la classe Auto_Caddie qui hérite de la classe Caddie et définit le construteur de la classe. Ce dernier initialise le panier avec 1 article de type numéro 10 dès que l'instruction "new" est appelée. La fonction constructeur peut prendre ou non des paramètres optionnels, ce qui la rend beaucoup plus pratique. Pour pouvoir utiliser cette classe sans paramètre, tous les paramètres du constructeurs devraient être optionnels, en fournissant une valeur par défaut, comme ci-dessous.

```
// Cette syntaxe est valable en PHP 3 et 4
class Constructor_Cart extends Cart {
    function Constructor_Cart ($item = "10", $num = 1) {
        $this->add_item ($item, $num);
    }
}
// Création du caddie
$default_cart   = new Constructor_Cart;
// Création d'un vrai caddie
$different_cart = new Constructor_Cart ("20", 17);
```

> **Attention**
>
> En PHP 3, les classes dérivées et les constructeurs ont un certains nombre de limitations. Les exemples suivants doivent être lus avec beaucoup d'attention pour comprendre ces limitations.

```
class A {
  function A() {
    echo "Je suis le constructeur de A.<br>\n";
  }
}
class B extends A {
  function C() {
```

```
    "Je suis une fonction standard.<br>\n";
  }
}
// Aucun constructeur n'est appelé en PHP 3!!
$b = new B;
```

En PHP 3, aucun constructeur ne sera appelé dans l'exemple ci-dessus. La règle en PHP 3 est : 'Un constructeur est une fonction qui a le même nom que la classe'. Le nom de la classe est B, et il n'y a pas de fonctions qui s'appelle B() dans la classe B. Rien ne se passe.

Ceci est corrigé en PHP 4, avec l'introduction d'une nouvelle règle : Si une classe n'a pas de constructeur, le constructeur de la classe de basse est appelé, s'il existe. L'exemple ci-dessus affichera 'Je suis le constructeur de A.
' en PHP 4.

```
class A {
  function A() {
    echo "Je suis le constructeur de A.<br>\n";
  }
  function B() {
    echo "Je suis une fonction standard appelée B dans la classe A.<br>\n";
    echo "Je ne suis pas le constructeur de A.<br>\n";
  }
}
class B extends A {
  function C() {
    "Je suis une fonction standard.<br>\n";
  }
}
// Cette syntaxe va appeler B() comme constructeur.
$b = new B;
```

En PHP 3, la fonction B() de la classe A va soudaiement devenir le constructeur de la classe B, bien qu'il n'ai pas été prévu pour. La règle de PHP 3 est 'Un constructeur est une fonction qui a le même nom que la classe'. PHP 3 ne se soucie guère si la fonction est définie dans la classe B ou si elle a été héritée.

Ceci est corrigé en PHP 4, avec l'introduction d'une nouvelle règle : 'Un constructeur est une classe de même nom, définit dans la classe elle-même'. Donc, en PHP 4, la classe B n'a pas de constructeur par elle-même, et le constructeur de la classe A aura été appelé, affichant : 'Je suis le constructeur de A.
'.

Attention

Ni PHP 3 ni PHP 4 n'appelle automatiquement le constructeur de la classe supérieure depuis le constructeur de la classe dérivée. Il est de votre responsabilité de propager l'appel des constructeurs.

Note : Il n'y a pas de destructeurs en PHP 3 et PHP 4. Vous pouvez utiliser la fonction **register_shutdown_function()** à la place, pour simuler un destructeur.

Les destructeurs sont des fonctions qui sont appelées lorsqu'un objet est détruit, soit avec la fonction **unset()** soit par simple sortie d'une fonction (cas des variables locales). Il n'y a pas de destructeurs en PHP.

Opérateur : :

Attention

La documentation suivante n'est valable que pour PHP 4.

Parfois, il est pratique de faire référence aux fonctions est variables d'une classe de base, ou bien d'utiliser des méthodes de classes qui n'ont pas encore d'objets créés. L'opérateur : : est là pour ces situations.

```
<?php
class A {
```

```
  function example() {
    echo "Je suis la fonction originale A::example().<br>\n";
  }
}
class B extends A {
  function example() {
    echo "Je suis la fonction redéfinie B::example().<br>\n";
    A::example();
  }
}
// Il n'y a pas d'objets de classe A.
// L'affichage est :
//   Je suis la fonction originale A::example().<br>
A::example();
// Création d'un objet de la classe B.
$b = new B;
// L'affichage est :
//   Je suis la fonction redéfinie B::example().<br>
//   Je suis la fonction originale A::example().<br>
$b->example();
?>
```

Les exemples ci-dessus appellent la fonction example() dans la classe A, mais il n'y a pas encore d'objet de classe A, alors il n'est pas possible d'écrire $a->example(). A la place, on appelle la fonction example() comme une fonction de classe, c'est-à-dire avec le nom de la classe elle-même, et sans objet.

Il y a des fonctions de classe, mais pas de variables de classe. En fait, il n'y a aucun objet au moment de l'appel de la fonction. Donc, une fonction de classe ne peut accéder à aucune variable (mais elle peut accéder aux variables locales et globales). Il faut proscire l'utilisation de $this.

Dans l'exemple ci-dessus, la classe B redéfinit la fonction example(). La définition originale dans la classe A est remplacée par celle de B, et n'est plus accessible depuis B, à moins que vous n'appeliez spécifiquement la fonction example() de la classe A avec l'opérateur ::. Ecrivez A::example() pour cela (en fait, il faudrait écrire parent::example(), comme expliqué dans la section suivante).

Dans ce contexte, il y a un objet courant, qui peut avoir d'autres variables objets. De ce fait, lorsqu'il est utilisé depuis une méthode d'un objet, vous pouvez utiliser $this.

`parent`

Il arrive que vous ayez à écrire du code qui faire référence aux variables et fonctions des classes de base. C'est particulièrement vrai si votre classe dérivée est une spécialisation de votre classe de base.

Au lieu d'utiliser le nom litéral de votre classe de base dans votre code, vous pouvez utiliser le mot réservé `parent`, qui représente votre classe de base (celle indiqué par `extends`, dans la déclaration de votre classe). En faisant cela, vous éviter d'appeler le nom de votre classe de base directement dans votre code. Si votre héritage change, vous n'aurez plus qu'à modifier le nom de la classe dans la déclaration `extends` de votre classe.

```
<?php
class A {
  function example() {
    echo "Je suis A::example() et je fournis une fonctionnalité de base.<br>\n";
  }
}
class B extends A {
  function example() {
    echo "Je suis B::example() et je fournis une fonctionnalité supplémentaire.<br>\n";
    parent::example();
  }
}
$b = new B;
// Cette syntaxe va appeler B::example(), qui, à sont tour, va appeler A::example().
$b->example();
?>
```

Sauvegarde d’objets - cas des sessions

Note : En PHP 3, les objets perdent leur association de classe à travers le processus de sauvegarde et relecture. Le type de la variable après relecture est bien objet mais il n’a plus de méthode ou de nom de classe. Cela rend la focntionnalité plutôt inutile (l’objet est devenu un tableau avec une syntaxe étrange).

Attention

La documentation suivante n’est valable que pour PHP 4.

serialize() retourne une chaîne représentant une valeur qui peut être stockée dans les sessions de PHP, ou une base de données. **unserialize()** peut relire cette chaîne pour recréer la valeur originale. **serialize()** va sauver toutes les variables d’un objet. Le nom de la classe sera sauvé mais par les méthodes de cet objet.

Pour permettre à **unserialize()** de lire un objet, la classe de cet objet doit être définie. C’est-à-dire, si vous avez un objet $a de la classe A dans une page php1.php, et que vous le linéarisez avec **serialize()**, vous obtiendrez une chaîne qui fait référence à la classe A, et contient toutes les valeurs de $a. Pour pouvoir le relire avec la fonction **unserialize()** dans une page page2.php, il faut que la définition de la classe A soit présente dans cette deuxième page. Cela peut se faire de manière pratique en sauvant la définition de la classe A dans un fichier séparé, et en l’incluant dans les deux pages page1.php et page2.php.

```
<?php
classa.inc:
  class A {
    var $one = 1;
    function show_one() {
      echo $this->one;
    }
  }
?>
page1.php:
<?php
  include("classa.inc");
  $a = new A;
  $s = serialize($a);
  // enregistrez $s où la page2.php pourra le trouver.
  $fp = fopen("store", "w");
  fputs($fp, $s);
  fclose($fp);
?>
page2.php:
<?php
  // Ceci est nécessaire pour que unserialize() fonctionne correctement
  include("classa.inc");
  $s = implode("", @file("store"));
  unserialize($s);
  // maintenant, utilisez la méthode show_one de l’objet $a.
  $a->show_one();
?>
```

Si vous utilisez les sessions et la fonction **session_register()** pour sauver des objets, ces objets seront linéarisés automatiquement avec la fonction **serialize()** à la fin de chaque script, et relus avec **unserialize()** au début du prochain script. Cela signifie que ces objets peuvent apparaître dans n’importe quelle page qui utilise vos sessions.

Il est vivement recommandé d’inclure la définition de classe dans toutes vos pages, même si vous n’utilisez pas ces classes dans toutes vos pages. Si vous l’oubliez et qu’un tel objet est présent, il perdra sa classe, et deviendra un objet de classe `stdClass` sans aucune fonction, et donc, plutôt inutile.

Si, dans l’exemple ci-dessus, $a devient un objet de session avec l’utilisation de `session_register("a")`, vous devez pensez à inclure le fichier `classa.inc` dans toutes vos pages, et pas seulement page1.php et page2.php.

Les fonctions magiques `__sleep` et `__wakeup`

serialize() s'assure que votre classe a une méthode avec le nom magique `__sleep`. Si c'est le cas, cette fonction est appelée avant toute linéarisation. Elle peut alors nettoyer l'objet et on s'attend à ce qu'elle retourne un tableau avec la liste des noms de variables qui doivent être sauvées.

Le but de cette fonction `__sleep` est de fermer proprement toute connexion à une base de données, de valider les requêtes, de finaliser toutes les actions commencées. Cette fonction est aussi pratique si vous ave de très grands objets qui n'ont pas besoin d'être sauvé entièrement.

A l'inverse, **unserialize()** s'assure de la présence de la fonction magique `__wakeup`. Si elle existe, cette fonction reconstruit toutes les ressources d'un objet.

Le but de cette fonction `__wakeup` est de rétablit toutes les connexions aux bases de données, et de recréer les variables qui n'ont pas été sauvées.

Références dans un constructeur

Créer des références dans un constructeur peut conduire à des résultats étranges. Ce tutorial vous guide pour éviter ces problèmes.

```
<?php
class foo {
    function foo($name) {
        // crée une référence dans le tableau global $globalref
        global $globalref;
$globalref[] = &$this;
        // donne le nom de la variable
        $this->setName($name);
// et l'affiche
        $this->echoName();
    }
    function echoName() {
        echo "<br>",$this->Name;
    }
function setName($name) {
$this->Name = $name;
    }
}
?>
```

Vérifions maintenant qu'il y a une différence entre `$bar1` qui a été créé avec = et `$bar2` qui a été créé avec l'opérateur de référence =& :

```
<?php
    $bar1 = new foo('crée dans le constructeur');
    $bar1->echoName();
    $globalref[0]->echoName();
    /* affiche :
    crée dans le constructeur
    crée dans le constructeur
    crée dans le constructeur */
    $bar2 =&new foo('crée dans le constructeur');
    $bar2->echoName();
    $globalref[1]->echoName();
    /* affiche :
    crée dans le constructeur
    crée dans le constructeur
    crée dans le constructeur */
?>
```

Apparament, il n'y a pas de différence, mais en fait, il y en a une significative : `$bar1` et `$globalref[0]` ne sont pas référencées, ces deux variables sont différentes. Cela est du au fait que l'opérateur "new"ne retourne par de référence, mais retourne une copie.

> **Note :** Il n'y a aucune perte de performance (puisque PHP 4 utilise un compteur de référence) à retourner des copies au lieu de références. Au contraire, il est souvent mieux de travailler sur les copies plutôt que sur les références, car créer une référence prend un peu plus de temps que de créer une copie qui ne prend virtuellement pas de temps (à moins de créer un tableau géant ou un objet monstreux, auquel cas il est préférable de passer par des références).

Pour prouver ceci, regardez le code suivant :

```
<?php
        // maintenant, nous allons changer de nom. Qu'attendez-vous?
        // Vous pouvez vous attendre à ce que les deux variables $bar
        // et  $globalref[0] changent de nom...
        $bar1->setName('modifié');
        // comme prédit, ce n'est pas le cas
        $bar1->echoName();
        $globalref[0]->echoName();
        /* affiche :
        crée dans le constructeur
        modifié */
        // quelle est la différence entre $bar2 et $globalref[1]
        $bar2->setName('modifié');
        // Heureusement, elles sont non seulement égales, mais
        // elles représentent la même variable.
        // donc $bar2->Name et $globalref[1]->Name sont les mêmes
        $bar2->echoName();
        $globalref[1]->echoName();
        /* affiche :
        modifié
        modifié */
?>
```

Un dernier exemple pour bien comprendre.

```
<?php
class a {
    function a($i) {
        $this->value = $i;
        // Essayez de comprendre on n'a pas besoin de
        // référence ici
        $this->b = new b($this);
    }
    function createRef() {
        $this->c = new b($this);
    }
    function echoValue() {
        echo "<br>","class ",get_class($this),': ',$this->value;
    }
}
class b  {
    function b(&$a) {
        $this->a = &$a;
    }
   function echoValue() {
        echo "<br>","class ",get_class($this),': ',$this->a->value;
        }
}
// Essayez de comprendre pourquoi une copie simple va
// conduire à un résultat indésirable à
// la ligne marquée d'une étoile
$a =&new a(10);
$a->createRef();
```

```
$a->echoValue();
$a->b->echoValue();
$a->c->echoValue();
$a->value = 11;
$a->echoValue();
$a->b->echoValue(); // *
$a->c->echoValue();
/*
output:
class a: 10
class b: 10
class b: 10
class a: 11
class b: 11
class b: 11
*/
?>
```

Chapitre 14. Les références

Qu’est ce qu’une référence?

En PHP, les références sont destinées à appeler le contenu d’une variable avec un autre nom. Ce n’est pas comme en C : ici, les références sont des alias dans la table des symboles. Le nom de la variable et son contenu ont des noms différents, ce qui fait que l’on peut donner plusieurs noms au même contenu. On peut faire l’analogie avec les fichiers sous Unix, et leur nom de fichier : les noms des variables sont les entrées dans un repertoire, tandis que le contenu de la variable est le contenu même du fichier. Faire des références en PHP revient alors à faire des liens sous Unix.

Que font les références ?

Les références vous permettent de faire pointer deux variables sur le même contenu. Par exemple, lorsque vous faites :

```
<?php
$a =& $b
?>
```

cela signifie que `$a` et `$b` pointent sur la même variable.

> **Note :** `$a` et `$b` sont complètement égales ici : ce n’est pas `$a` qui pointe sur `$b`, ou vice versa. C’est bien `$a` et `$b` qui pointent sur le même contenu.

La même syntaxe peut être utilisée avec les fonctions qui retournent des références, et avec l’opérateur `new` (PHP 4.0.4 et plus récent):

```
<?php
  $bar =& new fooclass();
  $foo =& find_var ($bar);
?>
```

> **Note :** A moins d’utiliser la syntaxe ci-dessus, le résultat de `$bar = new fooclass()` ne sera pas la même variable que `$this` dans le constructeur, ce qui signifie que si vous avez utilisé la référence `$this` dans le constructeur, vous devez assigner la référence, ou bien obtenir deux objets différents.

Le deuxième intérêt des références est de pouvoir passer des variables par référence. On réalise ceci en faisant pointer des variables locales vers le contenu des variables de fonction. Exemple :

```
<?php
  function foo(&$var) {
    $var++;
  }
  $a=5;
  foo($a);
?>
```

`$a` vaut 6. Cela provient du fait que dans la fonction `foo`, la variable `$var` pointe sur le même contenu que `$a`. Voir aussi les explications détaillées dans passage par référence.

Le troisième intérêt des références est de retourner des valeurs par référence.

Ce que les références ne sont pas

Comme précisé ci-dessus, les références ne sont pas des pointeurs. Cela signifie que le script suivant ne fera pas ce à quoi on peut s’attendre :

```
<?php
  function foo(&$var) {
    $var =& $GLOBALS["baz"];
  }
```

```
  foo($bar);
?>
```

Il va se passer que `$var` dans foo() sera lié à `$bar`, mais il sera aussi relié à `$GLOBALS["baz"]`. Il n'y a pas moyen de lier `$bar` à quelque chose d'autre en utilisant le mécanisme de référence, car `$bar` n'est pas accessible dans la fonction foo() (certes, il est représenté par `$var` et `$var` possède la même valeur, mais n'est pas relié par la table des symboles).

Passage par référence

Vous pouvez passer des variables par référence, de manière à ce que la fonction modifie ses arguments. La syntaxe est la suivante :

```
<?php
  function foo(&$var) {
    $var++;
  }
  $a=5;
  foo ($a);
// $a vaut 6 maintenant
?>
```

Notez qu'il n'y a pas de signe de référence dans l'appel de la fonction, uniquement sur sa définition. La définition de la fonction est suffisante pour passer correctement des arguments par référence.

Les objets suivants peuvent être passés par référence :

- Une variable, i.e. `foo($a)`
- Un nouvel objet, i.e. `foo(new foobar())`
- Une référence, retournée par une fonction :

```
<?php
  function &bar() {
   $a = 5;
   return $a;
  }
  foo(bar());
?>
```

Voir aussi des détails dans retourner des références.

Toutes les autres expressions ne doivent pas être passées par référence, car le résultat sera indéfini. Par exemple, les passages par référence suivants sont invalides :

```
<?php
  function bar() // Notez l'absence de &
  {
        $a = 5;
        return $a;
  }
  foo(bar);
  foo($a = 5) // Expression, pas une variable
  foo(5) // Constante, pas une variable
?>
```

Ces fonctionnalités sont valables à partir de PHP 4.0.4.

Retourner des références

Retourner des références est toujours utile lorsque vous voulez utiliser une fonction pour savoir à quoi est liée une variable. Lorsque vous retournez une variable par paramètre, utilisez le code suivant

```
<?php
  function &find_var($param) {
  //    ...code...
    return $found_var;
  }
  $foo =& find_var ($bar);
  $foo->x = 2;
?>
```

Dans cet exemple, la propriété de l'objet est retournée dans `find_var` et lui sera affectée, et non pas à la copie, comme cela sera le cas avec une syntaxe par référence.

> **Note :** Contrairement au passage de paramètre, vous devez utiliser `&` aux deux endroits, à la fois pour indiquer que vous retournez par référence (pas une copie habituelle), et pour indiquer que vous assignez aussi par référence (pas la copie habituelle).

Détruire une référence

Lorsque vous détruisez une référence, vous ne faites que casser le lien entre le nom de la variable et son contenu. Cela ne signifie pas que le contenu est détruit. Par exemple,

```
<?php
$a = 1;
$b =& $a;
unset ($a);
?>
```

Cet exemple ne détruira pas `$b`, mais juste `$a`.

Encore une fois, on peut comparer cette action avec la fonction unlink d'Unix.

Repérer une référence

De nombreuses syntaxes de PHP sont implémentées via le mécanisme de référence, et tout ce qui a été vu concernant les liaisons entre variables s'applique à ces syntaxes. Par exemple, le passage et le retour d'arguments par référence. Quelques autres exemples de syntaxes :

Références globales

Lorsque vous déclarez une variable **global $var**, vous créez en fait une référence sur une variable globale. Ce qui signifie que

```
<?php
$var =& $GLOBALS["var"];
?>
```

Et que, si vous détruisez la variable `$var`, la variable globale ne sera pas détruite.

`$this`

Dans une méthode d'objet `$this` est toujours une référence sur l'objet courant.

Partie III. Caractéristiques

Chapitre 15. Gestion des erreurs

Il y a plusieurs types d'erreur et d'alerte.

Tableau 15-1. Types d'erreur PHP

Valeur	Constante	Description	Note
1	E_ERROR	Erreur fatale d'exécution	
2	E_WARNING	Alerte d'exécution (erreur non-fatale)	
4	E_PARSE	Erreur de compilation	
8	E_NOTICE	Notes d'exécution (moins critique que les alertes)	
16	E_CORE_ERROR	Erreurs qui surviennent lors de l'initialisation de PHP	PHP 4 seulement
32	E_CORE_WARNING	Alertes qui surviennent lors de l'initialisation de PHP	PHP 4 seulement
64	E_COMPILE_ERROR	Erreur fatale de compilation	PHP 4 seulement
128	E_COMPILE_WARNING	Alerte de compilation (erreur non fatale)	PHP 4 seulement
256	E_USER_ERROR	Erreur générée par l'utilisateur	PHP 4 seulement
512	E_USER_WARNING	Alerte générée par l'utilisateur	PHP 4 seulement
1024	E_USER_NOTICE	Note générée par l'utilisateur	PHP 4 seulement
	E_ALL	Toutes les erreurs ci dessus	

Les valeurs ci-dessus (numériques ou symbolique) sont utilisée pour construire un champs de bit, qui spécifie quelles erreurs rapporter. Vous pouvez utiliser les opérateurs de bits pour combiner ces valeurs et masquer uniquement celle qui vous interesse Notez que seuls, '|', '~', '!', et '&' seront utilisables dans `php.ini`, et qu'aucun opérateur ne sera utilisable dans `php3.ini`.

En PHP 4, la valeur par défaut de error_reporting est à `E_ALL & ~E_NOTICE`, ce qui signifie que toutes les erreurs et alertes seront affichées, mais pas les notes. En PHP 3, la valeur par défaut est `(E_ERROR | E_WARNING | E_PARSE)`, c'est à dire la même chose. Notez bien que ces constantes ne sont pas supportées dans le fichier `php3.ini` de PHP 3, la valeur de error_reporting doit être numériques, c'est à dire `7`.

La valeur initiale peut être modifiée dans le fichier .ini, avec la directive error_reporting, dans le fichier de configuration d'Apache `httpd.conf`, avec la directive php_error_reporting (php3_error_reporting pour PHP 3), et enfin, dans le script même, en utilisant la fonction **error_reporting()**.

Avertissement

Lorsque vous portez votre code ou vos serveurs de PHP 3 en PHP 4 vous devez vérifier les options et les appels à **error_reporting()**. Sinon, vous courrez le risque d'inactiver certains types d'erreurs et notamment E_COMPILE_ERROR. Cela peut conduire à des documents vides, sans aucun retour d'erreur.

Toutes les expressions PHP peuvent être appelée avec le préfixe "@", qui annule le rapport d'erreur pour cette expression en particulier. Si une erreur survient durant une telle expression, et que l'option de suivi des erreurs est activée, vous pourrez trouver le message d'erreur dans la variable globale, `$php_errormsg`.

Note : Le préfixe opérateur @ ne supprimera pas les messages liés aux erreurs d'analyse.

> **Avertissement**
>
> Actuellement, le préfixe @, opérateur de rapport d'erreur désactive tous les rapports, y compris les erreurs critiques qui interrompent le script. Entre autre, cela signifique que si vous utilisez @ pour supprimer des erreurs dans une fonction qui n'existe pas, ou qui a été mal orthographiée, le script sera terminé sans aucune indication.

Ci dessous, voici un exemple de gestion des erreurs avec PHP. On définit une fonction de gestion des erreurs qui enregistre les informations dans un fichier (au format XML), et email le développeur en cas d'erreur critique.

Exemple 15-1. Utiliser le contrôle d'erreur dans un script

```
<?php
// Nous effectuons nous même notre contrôle d'erreur.
error_reporting(0);
// Fonction de gestion des erreurs utilisateur
function usererrorhandler($errno, $errmsg, $filename, $linenum, $vars) {
    // timestamp pour dater l'erreur
    $dt = date("Y-m-d H:i:s (T)");
    // definit un tableau associatif avec les chaînes d'erreur
    // en realité, les seules entrées que nous considérerons
    // seront 2,8,256,512 et 1024
    $errortype = array(
                1   =>  "Erreur",
                2   =>  "Alerte",
                4   =>  "Erreur d'analyse",
                8   =>  "Note",
                16  =>  "Erreur interne",
                32  =>  "Alerte interne",
                64  =>  "Erreur de compilation",
                128 =>  "Alerte  de compilation",
                256 =>  "Erreur utilisateur",
                512 =>  "Alerte utilisateur",
                1024=>  "Note utilisateur"
                );
    // ensemble d'erreur pour lesquelles une trace sera conservée
    $user_errors = array(E_USER_ERROR, E_USER_WARNING, E_USER_NOTICE);
    $err = "<errorentry>\n";
    $err .= "\t<datetime>".$dt."</datetime>\n";
    $err .= "\t<errornum>".$errno."</errnumber>\n";
    $err .= "\t<errortype>".$errortype[$errno]."</errortype>\n";
    $err .= "\t<errormsg>".$errmsg."</errormsg>\n";
    $err .= "\t<scriptname>".$filename."</scriptname>\n";
    $err .= "\t<scriptlinenum>".$linenum."</scriptlinenum>\n";
    if (in_array($errno, $user_errors))
        $err .= "\t<vartrace>".wddx_serialize_value($vars,"Variables")."</vartrace>\n";
    $err .= "</errorentry>\n\n";
    // pour test
    // echo $err;
    // sauve l'erreur dans le fichier, et emaile moi si l'erreur est critique
    error_log($err, 3, "/usr/local/php4/error.log");
    if ($errno == E_USER_ERROR)
        mail("phpdev@mydomain.com","Critical User Error",$err);
}
function distance($vect1, $vect2) {
    if (!is_array($vect1) || !is_array($vect2)) {
        trigger_error("Paramètres incorrects : arrays attendus", E_USER_ERROR);
        return NULL;
    }
    if (count($vect1) != count($vect2)) {
        trigger_error("Les vecteurs doivent être de la même taille", E_USER_ERROR);
        return NULL;
    }
    for ($i=0; $i<count($vect1); $i++) {
        $c1 = $vect1[$i]; $c2 = $vect2[$i];
```

```
        $d = 0.0;
        if (!is_numeric($c1)) {
            trigger_error("La coordonnée $i du vecteur 1 n'est pas un nombre. Rempla-
cée par zéro",
                          E_USER_WARNING);
            $c1 = 0.0;
        }
        if (!is_numeric($c2)) {
            trigger_error("La coordonnée $i du vecteur 2 n'est pas un nombre. Rempla-
cée par zéro",
                          E_USER_WARNING);
            $c2 = 0.0;
        }
        $d += $c2*$c2 - $c1*$c1;
    }
    return sqrt($d);
}
$old_error_handler = set_error_handler("userErrorHandler");
// Constante indéfinie, génére une alerte
$t = I_AM_NOT_DEFINED;
// definition de quelques "vecteurs"
$a = array(2,3,"bla");
$b = array(5.5, 4.3, -1.6);
$c = array(1,-3);
// génère une erreur utilisateur
$t1 = distance($c,$b)."\n";
// génère une autre erreur utilisateur
$t2 = distance($b,"i am not an array")."\n";
// génère une alerte
$t3 = distance($a,$b)."\n";
?>
```

Ceci est un exemple simple, qui montre comment utiliser les fonctions de Gestions des erreurs.

Voir aussi **error_reporting()**, **error_log()**, **set_error_handler()**, **restore_error_handler()**, **trigger_error()**, **user_error()**

Chapitre 16. Création d’images

PHP n’est pas limité à la création de fichier HTML. Il peut aussi servir à créer des images GIF, PNG, JPG, wbmp et xpm, à la volée, aussi bien pour les émettre que pour les sauver. Il faut alors compiler PHP avec la librairie GD. GD et PHP requièrent aussi d’autres librairies, suivant le format d’images que vous voulez supporter. GD a cessé de supporter le format GIF depuis la version 1.6.

Exemple 16-1. Création d’images GIF avec PHP

```
<?php
    header("Content-type: image/png");
    $string=implode($argv," ");
    $im = imagecreatefrompng("images/button1.png");
    $orange = imagecolorallocate($im, 220, 210, 60);
    $px = (imagesx($im)-7.5*strlen($string))/2;
    imagestring($im,3,$px,9,$string,$orange);
    imagepng($im);
    imagedestroy($im);
?>
```

Cet exemple sera appelé depuis une page HTML avec une balise telle que: <img src="button.php3?text">. Le script ci-dessus récupère le texte de la chaîne $string et l’ajoute sur l’image de fond"images/button1.gif". Le résultat est alors envoyé au client. C’est un moyen très pratique d’éviter d’avoir à redessiner des boutons à chaque fois que le texte du bouton change. Avec ce script, il est généré dynamiquement.

Chapitre 17. Authentification HTTP avec PHP

Les fonctions d'authentification HTTP de PHP ne sont disponibles que si PHP est exécuté comme module Apache, et non pas sous la forme d'un CGI. Sous cette forme, il est possible d'utiliser la fonction **header()** pour demander une authentification ("Authentication Required") au client, générant ainsi l'apparition d'une fenêtre de demande d'utilisateur et de mot de passe. Une fois que les champs ont été remplis, l'URL sera de nouveau appelée, avec les variables `$PHP_AUTH_USER`, `$PHP_AUTH_PW` et `$PHP_AUTH_TYPE` contenant respectivement le nom d'utilisateur, le mot de passe et le type d'authentification. Actuellement, seule l'authentification simple ("Basic") est supportée. Reportez vous à la fonction **header()** pour plus d'informations.

Voici un exemple de script qui force l'authentification du client pour accéder à une page :

Exemple 17-1. Exemple d'authentication HTTP

```
<?php
  if(!isset($PHP_AUTH_USER)) {
    Header("WWW-Authenticate: Basic realm=\"My Realm\"");
    Header("HTTP/1.0 401 Unauthorized");
    echo "Texte à envoyer si le client appuie sur le bouton d'annulation\n";
    exit;
  } else {
    echo "Bonjour $PHP_AUTH_USER.<P>"
    echo "Vous avez entré le mot de passe $PHP_AUTH_PW.<P>"
  }
?>
```

Au lieu d'afficher simplement les variables globales `$PHP_AUTH_USER` et `$PHP_AUTH_PW`, vous préférerez sÛrement vérifier la validité du nom d'utilisateur et du mot de passe. Par exemple, en envoyant ces informations à une base de données, ou en recherchant dans un fichier dbm.

Méfiez vous des navigateurs buggés, tels que Internet Explorer. Ils semblent très suceptibles concernant l'ordre des entêtes. Envoyer l'entête d'authentification (*WWW-Authenticate*) avant le code de HTTP/1.0 401 semble lui convenir jusqu'à présent.

Pour éviter que quelqu'un écrive un script qui révèle les mots de passe d'une page, à la quelle on a accédé par une authentification traditionnelle, les variables globales PHP_AUTH ne seront pas assignées si l'authentification externe a été activée pour cette page. Dans ce cas, la variable `$REMOTE_USER` peut être utilisée pour identifier l'utilisateur à l'extérieur.

Notez cependant que les manipulations ci-dessus n'empêchent pas quiconque qui possède une page non authentifiée de voler les mots de passes des pages protégées, sur le même serveur.

Netscape et Internet Explorer effaceront le cache d'authentification client si ils recoivent une réponse 401. Cela permet de déconnecter un utilisateur, pour le forcer à ré-entrer son nom de compte et son mot de passe. Certains programmeurs l'utilisent pour donner un délai d'éxpiration, ou alors, fournissent un bouton de déconnexion.

Exemple 17-2. Authentification HTTP avec nom d'utilisateur/mot de passe forcé

```
<?php
  function authenticate() {
   Header( "WWW-Authenticate: Basic realm=\"Test Authentication System\"");
   Header( "HTTP/1.0 401 Unauthorized");
   echo "Vous devez entrer un nom d'utilisateur valide et un mot de passe correct pour ac-
céder à cette ressource\n";
   exit;
  }
  if(!isset($PHP_AUTH_USER) || ($SeenBefore == 1 && !strcmp($OldAuth, $PHP_AUTH_USER)) ) {
   authenticate();
  }
  else {
   echo "Bienvenue $PHP_AUTH_USER<BR>";
   echo "Old: $OldAuth";
   echo "<FORM ACTION=\"$PHP_SELF\" METHOD=POST>\n";
   echo "<INPUT TYPE=HIDDEN NAME=\"SeenBefore\" VALUE=\"1\">\n";
   echo "<INPUT TYPE=HIDDEN NAME=\"OldAuth\" VALUE=\"$PHP_AUTH_USER\">\n";
   echo "<INPUT TYPE=Submit VALUE=\"Re Authenticate\">\n"
   echo "</FORM>\n";
```

```
  }
?>
```

Ce comportement n’est pas nécessaire par le standard d’authentification HTTP Basic. Les tests avec Lynx ont montré qu’il n’affectait pas les informations de session lors de la réception d’un message de type 401, ce qui fait que passer ces informations entre le serveur et le client, et donnera l’accès à la ressource. Cependant, l’utilisateur peut utiliser la touche ’_’ pour détruire les anciennes autentifications.

Notez aussi que tout ceci ne fonctionne pas sous Microsoft IIS et que les limitations de PHP en version CGI sont dues aux limitations de IIS.

Chapitre 18. Cookies

PHP supporte les cookies de manière transparente. Les cookies sont un mécanisme d'enregistrement d'informations sur le client, et de lecture de ces informations. Ce système permet d'authentifier et de suivre les visiteurs. Vous pouvez envoyer un cookie avec la commande **setcookie()**. Les cookies font partie de l'entête HTTP, ce qui impose que **setcookie()** soit appelé avant tout affichage sur le client. Ce sont les mêmes limitations que pour **header()**.

Tous les cookies qui sont envoyés au client seront automatiquement retournés au script PHP, et transformés en variable, exactement comme pour GET et POST. Si vous souhaitez affecter plusieurs valeurs à un seul cookie, ajoutez*[]* au nom du cookie. Pour plus details, reportez vous à la fonction **setcookie()**.

Chapitre 19. Gestion des chargements de fichier

Chargements de fichiers par méthode POST

PHP est capable de recevoir des fichiers émis par un navigateur conforme à la norme RFC-1867 (c'est-à-dire Netscape Navigator 3 ou supérieur, Microsoft Internet Explorer 3 avec un patch de Microsoft, ou supérieur sans le patch). Cette fonctionnalité permet de charger des fichiers textes ou binaires. Avec l'authentification et les fonctions de manipulation des fichiers, vous avez un contrôle total sur le chargement et la gestion des fichiers chargés.

Notez bien que PHP supporte aussi le chargement par la méthode PUT comme dans le navigateur Netscape Composer et les clients Amaya du W3C. Reportez-vous au chapitre sur le support de la méthode PUT.

Un écran de chargement de fichiers peut être constitué en créant un formulaire de la manière suivante :

Exemple 19-1. Formulaire de chargement de fichier

```
<FORM ENCTYPE="multipart/form-data" ACTION="_URL_" METHOD="POST">
<INPUT TYPE="hidden" name="MAX_FILE_SIZE" value="1000">
Envoyez ce fichier : <INPUT NAME="userfile" TYPE="file">
<INPUT TYPE="submit" VALUE="Send File">
</FORM>
```

Le paramètre _URL_ doit pointer sur un fichier PHP. L'option MAX_FILE_SIZE cachée doit précéder le nom du fichier à charger, et représente la taille maximale du fichier à charger. La valeur est donnée en octets. Dans ce script, les valeurs suivantes doivent être définies pour assurer un chargement correct :

En PHP 3, les variables suivantes seront définies dans le script de destination, en cas de téléchargement réussi, et en supposant que register_globals est activé dans le fichier `php.ini`. Si track_vars est activé, elles seront aussi disponibles dans le dossier `$HTTP_POST_VARS`. Notez que les noms des variables suivantes supposent que nom du fichier téléchargé est 'userfile', comme présenté dans l'exemple ci-dessus.

- `$userfile` - Le nom temporaire du fichier qui sera chargé sur la machine serveur.
- `$userfile_name` - Le nom du fichier original sur le système de l'envoyeur.
- `$userfile_size` - La taille du fichier envoyé en octets.
- `$userfile_type` - Le type MIME du fichier, si le navigateur a fourni cette information. Par exemple, `"image/gif"`.

Notez que "$userfile" prend la valeur qui est passée dans le champs INPUT de type TYPE=file. Dans l'exemple ci-dessus, nous avons choisi de l'appeler "userfile".

En PHP 4, le comportement est légèrement différent, car c'est la variable d'environnement `$HTTP_POST_FILES`, qui contiendra les informations sur les fichiers téléchargés. Ces informations sont disponibles dans si l'option track_vars est activée, mais track_vars est toujours activée dans les versions de PHP supérieures à la version 4.0.2.

Le contenu du tableau `$HTTP_POST_FILES` décrit ci-dessous. Notez que l'on suppose ici que le nom du fichier téléchargé est '`userfile`', comme présenté dans l'exemple ci-dessus :

`$HTTP_POST_FILES['userfile']['name']`

Le nom du fichier original sur la machine source.

`$HTTP_POST_FILES['userfile']['type']`

Le type MIME du fichier, si le navigateur a fourni cette information. Par exemple, `"image/gif"`.

`$HTTP_POST_FILES['userfile']['size']`

La taille du fichier envoyé, en octets.

`$HTTP_POST_FILES['userfile']['tmp_name']`

Le nom temporaire du fichier qui sera chargé sur la machine serveur.

Les fichiers seront enregistrés par défaut dans le dossier des fichiers temporaires, à moins qu'un autre dossier n'ait été fourni avec la directive de configuration upload_tmp_dir du fichier `php.ini`. Le dossier par défaut du serveur peut être modifié grâce à la variable d'environnement TMPDIR, de l'utilisateur qui exécute PHP. Sa modification avec **putenv()** depuis un script PHP ne fonctionnera pas. Cette variable d'environnement peut aussi être utilisée pour s'assurer que d'autres opérations fonctionnent avec les fichiers téléchargés.

Exemple 19-2. Validation de fichiers téléchargés

Les exemples suivants fonctionnent sur les versions de PHP 3 supérieures à la version 3.0.16, et supérieures à la version 4.0.2 pour PHP 4. Reportez-vous à la section des fonctions pour étudier **is_uploaded_file()** et **move_uploaded_file()**.

```
<?;php
if (is_uploaded_file($userfile)) {
    copy($userfile, "/dossier/des/fichiers/telecharges/");
} else {
    echo "Attaque potentielle par fichier téléchargé : fichier ’$userfile’.";
}
/* ...ou... */
move_uploaded_file($userfile, "/dossier/des/fichiers/telecharges");
?>
```

Pour les versions plus anciennes de PHP, vous devrez faire quelque chose comme ceci :

```
<?;php
/* Test du fichier téléchargé. */
function is_uploaded_file($filename) {
    if (!$tmp_file = get_cfg_var(’upload_tmp_dir’)) {
        $tmp_file = dirname(tempnam(”, ”));
    }
    $tmp_file .= ’/’ . basename($filename);
    /* L’utilisateur peut avoir un slash final dans php.ini... */
    return (ereg_replace(’/+’, ’/’, $tmp_file) == $filename);
}
if (is_uploaded_file($userfile)) {
    copy($userfile, "/place/to/put/uploaded/file");
} else {
    echo "Attaque potentielle par fichier téléchargé : fichier ’$userfile’.";
}
?>
```

> **Note :** Cela ne fonctionnera *PAS* avec les versions de PHP 4 supérieure à 4.0.2. Cela repose sur des fonctionnalités internes à PHP qui ont évolué après cette version.

Le script PHP qui reçoit le fichier chargé doit pouvoir gérer le fichier de manière appropriée. Vous pouvez utiliser la variable `$file_size` pour recaler tous les fichiers qui sont trop gros ou trop petits. Vous pouvez utiliser la variable `$file_type` pour recaler les fichiers qui n’ont pas le bon type. Quelques soient les actions, ce script doit pouvoir supprimer le fichier du dossier temporaire, ou le déplacer ailleurs.

Le fichier sera automatiquement effacé du fichier temporaire à la fin du script, s’il n’a pas été déplacé ou renommé.

Erreurs classiques

La variable MAX_FILE_SIZE ne peut pas spécifier une taille de fichier plus grande que la taille qui a été fixée par upload_max_filesize, dans le fichier `php3.ini`, ou par php3_upload_max_filesize dans les directives Apache. La valeur par défaut est 2 mégaoctets.

Ne pas valider les fichiers que vous manipulez peut donner l’accès aux utilisateurs à des fichiers sensibles dans d’autres dossiers!

Attention : il semble que CERN httpd supprime tout ce qui est après le premier caractère dans l’en-tête MIME. Tant que c’est le cas, CERN httpd ne pourra pas effectuer de chargement.

Chargement multiples de fichiers

Il est possible de charger plusieurs fichiers en même temps, et de recevoir les informations adéquates organisées sous forme de tableau. Pour ce faire, il faut utiliser la même syntaxe d'envoi dans le code HTML que pour les sélections ou boîtes à cocher multiples.

Note : Le support du chargement multiple de fichier a été ajouté dans la version 3.0.10.

Exemple 19-3. Chargement multiple de fichier

```
<form action="file-upload.html" method="post" enctype="multipart/form-data">
  Send these files:<br>
  <input name="userfile[]" type="file"><br>
  <input name="userfile[]" type="file"><br>
  <input type="submit" value="Send files">
</form>
```

Lorsque le formulaire ci-dessus a étéF envoyé, les tableaux `$userfile`, `$userfile_name`, et `$userfile_size` seront initialisés (ainsi que $HTTP_POST_VARS). Chaque tableau sera de type numérique, et contiendra les valeurs appropriées pour le chargement des fichiers.

Par exemple, supposons que les noms de fichier `/home/test/review.html` et `/home/test/xwp.out` soient envoyés. Dans ce cas, `$userfile_name[0]` va contenir `review.html`, et `$userfile_name[1]` contiendra `xwp.out`. Similairement, `$userfile_size[0]` contiendra la taille de `review.html`, etc...

`$userfile['name'][0]`, `$userfile['tmp_name'][0]`, `$userfile['size'][0]`, et `$userfile['type'][0]` sont aussi affectés.

Chargement par méthode PUT

PHP supporte la méthode HTTP PUT utilisée par les navigateurs tels que Netscape Composer et W3C Amaya. Les requêtes de type PUT sont beaucoup plus simples que les chargements de fichiers, et elles ressemblent à :

```
PUT /path/filename.html HTTP/1.1
```

Normalement, cela signifie que le client distant va sauver les données qui suivent dans le fichier: `/path/filename.html` de votre disque. Ce n'est évidemment pas très sécurisé de laisser Apache ou PHP écraser n'importe quel fichier de l'arborescence. Pour éviter ceci, il faut d'abord dire au serveur que vous voulez qu'un script PHP donné gère la requête. Avec Apache, il y a une directive pour cela : *Script*. Elle peut être placée n'importe où dans le fichier de configuration d'Apache. En général, les webmestres la place dans le bloc <Directory>, ou peut être dans le bloc <Virtualhost>. La ligne suivante fera très bien l'affaire :

```
Script PUT /put.php3
```

Elle indique à Apache qu'il doit envoyer les requêtes de chargement par méthode PUT au script put.php3. Bien entendu, cela présuppose que vous avez activé PHP pour qu'il prenne en charge les fichiers de type .php3, et que PHP est actif.

Dans le fichier put.php3 file vous pouvez mettre ceci :

```
<?php
 copy($PHP_UPLOADED_FILE_NAME,$DOCUMENT_ROOT.$REQUEST_URI);
?>
```

Ce script va copier le fichier chargé par le client distant à l'endroit désiré. Vous aurez probablement à effectuer quelques tests et des authentifications d'utilisateur, avant d'effectuer cette copie. Le seul piège est que lorsque PHP reçoit un chargement par méthode PUT, il va enregistrer le fichier dans le dossier temporaire, tout comme avec la méthode POST-method. A la fin de la requête, le fichier sera effacé. Ce qui fait que ce script doit placer le fichier chargé quelque part. Le nom du fichier temporaire est placé dans la variable globale $PHP_PUT_FILENAME, et la destination prévue est placée dans `$REQUEST_URI` (ces noms peuvent changer d'une configuration d'Apache à l'autre). Cette destination est celle qui est demandée par le client, et vous n'avez pas à obéir aveuglément au client. Vous pourriez par exemple, déplacer le fichier dans un dossier de chargement.

Chapitre 20. Utilisation des fichiers à distance

Aussi longtemps que le support de la fonction d'ouverture générique de fichiers ("URL fopen wrapper") est actif lorsque vous configurez PHP (il est inutile de passer explicitement l'option `--disable-url-fopen-wrapper` pour faire la configuration), vous pouvez utiliser des URLs (HTTP et FTP) avec la plupart des fonctions qui utilisent un nom de fichier comme paramètre, ceci incluant les expressions **require()** et **include()**.

> **Note :** Vous ne pouvez pas utiliser les fichiers distants dans les expressions **include()** et **require()** avec Windows.

Par exemple, vous pouvez suivre l'exemple suivant pour ouvrir un fichier sur un serveur web distant, analyser les résultats pour extraire les informations dont vous avez besoin, et ensuite l'utiliser dans une requête de base de données, ou simplement éditer les informations dans le style de votre site.

Exemple 20-1. Connaître le titre d'une page distante

```
<?;php
  $file = fopen("http://www.php.net/", "r");
  if (!$file) {
    echo "<p>Impossible d'ouvrir le fichier distant.\n";
    exit;
  }
  while (!feof($file)) {
    $line = fgets($file, 1024);
    /* Cela ne fonctionne que site le titre est écrit sur une ligne.*/
    if (eregi("<title>(.*)</title>", $line, $out)) {
      $title = $out[1];
      break;
    }
  }
  fclose($file);
?>
```

Vous pouvez aussi écrire des fichiers sur un serveur FTP aussi longtemps que vous êtes connecté avec un utilisateur ayant les bons droits d'accès, alors que le fichier n'existait pas encore. Pour vous connecter avec un utilisateur autre qu'anonyme, vous devez spécifier un nom d'utilisateur (et certainement le mot de passe) dans l'URL, comme par exemple 'ftp://user:password@ftp.example.com/path/to/file'. (Vous pouvez utiliser le même type de syntaxe pour accéder aux fichiers via HTTP lorsqu'ils nécessitent une authentification basique.)

Exemple 20-2. Stocker des données sur un serveur distant

```
<?;php
  $file = fopen("ftp://ftp.php.net/incoming/outputfile", "w");
  if (!$file) {
    echo "<p>Impossible d'ouvrir un fichier distant en écriture.\n";
    exit;
  }
  /* Ecriture des données. */
  fputs($file, "$HTTP_USER_AGENT\n");
  fclose($file);
?>
```

> **Note :** Remarque: Vous pouvez avoir l'idée,à partir de l'exemple ci-dessus, d'utiliser la même technique pour écrire sur un log distant, mais comme mentionné ci-dessus vous ne pouvez qu'écrire sur un nouveau fichier en utilisant les fonctions **fopen()** avec une URL. Pour faire des log distribués, nous vous conseillons de regarder la partie **syslog()**.

Chapitre 21. Gestion des connexions

Note : Les informations suivantes ne sont valables qu'à partir de la version 3.0.7.

Le statut des connexions est conservé en interne par PHP. Il y a trois états possibles :

- 0 - NORMAL (normal)
- 1 - ABORTED (annulé)
- 2 - TIMEOUT (périmé)

Lorsqu'un script PHP est en cours d'exécution, son état est NORMAL. Si le client distant se déconnecte, le statut devient ABORTED. En général, une telle déconnexion provient d'un arrêt temporaire. Si la durée maximale d'exécution de PHP est dépassée, (voir **set_time_limit()**), le script prend le statut TIMEOUT.

Vous pouvez en outre, décider si vous voulez que la déconnexion d'un client provoque l'arrêt de votre script. Il est parfois pratique de terminer le script, même si le client n'est plus là pour recevoir les informations. Cependant, par défaut, le script sera interrompu, et terminé dès que le client se déconnecte. Ce comportement peut être modifié avec la directive ignore_user_abort dans le fichier `php.ini` ou bien avec la directive Apache *`ignore_user_abort`* du fichier Apache `httpd.conf` ou avec la fonction **ignore_user_abort()**. Si vous ne demandez pas à PHP d'ignorer la déconnexion, et que l'utilisateur se déconnecte, le script sera terminé. La seule exception est si vous avez enregistré une fonction de fermeture, avec **register_shutdown_function()**. Avec une telle fonction, lorsque l'utilisateur interromp sa requête, à la prochaîne exécution du script, PHP va s'apercevoir que le dernier script n'a pas été terminé, et il va déclencher la fonction de fermeture. Cette fonction sera aussi appelée à la fin du script, si celui-ci se termine normalement. Pour pouvoir avoir un comportement différent suivant l'état du script lors de sa finalisation, vous pouvez exécutez des commandes spécifiques à la déconnexion grâce à la commande **connection_aborted()**. Cette fonction retournera `TRUE` si la connexion a été annulée.

Votre script peut aussi éxpirer après un laps de temps. Par défaut, le délai est de 30 secondes. Cette valeur peut être changée en utilisant la directive PHP *`max_execution_time`* dans le fichier `php.ini` ou avec la directive *`php3_max_execution_time`*, dans le fichier Apache `.conf` ou encore avec la fonction **set_time_limit()**. Lorsque le délai éxpire, le script est terminé, et comme pour la déconnexion du client, une fonction de finalisation sera appelée. Dans cette fonction, vous pouvez savoir si c'est le délai d'éxpiration qui a causé la fin du script, en appelant la fonction **connection_timeout()**. Cette fonction retournera vrai si le délai d'éxpiration a été dépassé.

Une chose à noter et que les deux cas ABORTED et TIMEOUT peuvent être appelés en même temps. Ceci est possible si vous demandez à PHP d'ignorer les déconnexions des utilisateurs. PHP va quand même noter le fait que l'utilisateur s'est déconnecté, mais le script va continuer. Puis, lorsqu'il atteint la limite de temps, le script va éxpirer. A ce moment là, les deux fonctions **connection_timeout()** et **connection_aborted()** vont retourner `TRUE`. Vous pouvez aussi vérifier les deux états en un seul appel avec la fonction **connection_status()**. Cette fonction va retourner un champs de bits, avec les états. Si les deux états sont actifs, l'état retourné prendra la valeur 3.

Chapitre 22. Connexions persistantes aux bases de données

Les connexions persistantes aux bases de données SQL sont des connexions qui ne se referment pas à la fin du script. Lorsqu'une connexion persistante est demandée, PHP s'assure qu'il n'y a pas une autre connexion identique (qui serait ouverte précédemment, avec le même nom d'hôte, d'utilisateur et le même mot de passe), et si une telle connexion existe, elle est utilisée. Sinon, elle est créée. Une connexion identique est une connexion qui a ouvert le même hôte, avec le même nom et même mot de passe (si ils sont nécessaires).

Ceux qui ne sont pas rompus aux techniques des serveurs web et leur distribution de la charge de travail, se font parfois une fausse idée de ces connexions persistantes. En particulier, les connexions persistantes ne permettent pas l'ouverture de plusieurs sessions avec le même lien, ne permettent pas la réalisation de transactions efficaces et ne font pas le café. En fait, pour être extrêmement clair sur le sujet, les connexions persistantes ne vous donnent aucune fonctionnalité de plus que les connexions non persistantes.

Alors pourquoi?

Cela s'explique par la manière avec laquelle les serveurs web fonctionnent. Il y a trois manières d'utiliser PHP pour générer des pages.

La première est d'utiliser PHP comme un CGI (Common Interface Gateway). Lorsque PHP fonctionne de cette manière, une instance de l'interpréteur PHP est créée puis détruit pour chaque page demandée. Etant donné qu'il est détruit après chaque requête, toutes les ressources acquises (comme une connexion à une base SQL), sont purement et simplement détruites.

La deuxième méthode, et de loin, la plus prisée, est d'exécuter PHP sous la forme d'un module sur un serveur multi-process, ce qui revient à dire : Apache. Un tel serveur a typiquement un processus parent qui coordonne un ensemble de processus fils, qui servent les fichiers. Lorsque les requêtes parviennent depuis un client, elles sont transmises à un fils disponible. Cela signifie que si un client fait une deuxième requête, il peut être servi par un processus client différent du premier. Les connexions persistantes vous permettent alors de ne vous connecter à une base SQL que la première fois. Lors des connexions ultérieures, les processus fils pourront réutiliser la connexion ouverte précédemment.

La dernière méthode est d'utiliser PHP sous la forme d'un module de serveur multi-threads. Actuellement, PHP 4 support ISAPI, WSAPI, et NSAPI (sous Windows), qui permettent tous d'utiliser PHP comme un module sur un serveur multi-thread tel que Netscape FastTrack, Microsoft's Internet Information Server (IIS), et O'Reilly's WebSite Pro. Le comportement est essentiellement le même que pour les serveurs multi-process décrit précédemment. Notez que SAPI n'est pas possible avec PHP 3.

Si les connexions persistantes n'ont aucune fonctionnalité de plus, à quoi servent-elles?

La réponse est extrêmement simple : efficacité. Les connexions persistantes sont un bon moyen d'accélérer les accès à une base SQL si le traitement de connexion à la base est long. Ce temps dépend de nombreux facteurs : le type de base de données, cette base est-elle sur le même serveur ou pas, quelle est la charge du serveur de base de données, etc... Si le temps de connexion est long, les connexions persistantes seront bien utiles, car une fois ouverte par un processus fils, la connexion est réutilisable sans avoir à se reconnecter. Si vous avez 20 processus fils, il suffit d'avoir 20 connexions persistantes ouvertes, une par fils.

Notez que les connexions persistantes ont quelques inconvénients si vous hébergez une base de données, dont le nombre maximal de connexion risque d'être atteint par les connexions persistantes. Si votre base de données accepte jusqu'à 16 connections simultanées et que, 17 processus essaient de se connecte, le dernier restera sur la touche. Si il y a des erreurs dans les scripts qui ne permettent pas de fermer la connexion (par exemple, une boucle infinie), votre serveur sera rapidement engorgé. Vérifier la documentation de votre base de données pour savoir comment elle traite les connexions inactives ou abandonnées.

Résumons nous : les connexions persistantes ont été définies pour avoir les mêmes fonctionnalités que les connexions non persistantes. Les deux types de connexions sont parfaitement interchangeables, et n'affecteront pas le comportement de votre script : uniquement son efficacité.

Chapitre 23. Safe mode

Le "Safe Mode" est le mode de sécurité de PHP : une solution au problème de partage de PHP sur un serveur. Ce système pêche au niveau de l'architecture car il n'est pas correct de tenter de résoudre ce problème au niveau de PHP, mais les solutions alternatives basées sur le serveur web et l'OS ne sont pas réalistes. De nombreux acteurs, notamment les fournisseurs d'hébergement, utilise le "Safe Mode".

Les directives de configuration qui contrôlent le safe mode sont :

```
safe_mode = Off
open_basedir =
safe_mode_exec_dir =
safe_mode_allowed_env_vars = PHP_
safe_mode_protected_env_vars = LD_LIBRARY_PATH
disable_functions =
```

Lorsque safe mod est actif, PHP vérifie que le propriétaire du script courant est le même que le propriétaire de fichier qui seront manipulé par ce script. Par exemple, si on a la situation suivante :

```
-rw-rw-r--    1 rasmus   rasmus       33 Jul  1 19:20 script.php
-rw-r--r-    1 root     root       1116 May 26 18:01 /etc/passwd
```

Exécuter le script `script.php`

```
<?php
 readfile('/etc/passwd');
?>
```

générera cette erreur, si le safe mode est activé :

```
Warning: SAFE MODE Restriction in effect. The script whose uid is 500 is not
allowed to access /etc/passwd owned by uid 0 in /docroot/script.php on line 2
```

Si vou utilisez la directive open_basedir au lieu du safe mode, alors les manipulations seront limitées aux fichiers situés dans les dossiers spécifiés. Par exemple :

```
<Directory /docroot>
php_admin_value open_basedir /docroot
</Directory>
```

Si vous exécutez le script `script.php` ci-dessus avec la configuration d'open_basedir le résultat sera l'affichage suivant :

```
Warning: open_basedir restriction in effect. File is in wrong directory in
/docroot/script.php on line 2
```

Vous pouvez aussi désactiver individuellement les fonctions. Par exemple, en ajoutant cette ligne dans le fichier `php.ini` :

```
disable_functions readfile,system
```

toute utilisation des fonctions **readfile()** et **system()** générera l'affichage suivant :

```
Warning: readfile() has been disabled for security reasons in
/docroot/script.php on line 2
```

Partie IV. Index des fonctions

I. Apache

ascii2ebcdic (PHP 3>= 3.0.17)

Transforme une chaîne ASCII en EBCDIC

```
int ascii2ebcdic (string ascii_str)
```

ascii2ebcdic() est une fonction spéficique à Apache, qui n'est disponible que sur les OS qui gèrent le format EBCDIC (OS/390, BS2000). Elle traduit la chaîne ASCII *ascii_str* en son équivalent EBCDIC (avec protection des données binaires) et retourne le résultat.

Voir aussi **ebcdic2ascii()**

ebcdic2ascii (PHP 3>= 3.0.17)

Transforme une chaîne EBCDIC en ASCII

```
int ebcdic2ascii (string ebcdic_str)
```

ebcdic2ascii() est une fonction spéficique à Apache, qui n'est disponible que sur les OS qui gèrent le format EBCDIC (OS/390, BS2000). Elle traduit la chaîne EBCDIC *ebcdic_str* en son équivalent ASCII (avec protection des données binaires) et retourne le résultat.

Voir aussi **ascii2ebcdic()**

apache_lookup_uri (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Effectue une requête partielle pour l'URI spécifiée et renvoie toutes les informations.

```
class apache_lookup_uri (string filename)
```

apache_lookup_uri() effectue une requête partielle pour l'URI spécifiée. Cette requête permet de récupérer toutes les informations importantes à propos de la ressource concernée. Les propriétés de la classe renvoyée sont les suivantes :
status
the_request
status_line
method
content_type
handler
uri
filename
path_info
args
boundary
no_cache
no_local_copy
allowed
send_bodyct
bytes_sent
byterange
clength
unparsed_uri
mtime
request_time

Note : apache_lookup_uri() ne fonctionne que lorsque le PHP est installé sous la forme d'un module Apache.

apache_note (PHP 3>= 3.0.2, PHP 4 >= 4.0b1)

Affiche ou affecte le paramètre "apache request notes".

```
string apache_note (string note_name, string [note_value])
```

apache_note() est une fonction spécifique au serveur Apache. Cette fonction affecte ou renvoie la valeur de la variable contenue dans la table `notes` d'Apache. Si la fonction est appelée avec un argument, elle renvoie la valeur courante de la variable `note_name`. Si **apache_note()** est appelée avec deux arguments, **apache_note()** affecte à la note `note_value` la valeur `note_name` et **apache_note()** retournera la valeur précédente de la variable `note_name`.

getallheaders (PHP 3, PHP 4 >= 4.0b1)

Récupère toutes les en-têtes des requêtes HTTP.

```
array getallheaders ()
```

getallheaders() renvoie un tableau associatif de toutes les en-têtes HTTP de la requête courante.

> **Note :** Vous pouvez récupérer la valeur d'une variable d'une CGI en la lisant à partir des variables d'environnement, ce qui fonctionne aussi bien dans le cas d'une installation en module ou d'une installation en CGI. Utilisez la fonction **phpinfo()** pour avoir une liste de toutes les variables d'environnement disponibles.

Exemple 1. Exemple avec getallheaders()

```
<?php
$headers = getallheaders();
while (list($entete, $valeur) = each($headers)) {
    echo "$entete: $valeur<br>\n";
}
?>
```

Cet exemple est un exemple d'affichage de toutes les en-têtes de la requête courante.

> **Note : getallheaders()** ne fonctionne que si PHP est installé comme module Apache.

virtual (PHP 3, PHP 4 >= 4.0b1)

Effectue une sous-requête Apache

```
int virtual (string filename)
```

virtual() est une fonction spécifique au serveur Apache. Elle est équivalente à la directive "`<!-#include virtual...-->`" lorsque vous utilisez le module include d'Apache. Cette fonction effectue une sous-requête Apache. C'est très utile lorsque vous utilisez des scripts CGI, des fichiers .shtml ou n'importe quel type de fichier qui doit être analysé par le serveur Apache. Il est à noter que lors de l'utilisation avec des scripts CGI, ces derniers doivent générer une en-tête valide, c'est-à-dire, au minimum une en-tête "Content-Type". Pour les fichiers PHP, il est conseillé d'utiliser les fonctions **include()** et **require()**. **virtual()** ne peut pas être utilisé pour inclure un fichier qui est lui-même un fichier PHP.

II. Tableaux

Ces fonctions vous permettent de manipuler et de traiter les tableaux de nombreuses façons. Les tableaux sont très efficaces dès qu'il s'agit de stocker, gérer et traiter des données en groupe.

Les tableaux simples et multi-dimensionnels sont supportés et peuvent être créés par l'utilisateur, ou par une fonction. Il y a des fonctions spécifiques qui remplissent des tableaux à partir de résultats de requêtes, et de nombreuses fonctions retournent un tableau.

Voir aussi **is_array()**, **explode()**, **implode()**, **split()** et **join()**.

array (unknown)

Crée un tableau

```
array array ([mixed ...])
```

array() retourne un tableau créé avec les paramètres passés. On peut attribuer un index particulier à une valeur avec l'opérateur =?>.

> **Note : array()** est un élément de langage utilisé pour représenter des tableaux litéraux, et non pas une fonction au sens strict du terme.

La syntaxe "index => valeur", séparée par des virgules, définit les index et leur valeur. Un index peut être une chaîne ou un nombre. Si l'index est omis, un index numérique sera automatiquement généré (commençant à 0). Si l'index est un entier, le prochain index généré prendra la valeur d'index la plus grande + 1. Notez que si deux index identiques sont définis, le dernier remplacera le premier.

L'exemple suivant montre comment créer un tableau à deux dimensions, comment spécifier les index d'un tableau associatif, et comment générer automatiquement des index numériques.

Exemple 1. Exemple avec array()

```
<?php
  $fruits = array (
    "fruits"  => array ("a" => "orange", "b" => "banane", "c" => "pomme"),
    "numbres" => array (1, 2, 3, 4, 5, 6),
    "trous"   => array ("premier", 5 => "deuxième", "troisième")
  );
?>
```

Exemple 2. Index automatique d'un tableau avec array()

```
<?php
  $array = array( 1, 1, 1, 1,  1, 8=>1,  4=>1, 19, 3=>13);
  print_r($array);
?>
```

qui affichera :

```
Array
(
    [0] => 1
    [1] => 1
    [2] => 1
    [3] => 13
    [4] => 1
    [8] => 1
    [9] => 19
)
```

Notez bien que l'index '3' est défini deux fois, et conserve finalement sa dernière valeur de 13. L'index '4' est défini après l'index '8', et l'index généré suivant (valeur 19) est 9, puisque le plus grand index est alors 8.

Cet exemple crée un tableau dont les index commencent à 1.

Exemple 3. Tableau d'index commençant à 1

```
<?php
  $firstquarter  = array(1 => 'Janvier', 'Février', 'Mars');
  print_r($firstquarter);
?>
```

qui affichera :

```
Array
(
[1] => 'Janvier'
[2] => 'Février'
[3] => 'Mars'
)
```

Voir aussi **list()**.

array_count_values (PHP 4 >= 4.0b4)

Compte le nombre de valeurs dans un tableau

```
array array_count_values (array input)
```

array_count_values() retourne un tableau contenant les valeurs du tableau *input* comme clés et leurs fréquence comme valeur.

Exemple 1. Exemple avec array_count_values()

```
<?php
  $array = array(1, "bonjour", 1, "monde", "bonjour");
  array_count_values($array);
// retourne array(1=>2, "bonjour"=>2, "monde"=>1)
?>
```

Note : array_count_values() a été ajoutée en PHP 4.0.

array_diff (PHP 4 >= 4.0.1)

Calcule la différence entre deux tableaux

```
array array_diff (array array1, array array2 [, array ...])
```

array_diff() retourne un tableau qui contient toutes les valeurs du tableau *array1* qui sont absentes de tous les autres arguments. Notez que les clés sont préservées.

Exemple 1. Exemple avec array_diff()

```
<?php
  $array1 = array ("a" => "vert", "rouge", "bleu", "rouge");
  $array2 = array ("b" => "vert", "jaune", "rouge");
  $result = array_diff ($array1, $array2);
?>
```

`$result` contient `array("bleu");`. Les valeurs multiples dans $array1 seront toutes traitées de la même façon.

Voir aussi **array_intersect()**.

array_filter (PHP 4 >= 4.0.6)

Filtre les éléments d'un tableau

```
array array_filter (array input [, mixed callback])
```

array_filter() retourne un tableau contenant les éléments du tableau *`input`*, filtrés grâce à la fonction *`callback`*. Si *`input`* est un tableau associatif, les clés sont préservées.

Exemple 1. Exemple avec array_filter()

```
<?php
  function pair($var) {
   return ($var % 2 == 1);
  }
  function impair($var) {
   return ($var % 2 == 0);
  }
  $array1 = array("a"=>1, "b"=>2, "c"=>3, "d"=>4, "e"=>5);
  $array2 = array(6, 7, 8, 9, 10, 11, 12);
  $tableau_pair = array_filter($array1, "pair");
  $tableau_impair = array_filter($array2, "impair");
?>
```

Cet exemple montre comment extraire les nombres pairs dans `$tableau_impair` : ce dernier contient `array ("a"=>1, "c"=>3, "e"=>5);`, et les nombres impairs dans `$tableau_pair` : ce dernier contient `array (6, 8, 10, 12);`,

Voir aussi **array_map()** et **array_reduce()**.

array_flip (PHP 4 >= 4.0b4)

Remplace les clés par les valeurs, et les valeurs par les clés

```
array array_flip (array trans)
```

array_flip() retourne un tableau dont les clés sont les valeurs du précédent tableau, et les valeurs sont les clés. **array_flip()** ne fonctionne que sur des entiers et des chaînes, et affichera une erreur s'il détecte une clé ou une valeur de type invalide (tableau, objet, booléen, nombre à virgule flottante).

Si une valeur n'est pas unique, seule la dernière clé sera utilisée comme valeur, et toutes les autres seront perdues.

array_flip() retourne `FALSE` en cas d'échec.

Exemple 1. Exemple avec array_flip()

```
<?php
  $trans = array_flip ($trans);
  $original = strtr ($str, $trans);
?>
```

Exemple 2. array_flip() example : collision

```
<?php
  $trans = array ("a" => 1, "b" => 1, "c" => 2);
  $trans = array_flip ($trans);
// et $trans vaut : array(1 => "b", 2 => "c");
?>
```

Note : array_flip() a été ajoutée en PHP 4.0.

array_intersect (PHP 4 >= 4.0.1)

Calcule l'intersection de tableaux

```
array array_intersect (array array1, array array2 [, array ...])
```

array_intersect() retourne un tableau contenant toutes les valeurs de *array1* qui sont présentes dans tous les autres arguments. Notez que les clés sont préservées.

Exemple 1. Exemple avec array_intersect()

```
<?php
  $array1 = array ("a" => "vert", "rouge", "bleu");
  $array2 = array ("b" => "vert", "jaune", "rouge");
  $result = array_intersect ($array1, $array2);
?>
```

`$result` contient `array ("a" => "vert", "rouge");`.

Voir aussi **array_diff()**.

array_keys (PHP 4 >= 4.0b1)

Retourne toutes les clés d'un tableau

```
array array_keys (array input, mixed [search_value])
```

array_keys() retourne les clés numériques et littérales du tableau *input*.

Si l'option *search_value* est spécifiée, seules les clés ayant cette valeur seront retournées. Sinon, toutes les clés de *input* sont retournées.

Exemple 1. Exemple avec array_keys()

```
<?php
  $array = array(0 => 100, "couleur" => "rouge");
  array_keys($array);
// retourne array(0, "couleur")
  $array = array("bleu", "rouge", "vert", "bleu", "bleu");
  array_keys($array, "bleu");
//  retourne  array(0, 3, 4)
  $array = array( "couleur" => array("bleu", "rouge", "vert"),
                "taille"  => array("petit", "moyen", "grand") );
  array_keys($array);
//  retourne array("couleur", "taille")
?>
```

Note : array_keys() a été ajoutée en PHP 4. Ci-dessous, voici une implémentation qui fonctionnera sous PHP 3:

Exemple 2. Implémentation de array_keys() pour les utilisateurs de PHP 3

```
<?php
  function array_keys ($arr, $term="") {
    $t = array();
    while (list($k,$v) = each($arr)) {
        if ($term && $v != $term)
            continue;
            $t[] = $k;
        }
        return $t;
  }
?>
```

Voir aussi **array_values()**.

array_map (PHP 4 >= 4.0.6)

Applique sur fonction sur des tableaux

```
array array_map (mixed callback, array arr1 [, array ...])
```

array_map() retourne un tableau contenant tous les éléments du tableau *arr1*, après leur avoir appliqué la fonction *callback*. Le nombre de paramètres de la fonction *callback* doit être égal au nombre de tableaux passés dans la fonction **array_map()**.

Exemple 1. Exemple avec array_map()

```
<?php
  function cube($n) {
      return $n*$n*$n;
  }
  $a = array(1, 2, 3, 4, 5);
  $b = array_map("cube", $a);
?>
```

Avec cet exemple, la variable $b contiendra array (1, 8, 27, 64, 125);.

Exemple 2. array_map() - utilisation de plusieurs tableaux

```
<?php
  function parle_espagnol($n, $m) {
      return "Le nombre $n se dit $m en espagnol";
  }
  function map_espagnol($n, $m) {
   return array($n => $m);
  }
  $a = array(1, 2, 3, 4, 5);
  $b = array("uno", "dos", "tres", "cuatro", "cinco");
  $c = array_map("parle_espagnol", $a, $b);
  print_r($c);
// Affichera :
// Array
// (
//     [0] => Le nombre 1 se dit uno en espagnol
//     [1] => Le nombre 2 se dit dos en espagnol
//     [2] => Le nombre 3 se dit tres en espagnol
//     [3] => Le nombre 4 se dit cuatro en espagnol
//     [4] => Le nombre 5 se dit cinco en espagnol
// )
  $d = array_map("map_espagnol", $a , $b);
  print_r($d);
// Affichera :
// Array
// (
//     [0] => Array
//         (
//             [1] => uno
//         )
//
//     [1] => Array
//         (
//             [2] => dos
//         )
//
//     [2] => Array
//         (
//             [3] => tres
//         )
//
//     [3] => Array
//         (
//             [4] => cuatro
//         )
//
//     [4] => Array
//         (
//             [5] => cinco
//         )
//
// )
?>
```

Généralement, lorsque vous utilisez plusieurs tableaux, ils doivent être de même longueur, car la fonction de callback est appliqué à un élément de chaque tableau. Si les tableaux sont de taille inégale, les plus petits seront complétés avec des éléments vides.

Une utilisation interessante de cette fonction est de construire des tableaux de tableaux, grâce à la fonction de callback `NULL`.

Exemple 3. array_map() - création d'un tableau de tableaux

```
<?php
  $a = array(1, 2, 3, 4, 5);
  $b = array("un", "deux", "trois", "quatre", "cinq");
  $c = array("uno", "dos", "tres", "cuatro", "cinco");
  $d = array_map(null, $a, $b, $c);
  print_r($d);
// affichera :
// Array
// (
//     [0] => Array
//         (
//             [0] => 1
//             [1] => un
//             [2] => uno
//         )
//
//     [1] => Array
//         (
//             [0] => 2
//             [1] => deux
//             [2] => dos
//         )
//
//     [2] => Array
//         (
//             [0] => 3
//             [1] => trois
//             [2] => tres
//         )
//
//     [3] => Array
//         (
//             [0] => 4
//             [1] => quatre
//             [2] => cuatro
//         )
//
//     [4] => Array
//         (
//             [0] => 5
//             [1] => cinq
//             [2] => cinco
//         )
//
// )
?>
```

Voir aussi **array_filter()** et **array_reduce()**.

array_merge (PHP 4 >= 4.0b1)

Rassemble plusieurs tableaux

```
array array_merge (array array1, array array2 [, array ...])
```

array_merge() rassemble les éléments de plusieurs tableaux ensembles, en ajoutant les valeurs de l'un à la fin de l'autre. Le résultat est un tableau.

Si les tableaux ont des clés en commun, la dernière valeur rencontrée écrasera l'ancienne. Pour les valeurs numériques, cela n'arrive pas, car alors, les valeurs sont ajoutées en fin de tableau.

Exemple 1. Exemple avec array_merge()

```
<?php
  $array1 = array ("couleur" => "rouge", 2, 4);
  $array2 = array ("a", "b", "couleur" => "vert", "forme" => "trapézoïde");
  array_merge ($array1, $array2);
?>
```

Le résultat sera array("couleur" => "vert", 2, 4, "a", "b", "forme" => "trapézoîde").

Note : array_merge() a été ajoutée dans PHP 4.0.

array_merge_recursive (PHP 4 >= 4.0.1)

Combine plusieurs tableaux ensembles, récursivement

```
array array_merge_recursive (array array1, array array2 [, array ...])
```

array_merge_recursive() rassemble tous les éléments de plusieurs tableaux ensembles, en ajoutant les éléments de l'un à la suite des éléments du précédent. **array_merge_recursive()** retourne le tableau résultant.

Si les tableaux passés en arguments ont les mêmes clés (chaînes de caractères), les valeurs sont alors rassemblées dans un tableau, de manière récursive, de façon à ce que, si l'une de ces valeurs est un tableau elle-même, la fonction la rassemblera avec les valeurs de l'entrée courante. Cependant, si deux tableaux ont la même clé numérique, la dernière valeur n'écrasera pas la précédente, mais sera ajoutée à la fin du tableau.

Exemple 1. Exemple avec array_merge_recursive()

```
<?php
  $ar1 = array ("couleur" => array ("favorie" ?> "rouge"), 5);
  $ar2 = array (10, "couleur" ?> array ("favorie" ?> "vert", "rouge"));
  $result = array_merge_recursive ($ar1, $ar2);
?>
```

Le résultat sera `array("couleur" => array("favorie" => array ("rouge", "vert"), "bleu"), 5, 10)`.

Voir aussi **array_merge()**.

array_multisort (PHP 4 >= 4.0b4)

Tri multi-dimensionnel

```
boolean array_multisort (array ar1 [, mixed arg [, mixed ... [, array ...]]])
```

array_multisort() sert à trier simultanément plusieurs tableaux, ou bien à trier un tableau multi-dimensionnel, suivant l'une ou l'autre de ses dimensions. Les clés sont préservées.

Les tableaux passés en arguments sont traités comme les colonnes d'une table, triées par lignes (un peu comme la clause SQL ORDER BY). Le premier tableau est la clé primaire de tri. Les valeurs du premier tableau qui sont égales, sont triées grâce au tableau suivant, et ainsi de suite...

La structure des arguments de **array_multisort()** est un peu inhabituelle, mais elle est plus souple. Le premier argument DOIT être un tableau, mais les arguments suivants peuvent être des tableaux ou une ou deux options de tri, prises dans les valeurs suivantes :

Options de tri :

- SORT_ASC - Tri en ordre ascendant
- SORT_DESC - Tri en ordre descendant

Options de type de tri:

- SORT_REGULAR - Comparaison normale des valeurs
- SORT_NUMERIC - Comparaison numérique des valeurs
- SORT_STRING - Comparaison alphabétique des valeurs

Une seule option de tri de chaque type peut être appliquée après un tableau. Une option ne s'applique qu'au tableau précédent. Tous les autres sont mis par défaut à SORT_ASC et SORT_REGULAR.

array_multisort() retourne `TRUE` en cas de succès, `FALSE` sinon.

Exemple 1. Trier plusieurs tableaux

```
<?php
  $ar1 = array ("10", 100, 100, "a");
  $ar2 = array (1, 3, "2", 1);
  array_multisort ($ar1, $ar2);
?>
```

Dans cet exemple, après le tri, le premier tableau contient 10, "a", 100, 100; Le deuxième tableau contient 1, 1, "2", 3. Les entrées du second tableau correspondant aux valeurs jumelles du premier tableau (100 et 100), sont aussi triées.

Exemple 2. Classer un tableau multidimensionnel

```
<?php
  $ar = array (array ("10", 100, 100, "a"), array (1, 3, "2", 1));
  array_multisort ($ar[0], SORT_ASC, SORT_STRING,
                   $ar[1], SORT_NUMERIC, SORT_DESC);
?>
```

Dans cet exemple, après le tri, le premier tableau contient 10, 100, 100, "a" (tri alphabétique, ordre croissant); Le deuxième tableau contient 1, 3, "2", 1 (tri numérique, ordre décroissant).

array_pad (PHP 4 >= 4.0b4)

Complète un tableau jusqu'à la longueur spécifiée, avec une valeur.

```
array array_pad (array input, int pad_size, mixed pad_value)
```

array_pad() retourne une copie du tableau *input* complété jusqu'à la taille de *pad_size* avec la valeur *pad_value*. Si *pad_size* est positif, alors le tableau est complété à droite, s'il est négatif, il est complété à gauche. Si la valeur absolue de *pad_size* est plus petite que la taille du tableau *input*, alors le tableau n'est pas complété.

Exemple 1. Exemple avec array_pad()

```
<?php
  $input = array(12, 10, 9);
  $result = array_pad($input, 5, 0);
// Le résultat est array (12, 10, 9, 0, 0)
  $result = array_pad($input, -7, -1);
// Le résultat est array (-1, -1, -1, -1, 12, 10, 9)
  $result = array_pad($input, 2, "noop");
// pas complété
?>
```

array_pop (PHP 4 >= 4.0b1)

Dépile un élément de la fin d'un tableau

```
mixed array_pop (array array)
```

array_pop() dépile et retourne le dernier élément du tableau *array*, le raccourcissant d'un élément. Si *array* est vide, ou n'est pas un tableau, **array_pop()** retourne NULL.

Exemple 1. Exemple avec array_pop()

```
<?php
  $stack = array("orange", "pomme", "framboise");
  $fruit = array_pop($stack);
?>
```

Après ceci, $stack n'a plus que 2 éléments: "orange" et "pomme", tandis que $fruit contient "framboise".

Voir aussi **array_push()**, **array_shift()** et **array_unshift()**.

> **Note : array_pop()** a été ajoutée en PHP 4.0.

array_push (PHP 4 >= 4.0b1)

Empile un ou plusieurs éléments à la fin d'un tableau

```
int array_push (array array, mixed var [, mixed ...])
```

array_push() considère *array* comme une pile, et empile les variables passées en paramètres à la fin de *array*. La longueur du tableau *array* augmente d'autant. Cela a le même effet que :

```
<?php
  $array[] = $var;
```

```
?>
```

repeté pour chaque *var*.

array_push() retourne le nouveau nombre d'éléments du tableau.

Exemple 1. Exemple avec array_push()

```
<?php
  $stack = array (1, 2);
  array_push($stack, "+", 3);
?>
```

Cet exemple fait que $stack a 4 éléments: 1, 2, "+", et 3.

Voir aussi **array_pop()**, **array_shift()** et **array_unshift()**.

> **Note : array_push()** a été ajoutée en PHP 4.0.

array_reverse (PHP 4 >= 4.0b4)

Renverse l'ordre des éléments d'un tableau

```
array array_reverse (array array [, boolean preserve_keys])
```

array_reverse() prend le tableau *array* et retourne un nouveau tableau qui contient les mêmes éléments mais dans l'ordre inverse, en préservant les clés si le paramètre *preserve_keys* vaut TRUE.

Exemple 1. Exemple avec array_reverse()

```
<?php
  $input = array ("php", 4.0, array ("rouge", "vert"));
  $result = array_reverse ($input);
  $result_keyed = array_reverse ($input, TRUE);
?>
```

Au final, $result et $result_keyed contiennent tous les deux array ("rouge", "vert"), 4.0, "php", dans cet ordre. Mais $result_keyed[0] vaut toujours "php".

> **Note : array_reverse()** a été ajoutée en PHP 4.0 Beta 3.

> **Note :** Le second paramètre *preserve_keys* a été ajouté en PHP 4.0.3.

array_reduce (PHP 4 >= 4.0.5)

Réduit itérativement un tableau

```
mixed array_reduce (array input, mixed callback [, int initial])
```

array_reduce() applique itérativement la fonction *callback* aux éléments du tableau *input*, de manière à réduire le tableau à une valeur simple. Si l'argument optionnel *intial* est disponible, il sera utilisé pour initialiser le processus, ou bien comme valeur finale si le tableau est vide.

Exemple 1. Exemple avec array_reduce()

```
<?php
  function rsum($v, $w) {
    $v += $w;
    return $v;
  }
  function rmul($v, $w) {
    $v *= $w;
    return $v;
  }
  $a = array(1, 2, 3, 4, 5);
  $x = array();
  $b = array_reduce($a, "rsum");
  $c = array_reduce($a, "rmul", 10);
  $d = array_reduce($x, "rsum", 1);
?>
```

Dans cet exemple, `$b` contiendra `15`, `$c` contiendra `1200` (= 1*2*3*4*5*10), et `$d` contiendra `1`.

Voir aussi **array_filter()** et **array_map()**.

array_rand (PHP 4 >= 4.0.0)

Prend une ou plusieurs valeurs, au hasard dans un tableau

```
mixed array_rand (array input [, int num_req])
```

array_rand() est pratique lorsque vous voulez sélectionner une ou plusieurs valeurs au hasard dans un tableau. Le paramètre *input* est un tableau, et *num_req* spéficie le nombre de valeurs que vous voulez obtenir (par défaut, c'est 1).

Si vous ne demandez qu'une entrée, **array_rand()** retourne l'index de la valeur. Sinon, elle retourne un tableau d'index. Cela vous permet de faire une sélection au hasard de clés, ou bien de valeurs.

N'oubliez pas d'appeler **srand()** pour initialiser le générateur de nombres aléatoires.

Exemple 1. Exemple avec array_rand()

```
<?php
  srand ((double) microtime() * 10000000);
  $input = array ("Neo", "Morpheus", "Trinitée", "Cypher", "Tank");
  $rand_keys = array_rand ($input, 2);
  print $input[$rand_keys[0]]."\n";
  print $input[$rand_keys[1]]."\n";
?>
```

array_shift (PHP 4 >= 4.0b1)

Dépile un élément au début d'un tableau

```
mixed array_shift (array array)
```

array_shift() extrait la première valeur d'un tableau et la retourne, en raccourcissant le tableau d'un élément, et en déplacant tous les éléments vers le bas. Si *array* est vide, ou n'est pas un tableau, **array_shift()** retourne NULL.

Exemple 1. Exemple avec array_shift()

```
<?php
  $args = array("-v", "-f");
  $opt = array_shift($args);
?>
```

Cet exemple aura pour résultat que $args ne contiendra plus que "-f", et $opt contient "-v".

Voir aussi **array_unshift()**, **array_push()** et **array_pop()**.

> **Note : array_shift()** a été ajoutée en PHP 4.0.

array_slice (PHP 4 >= 4.0b1)

Extrait une portion de tableau

```
array array_slice (array array, int offset, int [length])
```

array_slice() retourne une série d' élément du tableau *array* commençant à l'offset *offset* et représentant *length* éléments.

Si *offset* est positif, la série commencera à cet offset dans le tableau *array*. Si *offset* est négatif, cette série commencera à l'offset *offset* mais en commençant à la fin du tableau *array*.

Si *length* est fourni et positif, alors la série retournée aura autant d'éléments. Si *length* est fourni et négatif, alors la série contiendra les éléments depuis l'offset *offset* jusqu'à *length* éléments en partant de la fin. Si *length* est omis, la séquence lira tous les éléments du tableau, depuis l'*offset* précisé jusqu'à la fin du tableau.

Exemple 1. Exemple avec array_slice()

```
<?php
  $input = array("a", "b", "c", "d", "e");
  $output = array_slice($input, 2);      // retourne "c", "d", et "e"
// les trois exemples suivants sont équivalents
  $output = array_slice($input, 2, 2);  // retourne "c", "d"
  $output = array_slice($input, 2, -1);  // retourne "c", "d"
// Equivalent à :
  $offset = 2; $length = -1;
  $output = array_slice($input, 2, count($input) - $offset + $length);
// retourne "c", "d"
  $output = array_slice($input, -2, 1);  // retourne "d"
  $output = array_slice($input, 0, 3);   // retourne "a", "b", et "c"
?>
```

Voir aussi **array_splice()**.

> **Note : array_slice()** a été ajoutée en PHP 4.0.

array_splice (PHP 4 >= 4.0b1)

Efface et remplace une portion de tableau

```
array array_splice (array input, int offset [, int length, array [replacement]])
```

array_splice() supprime les éléments désignés par *offset* et *length* du tableau *input* et les remplace par les éléments du tableau *replacement*, si ce dernier est présent.

Si *offset* est positif, la série commencera à cet offset dans le tableau *input*. Si *offset* est négatif, cette série commencera à l'offset *offset* mais en commençant à la fin du tableau *input*.

Si *length* est donné et positif, alors la série aura autant d'éléments. Si *length* est donné et négatif, les éléments seront pris dans l'ordre inverse. Si *length* est omis, la séquence lira tous les éléments du tableau, depuis l'offset *offset* jusqu'à la fin du tableau. Conseil : pour supprimer tous les éléments du tableau depuis *offset* jusqu'à la fin, même si un tableau de remplacement *replacement* est spécifié, utilisez count(`count($input)`) à la place de length.

Si *replacement* est précisé, alors les éléments supprimés sont remplacés par les éléments de ce tableau. Si l'*offset* et *length* sont tels que la taille du tableau ne change pas, alors les éléments du tableau de remplacement replacement sont insérés à partir de l'offset *offset*.

Conseil : si le tableau de remplacement ne contient qu'un seul élément, il n'est pas obligatoire de forcer le type en tableau avec **array()**, à moins que cette variable ne soit elle-même un tableau.

Les codes suivants sont équivalents :

```
<?php
  array_push($input, $x, $y)     array_splice($input, count($input), 0, array($x, $y))
  array_pop($input)              array_splice($input, -1)
  array_shift($input)            array_splice($input, 0, 1)
  array_unshift($input, $x, $y)  array_splice($input, 0, 0, array($x, $y))
  $a[$x] = $y                    array_splice($input, $x, 1, $y)
?>
```

array_splice() retourne le tableau des éléments supprimés.

Exemple 1. Exemples avec array_splice()

```
<?php
// cas simple
  $input = array("rouge", "vert", "bleu", "jaune");
  array_splice($input, 2);
// $input est array("rouge", "vert")
// nombre d'éléments négatif
  $input = array("rouge", "vert", "bleu", "jaune");
  array_splice($input, 1, -1);
// $input est array("rouge", "jaune")
// avec un élément de remplacement
  $input = array("rouge", "vert", "bleu", "jaune");
  array_splice($input, 1, count($input), "orange");
// $input est array("rouge", "orange")
//  cas complexe
  $input = array("rouge", "vert", "bleu", "jaune");
  array_splice($input, -1, 1, array("noir", "marron"));
// $input est array("rouge", "vert",
//          "bleu", "noir", "marron")
?>
```

Voir aussi **array_slice()**.

Note : array_splice() a été ajoutée en PHP 4.0.

array_sum (PHP 4 >= 4.0.4)

Calcule la somme des valeurs du tableau

```
mixed array_sum (array arr)
```

array_sum() retourne la somme des valeurs du tableau, sous forme d'un entier ou d'un nombre à virgule flottante.

Exemple 1. Exemple avec array_sum()

```
<?php
  $a = array(2,4,6,8);
  echo "somme(a) = ".array_sum($a)."\n";
// affiche : somme(a) = 20
  $b = array("a"=>1.2,"b"=>2.3,"c"=>3.4);
  echo "somme(b) = ".array_sum($b)."\n";
// affiche : somme(b) = 6.9
?>
```

array_unique (PHP 4 >= 4.0.1)

Dédoublonne un tableau

```
array array_unique (array array)
```

array_unique() prend le tableau *array* et retourne un nouveau tableau, complètement dédoublonné.

Notez que les clés sont préservées. **array_unique()** conserve la clé de la première valeur rencontrée, et ignore toutes les suivantes.

Exemple 1. Exemple avec array_unique()

```
<?php
  $input = array ("a" => "vert", "rouge", "b" => "vert", "bleu", "rouge");
  $result = array_unique ($input);
  print_r($result);
// Cela va afficher :
//Array
//(
//    [a] => vert
//    [0] => rouge
//    [1] => bleu
//)
?>
```

Notez aussi que **array_unique()** tient compte du type de la valeur. Cela ne porte généralement pas à conséquence, sauf si votre tableau contient des nombres, qui peuvent être de différents types. Cela conduit à des résultats déroutants.

Exemple 2. array_unique() et les types de valeurs

```
<?php
  $input = array(4,"3",3,"4",4,4);
  $result = array_unique($input);
  print_r($result);
// Cela va afficher :
//Array
//(
//  [0] => 3
//  [1] => 3
//  [2] => 4
//  [3] => 4
//)
?>
```

array_unshift (PHP 4 >= 4.0b1)

Empile un ou plusieurs éléments au début d'un tableau

```
int array_unshift (array array, mixed var [, mixed ...])
```

array_unshift() ajoute les éléments passés en argument au début du tableau *array*. Notez que les éléments sont ajoutés comme un tout, et qu'ils restent dans le même ordre.

array_unshift() retourne le nouveau nombre d'éléments du tableau *array*.

Exemple 1. Exemples avec array_unshift()

```
<?php
  $queue = array("p1", "p3");
  array_unshift($queue, "p4", "p5", "p6");
?>
```

Le résultat de cet exemple est que $queue aura 5 éléments, à savoir: "p4", "p5", "p6", "p1", et "p3".

Voir aussi **array_shift()**, **array_push()**, et **array_pop()**.

> **Note : array_unshift()** a été ajoutée en PHP 4.0.

array_values (PHP 4 >= 4.0b1)

Retourne les valeurs d'un tableau

```
array array_values (array input)
```

array_values()retourne les valeurs du tableau *input*.

Exemple 1. Exemples avec array_values()

```
<?php
  $array = array("taille" => "XL", "couleur" => "or");
  array_values($array);    // // retourne array("XL", "or")
?>
```

Note : array_values() a été ajoutée en PHP 4. Ci-dessous, voici une implémentation pour ceux qui utilisent toujours PHP 3.

Exemple 2. Implémentation de array_values() pour les utilisateurs PHP 3

```
<?php
  function array_values($arr){
    $t = array();
    while (list($k, $v) = each($arr)){
        $t[] = $v;
    }
    return $t;
  }
?>
```

array_walk (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Exécute une fonction sur chacun des membres d'un tableau.

```
int array_walk (array arr, string func, mixed userdata)
```

array_walk() exécute la fonction *func* avec chaque élément du tableau *arr*. Les éléments sont passés en tant que premier argument de la fonction *func*. *func* doit être une fonction définie par l'utilisateur, et non pas une fonction native PHP. Vous ne pouvez pas utiliser **array_walk()** directement avec **str2lower()**, il faut absolument passer par une fonction utilisateur.

Si *func* a besoin de plus d'un argument, une alerte sera générée pour chaque appel de *func*. Ces alertes sont supprimées en ajoutant le suffixe '@' avant l'appel de **array_walk()** ou simplement en utilisant **error_reporting()**.

Note : Si *func* doit travailler avec les véritables valeurs du tableau, spécifiez que le premier paramètre de *func* doit être passé par référence. Alors, les éléments seront directement modifiés dans le tableau.

Note : Passer les clés et *userdata* à *func* a été ajouté en PHP 4.0.

En PHP 4, **reset()** doit être appelé si nécessaire, car **array_walk()** ne réinitialise pas automatiquement le tableau.

Exemple 1. Exemple avec array_walk()

```
<?php
  $fruits = array ("d"=>"citron", "a"=>"orange", "b"=>"banane", "c"=>"pomme");
  function test_alter (&$item1, $key, $prefix) {
    $item1 = "$prefix: $item1";
  }
  function test_print ($item2, $key) {
    echo "$key. $item2<br>\n";
  }
  array_walk ($fruits, 'test_print');
```

```
  reset ($fruits);
  array_walk ($fruits, 'test_alter', 'fruit');
  reset ($fruits);
  array_walk ($fruits, 'test_print');
?>
```

Voir aussi **each()** et **list()**.

arsort (PHP 3, PHP 4 >= 4.0b1)

Trie un tableau en ordre inverse

```
void arsort (array array)
```

arsort() trie un tableau de telle manière que la corrélation entre les index et les valeurs soit conservée. L'usage principal est lors de tri de tableaux associatifs où l'ordre des éléments est important.

Exemple 1. Exemple avec arsort()

```
<?php
  $fruits = array("d"=>"papaye","a"=>"orange","b"=>"banane","c"=>"ananas");
  arsort ($fruits);
  for (reset ($fruits); $key = key ($fruits); next ($fruits)) {
    echo "fruits[$key] = ".$fruits[$key]."\n";
  }
?>
```

Cet exemple va afficher: `fruits[d] = papaye fruits[a] = orange fruits[b] = banane fruits[c] = ananas` Les fruits ont été triés en ordre alphabétique inverse, et leurs index respectifs ont été conservés.

Voir aussi **array-multisort()**, **asort()**, **krsort()**, **ksort()**, **natsort()**, **natcasesort()**, **rsort()**, **sort()**, **uasort()**, **uksort()** et **usort()**.

asort (PHP 3, PHP 4 >= 4.0b1)

Trie un tableau en ordre

```
void asort (array array)
```

asort() trie un tableau de telle manière que la corrélation entre les index et les valeurs soit conservée. L'usage principal est lors de tri de tableaux associatifs où l'ordre des éléments est important.

Exemple 1. Exemple avec asort()

```
<?php
  $fruits = array(  "d"=>"papaye",
                    "a"=>"orange",
                    "b"=>"banane",
                    "c"=>"ananas");
  asort($fruits);
  for(reset($fruits); $key = key($fruits); next($fruits)) {
    echo "fruits[$key] = ".$fruits[$key]."\n";
  }
?>
```

Cet exemple va afficher: `fruits[c] = ananas fruits[b] = banane fruits[a] = orange fruits[d] = papaye` Les fruits ont été triés par ordre alphabétique, et leurs index respectifs ont été conservés.

Voir aussi **array-multisort()**, **arsort()**, **krsort()**, **ksort()**, **natsort()**, **natcasesort()**, **rsort()**, **sort()**, **uasort()**, **uksort()** et **usort()**.

compact (PHP 4 >= 4.0b1)

Crée un tableau contenant les variables et leur valeur

```
array compact (string|array varname [, mixed ...])
```

compact() accepte différents paramètres. Les paramètres peuvent être des variables contenant des chaînes, ou un tableau de chaînes, qui peut contenir d'autres tableaux de noms, que **compact()** traitera récursivement.

Pour chacun des arguments, **compact()** recherche une variable avec une variable de même nom dans la table courante des symboles, et l'ajoute dans le tableau, de manière à avoir la relation nom => 'valeur de variable'. En bref, c'est le contraire de la fonction **extract()**. **compact()** retourne le tableau ainsi créé.

Exemple 1. Exemple avec compact()

```
<?php
  $ville = "San Francisco";
  $etat = "CA";
  $evenement = "SIGGRAPH";
  $location_vars = array("ville", "etat");
  $result = compact("evenement", $location_vars);
?>
```

Après cette opération, $result sera le tableau suivant : array(("evenement" => "SIGGRAPH", "ville" => "San Francisco", "etat" => "CA").

Voir aussi **extract()**.

Note : compact() a été ajoutée en PHP 4.0.

count (PHP 3, PHP 4 >= 4.0b1)

Compte le nombre d'éléments d'un tableau

```
int count (mixed var)
```

count() retourne le nombre d'éléments dans *var*, qui est généralement un tableau (et tout le reste n'aura qu'un élément).

count() retourne 1 si la variable n'est pas un tableau.

count() retourne 0 si la variable n'est pas créée.

Avertissement

count() peut retourner 0 pour une variable qui n'a pas été affectée, ou pour un tableau vide. Utilisez plutôt **isset()** pour tester si la variable existe.

Exemple 1. Exemple avec count()

```
<?php
  $a[0] = 1;
  $a[1] = 3;
  $a[2] = 5;
  $result = count ($a);
  //$result == 3
?>
```

Voir aussi **sizeof()**, **isset()** et **is_array()**.

current (PHP 3, PHP 4 >= 4.0b1)

Transforme une variable en tableau

```
mixed current (array array)
```

Chaque tableau entretient un pointeur interne, qui est initialisé lorsque le premier élément est inséré dans le tableau.

current() ne fait que retourner l'élément courant pointé par le pointeur interne du tableau *array*. **current()** ne déplace pas le pointeur. Si le pointeur est au-delà du dernier élément de la liste, **current()** retourne FALSE.

> **Avertissement**
>
> Si le tableau des éléments vides ou des zéros (0 ou "", la chaîne vide) alors **current()** retournera FALSE pour ces éléments. Il est donc impossible de déterminer si vous êtes réellement à la fin de la liste en utilisant la fonction **current()**. Pour passer en revue proprement un tableau qui peut contenir des éléments vides ou des zéros, utilisez la fonction **each()**.

Voir aussi **end()**, **next()**, **prev()** et **reset()**.

each (PHP 3, PHP 4 >= 4.0b1)

Retourne chaque paire clé/valeur d'un tableau

```
array each (array array)
```

each() retourne la paire (clé/valeur) courante du tableau *array* et avance le pointeur de tableau. Cette paire est retournée dans un tableau de 4 éléments, avec les clés *0*, *1*, *key*, et *value*. Les éléments *0* et *key* contiennent le nom de la clé et, et *1* et *value* contiennent la valeur.

Si le pointeur interne de fichier est au-delà de la fin du tableau, **each()** retourne FALSE.

Exemple 1. Exemples avec each()

```
<?php
  $foo = array("bob", "fred", "jussi", "jouni", "egon", "marliese");
  $bar = each($foo);
?>
```

$bar contient maintenant les paires suivantes:

- 0 => 0
- 1 => 'bob'
- key => 0

- value => ’bob’

```
<?php
  $foo = array ("Robert" => "Bob", "Seppo" => "Sepi");
  $bar = each ($foo);
?>
```

`$bar` contient maintenant les paires suivantes:

- 0 => ’Robert’
- 1 => ’Bob’
- key => ’Robert’
- value => ’Bob’

each() est utilisé conjointement avec **list()** pour étudier tous les éléments d’un tableau; par exemple, $HTTP_POST_VARS:

Exemple 2. Affichage de $HTTP_POST_VARS avec each()

```
<?php
  echo "Valeurs transmises par la méthode POST:<br>";
  reset ($HTTP_POST_VARS);
  while (list ($key, $val) = each ($HTTP_POST_VARS)) {
    echo "$key => $val<br>";
  }
?>
```

Après chaque **each()**, le pointeur de tableau est déplacé au dernier élément, ou sur le dernier élément, lorsqu’on arrive à la fin.

Voir aussi **key()**, **list()**, **current()**, **reset()**, **next()** et **prev()**.

end (PHP 3, PHP 4 >= 4.0b1)

Positionne le pointeur de tableau en fin de tableau

```
end (array array)
```

end() déplace le pointeur interne du tableau *array* jusqu’au dernier élément.

Voir aussi **current()**, **each()**, **end()**, **next()** et **reset()**.

extract (PHP 3>= 3.0.7, PHP 4 >= 4.0b1)

Importe les variables dans la table des symboles

```
int extract (array var_array [, int extract_type [, string prefix]])
```

extract() sert à exporter un tableau vers la table des symboles. Elle prend un tableau associatif *var_array*, crée les variables dont les noms sont les index de ce tableau, et leur affecte la valeur associée. Pour chaque paire clé/valeur, **extract()** crée une variable, avec les paramètres *extract_type* et *prefix*.

Note : Depuis la version 4.0.5, **extract()** retourne le nombre de variables extraites.

extract() vérifie l'existence de la variable avant de la créer. Le traitement des collisions est déterminé par *extract_type*. Ce paramètre peut prendre une des valeurs suivantes :

EXTR_OVERWRITE

Lors d'une collision, réécrire la variable existante.

EXTR_SKIP

Lors d'une collision, ne pas réécrire la variable existante.

EXTR_PREFIX_SAME

Lors d'une collision, ajouter le préfixe *prefix*, et créer une nouvelle variable.

EXTR_PREFIX_ALL

Ajouter le préfixe *prefix*, et créer une nouvelle variable.

EXTR_PREFIX_INVALID

Préfixer uniquement les variables aux noms invalides ou numériques avec le préfixe *prefix*. Ceci a été ajouté en PHP 4.0.5.

Si *extract_type* est omis, **extract()** utilise EXTR_OVERWRITE par défaut.

Notez que *prefix* n'est nécessaire que pour les valeurs de *extract_type* suivantes : EXTR_PREFIX_SAME, EXTR_PREFIX_ALL ou EXTR_PREFIX_INVALID. Le résultat préfixé n'est pas un nom de variable valide, il ne sera pas importé dans la table des symboles.

extract() retourne le nombre de variables réellement importées dans la table des symboles.

Une utilisation possible de la fonction **extract()** est l'exportation vers la table des symboles de tableaux de variables retournés par **wddx_deserialize()**.

Exemple 1. Exemple avec extract()

```
<?php
/* Supposons que $var_array est un tableau retourné par
   wddx_deserialize() */
  $taille = "grand";
  $var_array = array("couleur" => "bleu",
                     "taille"  => "moyen",
                     "forme" => "sphere");
  extract($var_array, EXTR_PREFIX_SAME, "wddx");
  print "$couleur, $taille, $forme, $wddx_taille\n";
?>
```

L'exemple ci-dessus va afficher `bleu, large, sphere, moyen`

La variable $taille n'a pas été réécrite, car on avait spécifié le paramètre EXTR_PREFIX_SAME, qui a permis la création $wddx_size. Si EXTR_SKIP avait été utilisé, alors $wddx_size n'aurait pas été créé. Avec EXTR_OVERWRITE, $taille aurait pris la valeur "moyen", et avec EXTR_PREFIX_ALL, les variables créées seraient $wddx_couleur, $wddx_taille, et $wddx_forme.

in_array (PHP 4 >= 4.0b1)

Indique si une valeur appartient à un tableau

```
boolean in_array (mixed needle, array haystack [, boolean strict])
```

in_array() recherche *needle* dans *haystack* et retourne TRUE s'il s'y trouve, ou FALSE sinon.

Le troisième paramètre *strict* est optionnel. S'il vaut TRUE alors **in_array()** vérifiera aussi que le types du paramètre *needle* correspond à la valeur trouvée dans *haystack*.

Exemple 1. Exemple avec in_array()

```
<?php
  $os = array("Mac", "NT", "Irix", "Linux");
  if (in_array("Irix", $os))
       print "Irix trouve";
?>
```

Exemple 2. in_array() avec le paramètre *strict*

```
<?php
  $a = array('1.10', 12.4, 1.13);
  if (in_array('12.4', $a, TRUE))
    echo "'12.4' trouvé avec une recherche stricte\n";
  if (in_array(1.13, $a, TRUE))
    echo "1.13 trouvé avec une recherche stricte\n";
?>
```

L'affichage sera :

```
1.13 trouvé avec une recherche stricte
```

Note : in_array() a été ajoutée en PHP 4.0.

Voir aussi **array_search()**.

array_search (PHP 4 >= 4.0.5)

Recherche dans un tableau la clé associée à une valeur

```
mixed array_search (mixed needle, array haystack, boolean strict)
```

array_search() recherche *needle* dans *haystack* et retourne la clé associée s'il la trouve, ou FALSE sinon.

Si le troisième paramètre *strict* vaut TRUE, alors **array_search()** s'assurera aussi que le type de *needle* est le même que celui de la valeur trouvée dans *haystack*.

Voir aussi **in_array()**.

key (PHP 3, PHP 4 >= 4.0b1)

Retourne une clé d'un tableau associatif

```
mixed key (array array)
```

key() retourne l'index de la clé courante dans un tableau.

Voir aussi **current()** et **next()**

krsort (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Trie un tableau en sens inverse et suivant les clés

```
int krsort (array array)
```

krsort() trie un tableau en ordre inverse et suivant les clés, en maintenant la correspondance entre les clés et les valeurs. Cette fonction est pratique pour les tableaux associatifs.

Exemple 1. Exemple avec krsort()

```
<?php
  $fruits = array("d"=>"papaye","a"=>"orange","b"=>"banane","c"=>"ananas");
  ksort($fruits);
  for(reset($fruits); $key = key($fruits); next($fruits)) {
    echo "fruits[$key] = ".$fruits[$key]."\n";
  }
?>
```

Cet exemple va afficher : `fruits[d] = citron fruits[c] = ananas fruits[b] = banane fruits[a] = orange`

Voir aussi **array-multisort()**, **arsort()**, **asort()**, **ksort()**, **natsort()**, **natcasesort()**, **rsort()**, **sort()**, **uasort()**, **uksort()** et **usort()**.

ksort (PHP 3, PHP 4 >= 4.0b1)

Trie un tableau suivant les clés

```
int ksort (array array)
```

ksort() trie un tableau suivant les clés, en maintenant la correspondance entre les clés et les valeurs. Cette fonction est pratique pour les tableaux associatifs.

Exemple 1. Exemple avec ksort()

```
<?php
  $fruits = array("d"=>"papaye","a"=>"orange","b"=>"banane","c"=>"ananas");
  ksort($fruits);
  reset($fruits);
  while (list ($key, $val) = each ($fruits)) {
    echo "$key => $val\n";
  }
?>
```

Cet exemple va afficher : `fruits[a] = orange fruits[b] = banane fruits[c] = ananas fruits[d] = citron`

Vous pouvez modifier le comportement du tri avec les options *`sort_flags`*. Pour plus de détails, voyez **sort()**.

Voir aussi **array-multisort()**, **arsort()**, **asort()**, **krsort()**, **natsort()**, **natcasesort()**, **rsort()**, **sort()**, **uasort()**, **uksort()** et **usort()**.

> **Note :** Le second paramètre a été ajouté en PHP 4.0.

list (unknown)

Transforme une liste de variables en tableau

```
void list (void)
```

Tout comme **array()**, **list()** n’est pas une véritable fonction, mais une construction syntaxique, qui permet d’assigner une série de variables en une seule ligne.

Exemple 1. Exemple avec list()

```
<?php
<table>
 <tr>
  <th>Nom de l’employé</th>
  <th>Salaire</th>
 </tr>
<?php
  $result = mysql_query($conn, "SELECT id, name, salary FROM employees");
  while (list($id, $name, $salary) = mysql_fetch_row ($result)) {
    print (" <tr>\n".
           "  <td><a href=\"info.php3?id=$id\">$name</a></td>\n".
           "  <td>$salaire</td>\n".
           " </tr>\n");
  }
?>
</table>
?>
```

Voir aussi **each()** et **array()**.

natsort (PHP 4 >= 4.0RC2)

Tri d’un tableau avec l’algorithme à "ordre naturel"

```
void natsort (array array)
```

natsort() implémente un algorithme de tri qui traite les chaînes alpha-numériques comme un être humain : c’est ce qui est appelé l’"ordre naturel". Un exemple de la différence de traitement entre un tel algorithme et un algorithme de tri de chaînes (comme lorsqu’on utilise **sort()**) est illustré ci-dessous :

Exemple 1. Exemple avec natsort()

```
<?php
  $array1 = $array2 = array ("img12.png","img10.png","img2.png","img1.png");
  sort($array1);
  echo "Tri Standard\n";
  print_r($array1);
  natsort($array2);
  echo "\nTri par Ordre Naturel\n";
  print_r($array2);
?>
```

L’exemple ci-dessous génère l’affichage suivant :

```
Tri Standard
```

```
Array
(
    [0] => img1.png
    [1] => img10.png
    [2] => img12.png
    [3] => img2.png
)
Tri par Ordre Naturel
Array
(
    [3] => img1.png
    [2] => img2.png
    [1] => img10.png
    [0] => img12.png
)
?>
```

Pour plus de détails, rendez-vous sur le site de Martin Pool Natural Order String Comparison (http://www.linuxcare.com.au/projects/natsort/).

Voir aussi **array-multisort()**, **arsort()**, **asort()**, **krsort()**, **ksort()**, **natsort()**, **natcasesort()**, **rsort()**, **sort()**, **uasort()**, **uksort()**, **usort()**, **strnatcmp()** et **strnatcasecmp()**.

natcasesort (PHP 4 >= 4.0RC2)

Tri d'un tableau avec l'algorithme à "ordre naturel" insensible à la casse

```
void natcasesort (array array)
```

natcasesort() implémente un algorithme de tri qui traite les chaînes alpha-numériques comme un être humain : c'est ce qui est appelé l'"ordre naturel".

natcasesort() est la version insensible à la casse de **natsort()**. Voir aussi **natsort()** pour un exemple illustré.

Pour plus de détails, rendez-vous sur le site de : Martin Pool's Natural Order String Comparison (http://www.linuxcare.com.au/projects/natsort/).

Voir aussi **array-multisort()**, **arsort()**, **asort()**, **krsort()**, **ksort()**, **natsort()**, **rsort()**, **sort()**, **uasort()**, **uksort()**, **usort()**, **strnatcmp()** et **strnatcasecmp()**.

next (PHP 3, PHP 4 >= 4.0b1)

Avance le pointeur interne d'un tableau

```
mixed next (array array)
```

next() retourne l'élément suivant du tableau, ou FALSE s'il n'y a plus d'éléments. Le pointeur de interne de tableau est avancé d'un élément.

next() se comporte comme **current()**, mais avec une différence : il avance le pointeur interne de tableau d'un élément avant de retourner la valeur qu'il pointe. Lorsque le pointeur dépasse le dernier élément, **next()** retourne FALSE.

> **Avertissement**
>
> Si le tableau contient des éléments vides ou des zéros, **next()** retournera FALSE pour ces éléments. Pour passer proprement en revue un tableau, il faut utiliser **each()**.

Voir aussi **current()**, **end()**, **prev()** et **reset()**.

pos (PHP 3, PHP 4 >= 4.0b1)

Retourne l'élément courant d'un tableau

```
mixed pos (array array)
```

pos() est une fonction alias de **current()**.

Voir aussi **end()**, **next()**, **prev()** et **reset()**.

prev (PHP 3, PHP 4 >= 4.0b1)

Recule le pointeur courant de tableau

```
mixed prev (array array)
```

prev() repositionne le pointeur interne de tableau à la dernière place qu'il occupait, ou bien retourne FALSE s'il ne reste plus d'éléments.

> **Avertissement**
>
> Si le tableau contient des éléments vides, **prev()** retournera FALSE pour ces éléments aussi. Pour passer en revue tous les éléments, utilisez plutôt **each()**.

prev() se comporte exactement comme **next()**, mais il fait reculer le pointeur plutôt que de l'avancer.

Voir aussi **current()**, **end() next()** et **reset()**.

range (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Crée un tableau contenant un intervalle d'éléments

```
array range (int low, int high)
```

range() retourne un tableau contenant tous les entiers depuis *low* jusqu'à *high*, inclus.

Voir aussi **shuffle()** (pour un exemple d'utilisation).

reset (PHP 3, PHP 4 >= 4.0b1)

Remet le pointeur interne de tableau au début

```
mixed reset (array array)
```

reset() replace le pointeur de tableau *array* au premier élément.

reset() retourne la valeur du premier élément.

Voir aussi **current()**, **each()**, **next()**, **prev()** et **reset()**.

rsort (PHP 3, PHP 4 >= 4.0b1)

Trie en ordre inverse

```
void rsort (array array)
```

rsort() effectue un tri en ordre décroissant (du plus grand au plus petit).

Exemple 1. Exemple avec rsort()

```
<?php
  $fruits = array("papaye","orange","banane","ananas");
  rsort($fruits);
  for (reset($fruits); list($key,$value) = each($fruits); ) {
    echo "fruits[$key] = ", $value, "\n";
}
?>
```

Cet exemple va afficher: `fruits[0] = papaye fruits[1] = orange fruits[2] = banane fruits[3] = ananas` Les fruits ont été classés dans l'ordre alphabétique inverse.

Voir aussi **array-multisort()**, **arsort()**, **asort()**, **krsort()**, **ksort()**, **natsort()**, **natcasesort()**, **sort()**, **uasort()**, **uksort()** et **usort()**.

shuffle (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Mélange les éléments d'un tableau

```
void shuffle (array array)
```

shuffle() mélange les éléments d'un tableau.

Exemple 1. Exemple avec shuffle()

```
<?php
  $numbers = range (1,20);
  srand (time());
  shuffle ($numbers);
  while (list(, $number) = each ($numbers)) {
    echo "$number ";
}
?>
```

Voir aussi **array-multisort()**, **arsort()**, **asort()**, **krsort()**, **ksort()**, **natsort()**, **natcasesort()**, **rsort()**, **sort()**, **uasort()**, **uksort()** et **usort()**.

sizeof (PHP 3, PHP 4 >= 4.0b1)

Retourne le nombre d'élément d'un tableau

```
int sizeof (array array)
```

sizeof() retourne le nombre d'élément d'un tableau.

Voir aussi **count()**.

sort (PHP 3, PHP 4 >= 4.0b1)

Trie le tableau

```
void sort (array array)
```

sort() trie le tableau *array*. Les éléments seront triés du plus petit au plus grand.

Exemple 1. Exemple avec sort()

```
<?php
  $fruits = array("papaye","orange","banane","ananas");
  sort($fruits);
  for(reset($fruits); $key = key($fruits); next($fruits)) {
    echo "fruits[$key] = ".$fruits[$key]."\n";
  }
?>
```

Cet exemple va afficher : `fruits[0] = ananas fruits[1] = banane fruits[2] = orange fruits[3] = papaye` Les fruits ont été classés dans l'ordre alphabétique.

Voir aussi **array-multisort()**, **arsort()**, **asort()**, **krsort()**, **ksort()**, **natsort()**, **natcasesort()**, **rsort()**, **uasort()**, **uksort()** et **usort()**.

uasort (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Trie d'un tableau en utilisant une fonction de comparaison définie par l'utilisateur.

```
void uasort (array array, function cmp_function)
```

uasort() trie un tableau en conservant la correspondance entre les index et leurs valeurs. **uasort()** sert essentiellement lors de tri de tableaux associatifs où l'ordre des éléments est significatif. La fonction de comparaison utilisée est définie par l'utilisateur.

uksort (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Trie un tableau par ses clés en utilisant une fonction de comparaison définie par l'utilisateur

```
void uksort (array array, function cmp_function)
```

uksort() trie les clés du tableau en utilisant une fonction définie par l'utilisateur. Si un tableau doit être trié avec un critère complexe, il est préférable d'utiliser **uksort()**.

Exemple 1. Exemple avec uksort()

```
<?php
  function mycompare($a, $b) {
    if ($a == $b) return 0;
    return ($a > $b) ? -1 : 1;
  }
  $a = array(4 => "quatre", 3 => "trois", 20 => "vingt", 10 => "dix");
  uksort($a, mycompare);
  while(list($key, $value) = each($a)) {
    echo "$key: $value\n";
  }
?>
```

Cet exemple affichera: `20: vingt 10: dix 4: quatre 3: trois`

Voir aussi **array-multisort()**, **arsort()**, **asort()**, **krsort()**, **ksort()**, **natsort()**, **natcasesort()**, **rsort()**, **sort()**, **uasort()** et **usort()**.

usort (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Trie un tableau en utilisant une fonction de comparaison définie par l'utilisateur

```
void usort (array array, function cmp_function)
```

usort() va trier un tableau avec ses valeurs, en utilisant une fonction définie par l'utilisateur. Si un tableau doit être trié avec un critère complexe, il est préférable d'utiliser cette méthode.

La fonction de comparaison *cmp_function* doit retourner un entier, qui sera inférieur, égal ou supérieur à zéro suivant que le premier argument est considéré comme plus petit, égal ou plus grand que le second argument. Si les deux arguments sont égaux, leur ordre est indéfini.

Exemple 1. Exemple avec usort()

```
<?php
  function cmp($a,$b) {
    if ($a == $b) return 0;
    return ($a < $b) ? -1 : 1;
  }
  $tableau = array(3,2,5,6,1);
  usort($a, "cmp");
  while(list($cle,$valeur) = each($tableau)) {
    echo "$cle: $valeur\n";
  }
?>
```

Cet exemple va afficher : `0: 6 1: 5 2: 3 3: 2 4: 1`

Note : Evidemment dans ce cas trivial, **rsort()** serait plus approprié.

> **Avertissement**
>
> Les bibliothèques de tri rapides sur lesquelles reposent PHP peuvent le conduire à un plantage, si la fonction de comparaison ne retourne pas une valeur cohérente.

Voir aussi **array-multisort()**, **arsort()**, **asort()**, **krsort()**, **ksort()**, **natsort()**, **natcasesort()**, **rsort()**, **sort()**, **uasort()** et **uksort()**.

III. Aspell

Les fonctions Aspell vous permettent de vérifier l'orthographe d'un mot, et d'offrir des suggestions de corrections. Plusieurs langues sont disponibles, comme le franç, l'allemand, le suédois et le danois.

> **Note :** aspell fonctionne avec de très vielles versions (jusqu'à la version .27.* ou presque) de la librairie aspell. Ce module, et ces versions d'Aspell ne sont plus supportées. Si vous voulez utiliser les possibilités de vérifications d'orthographe en PHP, utilisez plutôt pspell. Ce module utilise la librairie pspell qui fonctionne avec les nouvelles versions de Aspell.

Vous avez besoin de la librairie Aspell, disponible à : http://aspell.sourceforge.net/.

aspell_new (PHP 3>= 3.0.7, PHP 4 >= 4.0b1)

Charge un nouveau dictionnaire

```
resource aspell_new (string master, string personal)
```

aspell_new() ouvre un nouveau dictionaire, et retourne un identifiant de dictionnaire pour utilisation ultérieure dans les fonctions aspell.

Exemple 1. Exemple avec aspell_new()

```
<?php
$aspell_link=aspell_new("english");
?>
```

aspell_check (PHP 3>= 3.0.7, PHP 4 >= 4.0b1)

Vérifie un mot

```
boolean aspell_check (resource dictionary_link, string word)
```

aspell_check() vérifie l'orthographe d'un mot et retourne TRUE si l'orthographe est correcte, et FALSE sinon.

Exemple 1. Exemple avec aspell_check()

```
<?php
$aspell_link=aspell_new("english");
if (aspell_check($aspell_link,"testt")) {
    echo "L'orthographe est correcte.";
} else {
    echo "Désolé, l'orthographe est incorrecte.";
}
?>
```

aspell_check_raw (PHP 3>= 3.0.7, PHP 4 >= 4.0b1)

Vérifie un mot sans en changer la casse et sans essayer de supprimer les espaces aux extrémités.

```
boolean aspell_check_raw (resource dictionary_link, string word)
```

aspell_check_raw() vérifie l'orthographe d'un mot sans en changer la casse, et sans essayer de supprimer les espaces aux extrémités. Elle retourne TRUE si l'orthographe est bonne, et FALSE sinon.

Exemple 1. Exemple avec aspell_check_raw()

```
<?php
$aspell_link=aspell_new("french");
if (aspell_check_raw($aspell_link,"testt")) {
    echo "L'orthographe est OK";
} else {
    echo "Attention : faute d'orthographe";
```

```
}
?>
```

aspell_suggest (PHP 3>= 3.0.7, PHP 4 >= 4.0b1)

Suggère l'orthographe d'un mot

```
array aspell_suggest (resource dictionary_link, string word)
```

aspell_suggest() retourne un tableau contenant les orthographes possibles d'un mot mal formé.

Exemple 1. Exemple avec aspell_suggest()

```
<?php
$aspell_link=aspell_new("french");
if (!aspell_check($aspell_link,"testt")) {
    $suggestions=aspell_suggest($aspell_link,"testt");
    for($i=0; $i < count($suggestions); $i++) {
       echo "Orthographes envisageables : " . $suggestions[$i] . "<br>";
    }
}
?>
```

IV. Nombres de grande taille

En PHP 4, ces fonctions ne sont disponibles que si l'option de configuration `--enable-bcmath` a été activée lors de la compilation. En PHP 3, ces fonctions ne sont disponibles que si l'option de configuration `--disable-bcmath` a été n' pas été activée lors de la compilation.

> **Note :** Suite aux changement de licence, la librairie BCMATH est désormais distribuée séparemment. Vous pouvez télécharger l'archive à http://www.php.net/extra/number4.tar.gz. Lisez attentivement le fichier `README.BCMATH` de la distribution PHP.

bcadd (PHP 3, PHP 4 >= 4.0b1)

Additionne deux nombres de grande taille.

```
string bcadd (string left_operand, string right_operand [, int scale])
```

bcadd() additionne *left_operand* avec l'opérande *right_operand* et renvoie la somme sous forme de chaîne de caractères. Le paramètre optionnel *scale* est utilisé pour définir le nombre de chiffres après la virgule dans le résultat.

Voir aussi **bcsub()**.

bccomp (PHP 3, PHP 4 >= 4.0b1)

Compare deux nombres de grande taille.

```
int bccomp (string left_operand, string right_operand [, int scale])
```

bccomp() compare l'opérande *left_operand* avec l'opérande *right_operand* et renvoie le résultat sous forme de valeur numérique (entier). Le paramètre optionnel *scale* est utilisé pour définir le nombre de chiffres après la virgule utilisés lors de la comparaison. Le résultat est 0 si les deux opérandes sont égales. Si l'opérande *left_operand* est plus grande que l'opérande *right_operand*, le résultat est 1. Si l'opérande *left_operand* est plus petite que l'opérande *right_operand*, le résultat est -1.

bcdiv (PHP 3, PHP 4 >= 4.0b1)

Divise deux nombres de grande taille.

```
string bcdiv (string left_operand, string right_operand [, int scale])
```

bcdiv() divise l'opérande *left_operand* par l'opérande *right_operand* et renvoie le résultat. Le paramètre optionnel *scale* définit le nombre de chiffres après la virgule dans le résultat.

Voir aussi **bcmul()**.

bcmod (PHP 3, PHP 4 >= 4.0b1)

Retourne le reste d'une division entre nombre de grande taille.

```
string bcmod (string left_operand, string modulus)
```

bcmod() retourne le reste de la division entre *left_operand* en utilisant *modulus*.

Voir aussi **bcdiv()**.

bcmul (PHP 3, PHP 4 >= 4.0b1)

Multiplie deux nombres de grande taille.

```
string bcmul (string left_operand, string right_operand [, int scale])
```

bcmul() multiplie l'opérande *left_operand* par l'opérande *right_operand* et renvoie le résultat. Le paramètre optionnel *scale* définit le nombre de chiffres après la virgule dans le résultat.

Voir aussi **bcdiv()**.

bcpow (PHP 3, PHP 4 >= 4.0b1)

Elève un nombre à la puissance n-ième.

```
string bcpow (string x, string y [, int scale])
```

bcpow() élève *x* à la puissance *y*. Le paramètre optionnel *scale* définit le nombre de chiffres après la virgule dans le résultat.

Voir aussi **bcsqrt()**.

bcscale (PHP 3, PHP 4 >= 4.0b1)

Détermine le nombre de décimales par défaut

```
string bcscale (int scale)
```

bcscale() définit la précision par défaut pour toutes les fonctions mathématiques sur des nombres de taille arbitraire qui suivent et qui omettent le paramètre *scale*.

bcsqrt (PHP 3, PHP 4 >= 4.0b1)

Renvoie la racine carrée d'un nombre de grande taille.

```
string bcsqrt (string operand [, int scale])
```

bcsqrt() renvoie la racine carrée de l'opérande *operand*. Le paramètre optionnel *scale* définit le nombre de chiffres après la virgule dans le résultat.

Voir aussi **bcpow()**.

bcsub (PHP 3, PHP 4 >= 4.0b1)

Soustrait un nombre de grande taille à un autre.

```
string bcsub (string left_operand, string right_operand [, int scale])
```

bcsub() soustrait l'opérande *right_operand* à l'opérande *left_operand* et renvoie le résultat sous forme de chaîne de caractères. Le paramètre optionnel *scale* définit le nombre de chiffres après la virgule dans le résultat.

Voir aussi **bcadd()**.

V. Compression Bzip2

Ce module utilise les fonctions de la librairie bzip2 (http://sources.redhat.com/bzip2/), de Julian Seward pour écrire et lire des fichier bzip2 (.bz2) de manière transparente.

Le support bzip2 par PHP n'est pas activé par défaut. Vous devez utiliser l'option de configuration `--with-bz2[=DIR]` lors de la compilation de PHP pour l'activer. Ce module requiert la librairie bzip2/libbzip2, version >= 1.0.x.

Exemple de compression bzip2

Cet exemple ouvre un fichier temporaire, et écrit une ligne de test, puis il en affiche le contenu.

Exemple 1. Exemple avec bzip2

```
<?php
  $filename = "/tmp/fichier_de_test.bz2";
  $str = "Ceci est une chaîne de test.\n";
// ouvre le fichier en écriture
  $bz = bzopen($filename, "w");
// écrit une chaîne dans le fichier
  bzwrite($bz, $str);
// ferme le fichier
  bzclose($bz);
// ouvre le fichier en lecture
  $bz = bzopen($filename, "r");
// lit 10 caractères
  print bzread($bz, 10);
// affiche tout le reste du fichier, puis le ferme
  print bzread($bz);
  bzclose($bz);
?>
```

bzclose (PHP 4 >= 4.0.4)

Ferme un fichier bzip2

```
int bzclose (resource bz)
```

bzclose() ferme le fichier bzip2 représenté par le pointeur *bz*.

bzclose() retourne TRUE en cas de succès, et FALSE sinon.

Le pointeur de fichier *bz* doit être valide, et avoir été ouvert avec **bzopen()**.

Voir aussi **bzopen()**.

bzcompress (PHP 4 >= 4.0.4)

Compresse une chaîne avec bzip2

```
string bzcompress (string source [, int blocksize [, int workfactor]])
```

bzcompress() compresse la chaîne *source* et retourne les données ainsi encodée.

Le paramètre optionnel *blocksize* spécifie la taille de bloc utilisé durant la compression, et doit être un nombre de 1 à 9, sachant que 9 représente la meilleure compression, mais qu'elle utilise plus de ressource pour ce faire. *blocksize* vaut par défaut 4.

Le paramètre optionnel *workfactor* contrôle le comportement de la compression dans les pires cas de données hautement répétitives. Cette valeur peut aller de 0 à 250 (0 est une valeur spéciale, et 30 la valeur par défaut). En dehors de *workfactor*, le résultat sera le même.

Exemple 1. Exemple avec bzcompress()

```
<?php
  $str = "données de test";
  $bzstr = bzcompress($str, 9);
?>
```

Voir aussi **bzdecompress()**.

bzdecompress (PHP 4 >= 4.0.4)

Décompresse une chaîne bzip2

```
string bzdecompress (string source [, int small])
```

bzdecompress() décompresse la chaîne *source*, en supposant qu'elle a été compressée avec bzip2, puis la retourne. Si le paramètre optionnel *small* vaut TRUE, un autre algorithme de décompression sera utilisé : il consomme moins de mémoire (le maximum demandé tombe autour de 2300 ko), mais fonctionne globalement à la moitié de la vitesse. Reportez-vous à la documentation bzip2 (http://sources.redhat.com/bzip2/) pour plus de détails sur cette fonctionnalité.

Exemple 1. Exemple avec bzdecompress()

```
<?php
  $str = $bzdecompress($bzstr);
?>
```

Voir aussi **bzcompress()**.

bzerrno (PHP 4 >= 4.0.4)

Retourne le numéro d'erreur bzip2

```
int bzerrno (resource bz)
```

bzerrno() retourne le numéro d'erreur du fichier bz2 représenté par le pointeur *bz*.

Voir aussi **bzerror()** et **bzerrstr()**.

bzerror (PHP 4 >= 4.0.4)

Retourne le numéro et le message d'erreur bzip2 dans un tableau

```
array bzerror (int bz)
```

bzerror() retourne le numéro et le message d'erreur du fichier bz2 représenté par le pointeur *bz*. **bzerror()** retourne un tableau associatif.

Exemple 1. Exemple avec bzerror()

```
<?php
  $error = bzerror($bz);
  echo $error["errno"];
  echo $error["errstr"];
?>
```

Voir aussi **bzerrno()** et **bzerrstr()**.

bzerrstr (PHP 4 >= 4.0.4)

Retourne le message d'erreur bzip2

```
string bzerrstr (resource bz)
```

bzerrstr() retourne le message d'erreur du fichier bz2 représenté par le pointeur *bz*.

Voir aussi **bzerrno()** et **bzerror()**.

bzflush (PHP 4 >= 4.0.4)

Force l'écriture de toutes les données compressées

```
int bzflush (resource bz)
```

bzflush() vide les buffers d'écriture du fichier représenté par *bz*.

bzflush() retourne TRUE en cas de succès, et FALSE sinon.

Voir aussi **bzread()** et **bzwrite()**.

bzopen (PHP 4 >= 4.0.4)

Ouvre un fichier compressé avec bzip2

```
resource bzopen (string filename, string mode)
```

bzopen() ouvre un fichier bzip2 (.bz2) en écriture ou en lecture. *filename* est le nom du fichier à ouvrir. *mode* est similaire au même paramètre de la fonction **fopen()** ('r' pour lecture, 'w' pour écriture, etc.).

Si l'ouverture échoue, **bzopen()** retourne FALSE, sinon, elle retourne un pointeur de fichier.

Exemple 1. Exemple avec bzopen()

```
<?php
  $bz = bzopen("/tmp/foo.bz2", "r");
?>
```

Voir aussi **bzclose()**.

bzread (PHP 4 >= 4.0.4)

Lecture binaire d'un fichier bzip2

```
string bzread (resource bz [, int length])
```

bzread() lit jusqu'à *length* octets depuis le fichier bzip2, référencé par le pointeur *bz*. La lecture s'arrête lorsque *length* octets (non compressés) ont été lus, qu'une erreur est rencontrée, ou bien que la fin du fichier a été atteinte : le premier des trois qui survient. Si le paramètre optionnel *length* est omis, **bzread()** lit 1024 octets (non compressés) en même temps.

Exemple 1. Exemple avec bzread()

```
<?php
  $bz = bzopen("/tmp/foo.bz2", "r");
  $str = bzread($bz, 2048);
  echo $str;
?>
```

Voir aussi **bzwrite()** et **bzopen()**.

bzwrite (PHP 4 >= 4.0.4)

Ecriture binaire dans un fichier bzip2

```
int bzwrite (resource bz, string data [, int length])
```

bzwrite() écrit le contenu de la chaîne *data* dans le fichier bzip2 représenté par *bz*. Si le paramètre optionnel *length* est fourni, l'écriture sera arrêtée après l'écriture de *length* octets (non compressés), ou la fin de la chaîne (le premier qui survient).

Exemple 1. Exemple bzwrite()

```
<?php
  $str = "données non compressées";
  $bz = bzopen("/tmp/foo.bz2", "w");
  bzwrite($bz, $str, strlen($str));
?>
```

Voir aussi **bzread()** et **bzopen()**.

VI. Calendrier

Les fonctions de calendrier ne sont disponibles que si l’extension calendrier a été compilée. Elle est située dans les sous-dossiers "dl" ou "ext" de votre distribution de PHP. Lisez le fichier README pour plus de détails.

L’extension de calendrier propose une série de fonctions qui simplifie les conversions entre les différents formats de calendrier. La référence est le nombre de jour du calendrier Julien. C’est le nombre de jours depuis une date qui commence bien au delà des dates les plus reculées dont on a besoin (située en 4000 avant J.C.). Pour convertir une date d’un calendrier à un autre, il faut d’abord la convertir dans ce calendrier, puis convertir le résultat dans le calendrier désiré. Attention, le nombre de jour du calendrier Julien est un système très différent du calendrier Julien!. Pour plus d’informations (en anglais), reportez vous à http://genealogy.org/~scottlee/cal-overview.html. Les traductions issues de ces pages seront mises entre guillemets.

JDToGregorian (PHP 3, PHP 4 >= 4.0b1)

Convertit le nombre de jours du calendrier Julien en date grégorienne.

```
string jdtogregorian (int julianday)
```

jdtogregorian() convertit le nombre de jours du calendrier Julien en une chaîne contenant une date du calendrier grégorien, au format "mois/jour/année".

GregorianToJD (PHP 3, PHP 4 >= 4.0b1)

Convertit une date grégorienne en nombre de jours du calendrier julien.

```
int gregoriantojd (int month, int day, int year)
```

Intervalle de validité pour le calendrier grégorien : 4714 avant JC à 9999 après JC.A.D.

Bien qu'il soit possible de manipuler des dates jusqu'en 4714 avant JC, une telle utilisation n'est pas significative. En effet, ce calendrier fut créé le 18 octobre 1582 après J.C. (ou 5 octobre 1582 en calendrier grec). Certains pays ne l'acceptèrent que bien plus tard. Par exemple, les britanniques n'y passèrent en 1752, les Russes en 1918 et les Grecs en 1923. La plus part des pays européens utilisaient le calendrier Julien avant le Grégorien.

Exemple 1. Fonctions calendrier

```
<?php
$jd = gregoriantojd(10,11,1970);
echo("$jd\n");
$gregorian = jdtogregorian($jd);
echo("$gregorian\n");
?>
```

JDToJulian (PHP 3, PHP 4 >= 4.0b1)

Convertit le nombre de jours du calendrier Julien en date du calendrier Julien.

```
string jdtojulian (int julianday)
```

jdtojulian() convertit le nombre de jours du calendrier Julien en une chaîne contenant la date du calendrier Julien, au format "mois/jour/année".

JulianToJD (PHP 3, PHP 4 >= 4.0b1)

Convertit le nombre de jour du calendrier Julien en date du calendrier Julien.

```
int juliantojd (int month, int day, int year)
```

Intervalle de validité du calendrier Julien : 4713 avant JC à 9999 après J.C..

Bien qu'il soit possible de manipuler des dates jusqu'en 4713 avant JC, une telle utilisation n'est pas significative. En effet, ce calendrier fut créé en 46 avant JC, et ses détails ne furent finalisés qu'au plus tôt en 8 après JC, et probablement pas avant le 4ème siècle après JC. De plus, le début de l'année variait suivant les peuples, certains n'acceptant pas janvier comme premier mois de l'année.

JDToJewish (PHP 3, PHP 4 >= 4.0b1)

Convertit le nombre de jours du calendrier julien en date du calendrier juif.

```
string jdtojewish (int julianday)
```

jdtojewish() convertit le nombre de jours du calendrier julien en date du calendrier juif.

JewishToJD (PHP 3, PHP 4 >= 4.0b1)

Convertit une date du calendrier juif en nombre de jours du calendrier julien.

```
int jewishtojd (int month, int day, int year)
```

Bien qu'il soit possible de manipuler des dates à partir de l'an 1 (3761 avant JC), une telle utilisation a peu de sens.

Le calendrier juif a été utilisé depuis plusieurs dizaines de siècles, mais dans les premiers temps, il n'y avait pas de formule pour déterminer le début du mois. Un nouveau mois commencait quand une nouvelle lune était observée.

JDToFrench (PHP 3, PHP 4 >= 4.0b1)

Convertit le nombre de jours du calendrier julien en date du calendrier français républicain

```
string jdtofrench (int juliandaycount)
```

jdtofrench() convertit le nombre de jours du calendrier julien en date du calendrier français républicain.

FrenchToJD (PHP 3, PHP 4 >= 4.0b1)

Convertit une date du calendrier français républicain en nombre de jours du calendrier julien.

```
int frenchtojd (int month, int day, int year)
```

frenchtojd() convertit une date du calendrier français républicain en nombre de jour du calendrier julien.

Ces fonctions convertissent les dates comprises entre l'an 1 et l'an 14 (22 September 1792 à 22 September 1806 en calendrier grégorien). Cela couvre plus que la durée d'existence de ce calendrier.

JDMonthName (PHP 3, PHP 4 >= 4.0b1)

Retourne le nom du mois.

```
string jdmonthname (int julianday, int mode)
```

jdmonthname() retourne une chaîne contenant le nom du mois. *mode* indique de quel calendrier dépend ce mois, et quel type de nom doit être retourné.

Mode	Signification

Tableau 1. Modes de calendrier

Mode	Signification
0	Grégorien - abrégé
1	Grégorien
2	Julien - abrégé
3	Julien
4	Juif
5	Républicain français

JDDayOfWeek (PHP 3, PHP 4 >= 4.0b1)

Retourne le numéro du jour de la semaine.

```
mixed jddayofweek (int julianday, int mode)
```

jddayofweek() retourne le numéro du jour de la semaine. Peut retourner une chaîne ou un entier, en fonction du mode.

Tableau 1. Modes

Mode	Signification
0	Retourne le numéro du jour comme un entier (0=dimanche, 1=lundi, etc.)
1	Retourne une chaîne contenant le nom du jour (anglais grégorien)
2	Retourne une chaîne contenant le nom abrégé du jour de la semaine (anglais grégorien).

easter_date (PHP 3>= 3.0.9, PHP 4 >= 4.0RC2)

Retourne un timestamp UNIX pour Pàques, à minuit, pour une année donnée.

```
int easter_date (int year)
```

easter_date() retourne un timestamp UNIX pour Pàques, à minuit, pour une année donnée. Si l'année n'est pas précisée, c'est l'année en cours qui est utilisée.

ATTENTION: **easter_date()** génére une alerte (Warning) si la date tombe hors de la zone de validité des timestamp UNIX (i.e. avant 1970 ou après 2037).

Exemple 1. Exemples avec easter_date()

```
echo date( "M-d-Y", easter_date(1999) );        /* "04 avril 1999" */
echo date( "M-d-Y", easter_date(2000) );        /* "23 avril 2000" */
echo date( "M-d-Y", easter_date(2001) );        /* "15 avril 2001" */
```

La date de Pàques a été fixée par le concile de Nicée, en 325 de notre ère, comme étant le dimanche après la première lune pleine qui suit l'équinoxe de printemps. L'équinoxe de printemps est considéré comme étant toujours le 21 mars, ce qui

réduit le problème au calcul de la date de la lune pleine qui suit, et le dimanche suivant. L'algorithme fut introduit vers 532, par Dionysius Exiguus. Avec le calendrier Julien, (pour les années avant 1753), un cycle de 19 ans suffit pour connaître les date des phases de la lune. Avec le calendrier grégorien, (à partir des années 1753, concu par Clavius et Lilius, puis introduit par le pape Gregoire XIII en Octobre 1582, et en Grande Bretagne et ses colonies en septembre 1752), deux facteurs de corrections ont été ajoutés pour rendre le cycle plus précis.

(Ce code est basé sur le programme en C de Simon Kershaw, <webmaster@ely.anglican.org>)

Voir **easter_days()** pour les calculs de date de Pàques avant 1970 et apres 2037.

easter_days (PHP 3>= 3.0.9, PHP 4 >= 4.0RC2)

Retourne le nombre de jour entre le 21 Mars et Pàques, pour une année donnée.

```
int easter_days (int year)
```

easter_days() retourne le nombre de jour entre le 21 Mars et Pàques, pour une année donnée. Si l'année n'est pas précisée, l'année en cours est utilisée par défaut.

easter_days() peut être utilisée à la place de **easter_date()** pour calculer la date de Pàques, pour les années qui tombent hors de l'intervalle de validité des timestamps UNIX (i.e. avant 1970 ou après 2037).

Exemple 1. Exemple avec easter_days()

```
<?php
echo easter_days(1999);        /* 14, i.e. 4 Avril   */
echo easter_days(1492);        /* 32, i.e. 22 Avril  */
echo easter_days(1913);        /*  2, i.e. 23 Mars   */
?>
```

La date de Pàques a été fixée par le concile de Nicée, en 325 de notre ère, comme étant le dimanche après la première lune pleine qui suit l'équinoxe de printemps. L'équinoxe de printemps est considéré comme étant toujours le 21 mars, ce qui réduit le problème au calcul de la date de la lune pleine qui suit, et le dimanche suivant. L'algorithme fut introduit vers 532, par Dionysius Exiguus. Avec le calendrier Julien, (pour les années avant 1753), un cycle de 19 ans suffit pour connaître les date des phases de la lune. Avec le calendrier grégorien, (à partir des années 1753, concu par Clavius et Lilius, puis introduit par le pape Gregoire XIII en Octobre 1582, et en Grande Bretagne et ses colonies en septembre 1752), deux facteurs de corrections ont été ajoutés pour rendre le cycle plus précis.

(Ce code est basé sur le programme en C de Simon Kershaw, <webmaster@ely.anglican.org>)

Voir aussi **easter_date()**.

unixtojd (PHP 4 >= 4.0RC2)

Convertit un timestamp UNIX en nombre de jours Julien

```
int unixtojd ([int timestamp])
```

unixtojd() retourne le nombre de jours juliens du timestamp UNIX *timestamp* (nombre de secondes depuis le 1/1/1970), ou pour le jour courant si *timestamp* est omis.

Voir aussi **jdtounix()**.

> **Note : unixtojd()** n'est disponible qu'à partir de la version PHP 4.0RC1.

jdtounix (PHP 4 >= 4.0RC2)

Convertit un nombre de jour Julien en timestamp UNIX

```
int jdtounix (int jday)
```

jdtounix() retourne un timestamp UNIX correspondant au nombre de jour julien *jday* ou FALSE si *jday* n'est pas dans l'intervalle de validité de l'époque UNIX. (années grégorienne entre 1970 et 2037 ou 2440588 <= *jday* <= 2465342).

Voir aussi **jdtounix()**.

Note : jdtounix() n'est disponible qu'à partir de la version PHP 4.0RC1.

VII. Paiement CCVS

Ces fonctions font l'interface avec les API CCVS, vous permettant de travailler directement avec CCVS depuis vos scripts PHP. CCVS est la solution apportée par RedHat (http://www.redhat.com/) au problème de l'intermediaire, lors du traitement de transactions de cartes de crédit. Il vous permet travailler directment avec les maisons de crédits, via votre boîte *nix et un modem. En utilisant le module CCVS pour PHP, vous pouvez effectuer des transactions avec les cartes de crédits, directement depuis vos scripts PHP via CCVS. La suite vous montrera comment procéder.

Pour activer le support CCVS de PHP, commencez par vérifier votre installation CCVS. Vous devez configurer PHP avec l'option `--with-ccvs`. Si vous utilisez cette option sans spécifier le chemin de votre installation, PHP essaiera de la trouver à sa position par défaut (/usr/local/ccvs). Si CCVS est installé dans un autre dossier, lancez la configuration avec : `--with-ccvs=$ccvs_path`, où $ccvs_path est le chemin de votre installation CCVS. Notez bien que CCVS requiert que $ccvs_path/lib et $ccvs_path/include existent, et qu'ils contiennent respectivement cv_api.h et libccvs.a sous include et lib.

De plus, un démon ccvsd doit être disponible sur votre configuration, et qu'il soit accessible à vos scripts PHP. Assurez vous aussi que l'utilisateur qui exécute les scripts PHP est le même que celui qui a installé CCVS (i.e. si vous avez installé CCVS avec l'utilisateur 'ccvs', vos scripts PHP doivent tourner aussi en 'ccvs').

Plus de détails sur CCVS sont disponibles à http://www.redhat.com/products/ccvs.

Cette documentation est en chantier. Jusqu'à sa finalisation, RedHat entretient une version légèrement démodée mais bien pratique à http://www.redhat.com/products/ccvs/support/CCVS3.3docs/ProgPHP.html.

(unknown)

( )

VIII. Support COM pour Windows

Ces fonctions ne sont disponibles que sous les versions Windows de PHP. Elles ont été ajoutées dans PHP 4.

com_load (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Crée une référence sur un composant COM

```
resource com_load (string module_name [, string server_name])
```

com_load() crée une nouvelle référence au composant COM *module_name*, et retourne une référence dessus. **com_load()** retourne FALSE en cas d'échec.

com_invoke (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Appelle une méthode d'un composant

```
mixed com_invoke (resource com_object, string function_name, mixed [function parameters, ...])
```

com_invoke() appelle la méthode *function_name* du composant COM *com_object*. **com_invoke()** retourne FALSE en cas d'erreur, sinon retourne le résultat de la fonction *function_name* en cas de succès.

com_propget (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Lit la valeur d'un propriété d'un composant COM

```
mixed com_propget (resource com_object, string property)
```

com_propget() est un alias de **com_get()**.

com_get (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Lit la valeur d'un propriété d'un composant COM

```
mixed com_get (resource com_object, string property)
```

com_get() retourne la valeur de la propriété *property* du composant COM *com_object*. **com_get()** retourne FALSE en cas d'erreur.

com_propput (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Modifie une propriété d'un composant COM

```
void com_propput (resource com_object, string property, mixed value)
```

com_propput() est un alias de **com_set()**.

com_propset (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Modifie une propriété d’un composant COM

```
void com_propset (resource com_object, string property, mixed value)
```

Cette fonction est un alias de **com_propput()**.

com_set (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Modifie une propriété d’un composant COM

```
void com_set (resource com_object, string property, mixed value)
```

com_set() remplace la valeur de la propriété *property* du composante COM *com_object* par *value*. **com_set()** retourne TRUE en cas de succès, et FALSE sinon.

IX. Objets

Introduction

About

Ces fonctions permettent de gérer les classes et les objets. Vous pouvez notamment connaître le nom de la classe d'un objet, ses membres et ses méthodes, et tous les objets parents (les classes qui sont étendues par la classe d'un objet).

Exemple d'utilisation

Dans cet exemple, on définit une classe de base, et une extension. La classe de base définit un légume, s'il est mangeable ou pas, et sa couleur. La sous-classe `epinard` ajoute une méthode pour le cuisiner, et une autre pour savoir s'il est cuisiné.

Exemple 1. classes.inc

```
<?php
// classe de base, avec ses membres et ses méthodes
class Legume {
    var $mangeable;
    var $couleur;
    function legume( $mangeable, $couleur="green" ) {
        $this->mangeable = $mangeable;
        $this->couleur = $couleur;
    }
    function est_mangeable() {
        return $this->mangeable;
    }
    function quelle_couleur() {
        return $this->couleur;
    }
} // fin de la classe Legume
// extend la classe de base
class Epinard extends Legume {
    var $cuit = FALSE;
    function Epinard() {
        $this->Legume( TRUE, "green" );
    }
    function cuisine() {
        $this->cuit = TRUE;
    }
    function est_cuit() {
        return $this->cuit;
    }
} // fin de la classe Epinard
?>
```

Lorsqu'on instantie deux objets de ces classes, et qu'on affiche leurs informations, on affiche aussi leur héritage. On définit ici des utilitaires qui servent essentiellement à afficher ces informations proprement.

Exemple 2. test_script.php

```
<pre>
<?php
  include "classes.inc";
// utilitaires
  function print_vars($obj) {
    $arr = get_object_vars($obj);
    while (list($prop, $val) = each($arr))
```

```
        echo "\t$prop = $val\n";
  }
  function print_methods($obj) {
    $arr = get_class_methods(get_class($obj));
    foreach ($arr as $method)
        echo "\tfunction $method()\n";
  }
  function class_parentage($obj, $class) {
    global $$obj;
    if (is_subclass_of($$obj, $class)) {
        echo "L'objet $obj belongs to class ".get_class($$obj);
        echo " est une sous-classe de $class\n";
    } else {
        echo "L'objet $obj n'est pas une sous classe $class\n";
    }
  }
// instantie 2 objets
  $legume = new Legume(TRUE,"blue");
  $feuilles = new Epinard();
// affiche les informations sur ces objets
  echo "legume: CLASS ".get_class($legume)."\n";
  echo "feuilles: CLASS ".get_class($feuilles);
  echo ", PARENT ".get_parent_class($feuilles)."\n";
// affiche les propriétés du légume
  echo "\nlégume: Propriétés \n";
  print_vars($legume);
// et les méthodes de "feuilles"
  echo "\nfeuilles: Methods\n";
  print_methods($feuilles);
  echo "\nParentée:\n";
  class_parentage("feuilles", "Epinard");
  class_parentage("feuilles", "Legume");
?>
</pre>
```

Il est important de noter que dans les exemples ci-dessus, les objets `$feuilles` sont une instance de `Epinard` qui est une sous-classe de `Legume`, donc la dernière partie du script va afficher :

```
[...]
Parentée:
L'objet feuilles n'est pas une sous classe Spinach
L'objet feuilles est une sous-classe de Legume
```

call_user_method (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Appelle une méthode utilisateur d'un objet

```
mixed call_user_method  (string method_name, object obj [, mixed parameter [, mixed ...]])
```

Appelle la méthode *method_name* depuis l'objet *obj*. Un exemple d'utilisation de cet objet est présenté ci-dessous, où une classe est définie, puis instantiée. On utilise alors **call_user_method()** pour appeler indirectement les méthodes `print_info`.

```
<?php
class Pays {
    var $NOM;
    var $TLD;
    function Pays($nom, $tld) {
        $this->NOM = $nom;
        $this->TLD = $tld;
    }
    function print_info($prestr="") {
        echo $prestr."Pays: ".$this->NOM."\n";
        echo $prestr."Nom de domaine: ".$this->TLD."\n";
    }
}
$unPays = new Pays("Pérou","pe");
echo "* Appel de la méthode directement\n";
$unPays->print_info();
echo "\n* Appel de la méthode indirectement\n";
call_user_method ("print_info", $unPays, "\t");
?>
```

Voir aussi **call_user_func_array()**. **call_user_func()** et **call_user_method_array()**.

call_user_method_array (PHP 4 >= 4.0.5)

Appelle une méthode utilisateur avec un tableau de paramètres

```
mixed call_user_method_array  (string method_name, object obj [, array paramarr])
```

call_user_method_array() appelle la méthode *method_name* de l'object *obj*, en utilisant les paramètres *paramarr*, rassemblés sous forme de tableau.

Voir aussi **call_user_func_array()**, **call_user_func()** et **call_user_method()**.

> **Note : call_user_method_array()** a été ajoutée en version PHP 4.05.

class_exists (PHP 4 >= 4.0b4)

Vérifie qu'une classe a été définie

```
boolean class_exists (string class_name)
```

class_exists() retourne `TRUE` si la classe *class_name* a été définie, et `FALSE` sinon.

get_class (PHP 4)

Retourne la classe d'un objet

```
string get_class (object obj)
```

get_class() retourne la classe de l'objet *obj*.

Voir aussi **get_parent_class()** et **is_subclass_of()**

get_class_methods (PHP 4 >= 4.0RC1)

Retourne les noms des méthodes d'une classe.

```
array get_class_methods (string class_name)
```

get_class_methods() retourne un tableau contenant les noms des méthodes de la classe *class_name*.

> **Note :** A partir de PHP 4.0.6, vous pouvez spécifier l'objet lui-même, au lieu de sa classe *class_name*. Par exemple :
>
> ```
> <?php
>   $class_methods = get_class_methods($my_class);
> ?>
> ```

Exemple 1. Exemple avec get_class_methods()

```
<?php
class myclass {
    // constructeur
    function maclasse() {
        return(TRUE);
    }
    // méthode 1
    function myfunc1() {
        return(TRUE);
    }
    // méthode 2
    function mafunc2() {
        return(TRUE);
    }
}
$ma_classe = new maclasse();
$class_methods = get_class_methods(get_class($ma_class));
foreach ($class_methods as $method_name) {
    echo "$method_name\n";
}
?>
```

Va afficher :

```
maclass
mafunc1
mafunc2
```

Voir aussi **get_class_vars()** et **get_object_vars()**

get_class_vars (PHP 4 >= 4.0RC1)

Retourne les valeurs par défaut des attributs d'une classe.

```
array get_class_vars (string class_name)
```

get_class_vars() retourne un tableau contenant les valeurs par défaut des attributs de la classe *class_name*.

Note : Les variables de classe qui ne sont pas encore initialisées ne seront pas retournées par **get_class_vars()**.

Exemple 1. Exemple get_class_vars()

```
<?php
class maclasse {
    var $var1; // Pas de valeur par défaut
    var $var2 = "xyz";
    var $var3 = 100;
    // constructeur
    function maclasse() {
        return(TRUE);
    }
}
$ma_classe = new maclasse();
$class_vars = get_class_vars(get_class($ma_classe));
foreach ($class_vars as $name => $value) {
    echo "$name : $value\n";
}
?>
```

va afficher :

```
var2 : xyz
var3 : 100
```

get_declared_classes (PHP 4 >= 4.0RC2)

Liste toutes les classes définies

```
array get_declared_classes (void)
```

get_declared_classes() retourne un tableau contenant la liste des fonctions déclarées dans le script courant.

Note : En PHP 4.0.1pl2, trois classes supplémentaires sont retournées, au début de ce tableau : `stdClass` (définie dans `Zend/zend.c`), `OverloadedTestClass` (définie dans `ext/standard/basic_functions.c`) et `Directory` (définie dans `ext/standard/dir.c`).

get_object_vars (PHP 4 >= 4.0RC1)

Retourne un tableau associatif des propriétés d'un objet

```
array get_object_vars (object obj)
```

get_object_vars() retourne un tableau associatif contenant les propriétés de l'objet *obj*. Les clés du tableau sont les noms des propriétés de l'objet.Si des variables déclarées dans la classe de l'objet *obj*, n'ont pas été assignées, elles ne seront pas retournées dans le tableau.

Exemple 1. Exemple avec get_object_vars()

```
<?php
class Point2D {
    var $x, $y;
    var $nom;
    function Point2D($x, $y) {
        $this->x = $x;
        $this->y = $y;
    }
    function donne_nom($nom) {
        $this->nom = $nom;
    }
    function LitPoint() {
        return array("x" -> $this->x,
                     "y" -> $this->y,
                     "nom" -> $this->nom);
    }
}
$p1 = new Point2D(1.233, 3.445);
print_r(get_object_vars($p1));
// "$nom" est déclaré, mais non défini
// Array
// (
//     [x] -> 1.233
//     [y] -> 3.445
// )
$p1->setnom("point #1");
print_r(get_object_vars($p1));
// Array
// (
//     [x] -> 1.233
//     [y] -> 3.445
//     [nom] -> point #1
// )
?>
```

Voir aussi **get_class_methods()** et **get_class_vars()**

get_parent_class (PHP 4)

Retourne le nom de la classe d'un objet

```
string get_parent_class (object obj)
```

get_parent_class() retourne le nom de la classe de l'objet *obj*.

Voir aussi **get_class()** et **is_subclass_of()**

is_subclass_of (PHP 4 >= 4.0b4)

Détermine si un objet est une sous-classe

```
boolean is_subclass_of (object obj, string superclass)
```

is_subclass_of() retourne TRUE si l'objet *obj* est une sous-classe de *superclass*, FALSE sinon.

Voir aussi **get_class()** et **get_parent_class()**

method_exists (PHP 4)

Vérifie que la méthode existe pour une classe.

```
boolean method_exists (object object, string method_name)
```

method_exists() retourne TRUE si la méthode *method_name* a été définie pour la classe *object*, et sinon, retourne FALSE.

X. ClibPDF

ClibPDF vous permet de créer des documents PDF avec PHP. Cette librairie est disponible à FastIO (http://www.fastio.com/) mais n'est pas gratuite. Vous devez lire la licence avant de l'utiliser. Si vous ne pouvez pas accepter la licence, essayez plutôt pdflib de Thomas Merz, qui est aussi très puissante. Les fonctionnalités de ClibPDF et ses API sont très similaires à celles de Thomas Merz's pdflib mais, selon FastIO, ClibPDF est plus rapide, et crée des documents plus compacts. Cela peut avoir changé depuis la version 2.0 de pdflib. Un test de vitesse (avec pdfclock.c issue des exemples de pdflib 2.0 transformé en script PHP) ne montre aucune différence de vitesse. La taille des fichiers est similaire si la compression n'est pas utilisée. Il vaut mieux alors essayer les deux, et choisir celui qui vous convient le mieux.

Cette documentation devrait être lue avec le manuel ClibPDF sous la main, car il est beaucoup plus détaillé.

Beaucoup de fonctions sont natives de ClibPDF et se retrouvent dans le module PHP, et tout comme pdflib, elles ont le même nom. Toutes les fonctions, hormis **cpdf_open()** utilisent un pointeur sur un document comme premier paramètre. Actuellement, ce pointeur n'est pas utilisé en interne, car ClibPDF ne supporte pas la création de plusieurs documents PDF simultanément. En fait, il ne vaut mieux pas l'envisager, car les résultats sont aléatoires. Je ne veux même pas imaginer les problèmes qui pourrait se poser avec les environnements multi-tâches. Selon l'auteur de ClibPDF, cette situation va changer dans les prochaînes versions (lorsque cette documentation a été traduite, c'était la version 1.10). Si vous avez besoin de cette fonctionnalité, utilisez pdflib.

> **Note :** La fonction **cpdf_set_font()** a changé depuis le PHP 3.0 pour supporter les polices asiatiques. Le paramètre d'encodage n'est plus un entier, mais une chaîne.

Un des gros avantage de ClibPDF sur pdflib est la possibilité de créer complétement un document sans passer par des fichiers temporaires. Il est aussi possible d'utiliser des coordonnées avec une unité de longeur prédéfinie. C'est une fonctionnalité bien pratique mais qui peut être simulée avec **pdf_translate()**.

Un autre atout de ClibPDF est que chaque page peut être modifiée à tout moment même si une nouvelle page a été ouverte. La fonction **cpdf_set_current_page()** vous permet de quitter temporairement une page, et d'en modifier une autre.

La plus part des fonctions sont très simples d'emploi. Le plus difficile est probablement de créer un document PDF simple. L'exemple suivant devrait vous aider à démarrer. La page contient du texte qui utilise la police "Times-Roman" en taille 30, outlined. Le texte est souligné.

Exemple 1. Exemple simple ClibPDF

```
<?php
$cpdf = cpdf_open(0);
cpdf_page_init($cpdf, 1, 0, 595, 842);
cpdf_add_outline($cpdf, 0, 0, 0, 1, "Page 1");
cpdf_set_font($cpdf, "Times-Roman", 30, "WinAnsiEncoding");
cpdf_set_text_rendering($cpdf, 1);
cpdf_text($cpdf, "Times Roman outlined", 50, 750);
cpdf_moveto($cpdf, 50, 740);
cpdf_lineto($cpdf, 330, 740);
cpdf_stroke($cpdf);
cpdf_finalize($cpdf);
Header("Content-type: application/pdf");
cpdf_output_buffer($cpdf);
cpdf_close($cpdf);
?>
```

La distribution pdflib contient un exemple plus complet, qui crée des séries de pages avec une horloge. Voici cet exemple convertit en script PHP qui utilise l'extension ClibPDF :

Exemple 2. Exemple pdfclock de la distribution pdflib 2.0

```
<?php
$radius = 200;
$margin = 20;
$pagecount = 40;
$pdf = cpdf_open(0);
```

```
cpdf_set_creator($pdf, "pdf_clock.php3");
cpdf_set_title($pdf, "Analog Clock");
while($pagecount-- > 0) {
  cpdf_page_init($pdf, $pagecount+1, 0, 2 * ($radius + $margin), 2 * ($radius + $mar-
gin), 1.0);
  cpdf_set_page_animation($pdf, 4, 0.5, 0, 0, 0);  /* wipe */
  cpdf_translate($pdf, $radius + $margin, $radius + $margin);
  cpdf_save($pdf);
  cpdf_setrgbcolor($pdf, 0.0, 0.0, 1.0);
  /* indications des minutes */
  cpdf_setlinewidth($pdf, 2.0);
  for ($alpha = 0; $alpha < 360; $alpha += 6)
    {
    cpdf_rotate($pdf, 6.0);
    cpdf_moveto($pdf, $radius, 0.0);
    cpdf_lineto($pdf, $radius-$margin/3, 0.0);
    cpdf_stroke($pdf);
    }
  cpdf_restore($pdf);
  cpdf_save($pdf);
  /* Indications des 5 minutes */
  cpdf_setlinewidth($pdf, 3.0);
  for ($alpha = 0; $alpha < 360; $alpha += 30)
  {
    cpdf_rotate($pdf, 30.0);
    cpdf_moveto($pdf, $radius, 0.0);
    cpdf_lineto($pdf, $radius-$margin, 0.0);
    cpdf_stroke($pdf);
  }
  $ltime = getdate();
  /* aiguille des heures */
  cpdf_save($pdf);
  cpdf_rotate($pdf, -(($ltime['minutes']/60.0) + $ltime['hours'] - 3.0) * 30.0);
  cpdf_moveto($pdf, -$radius/10, -$radius/20);
  cpdf_lineto($pdf, $radius/2, 0.0);
  cpdf_lineto($pdf, -$radius/10, $radius/20);
  cpdf_closepath($pdf);
  cpdf_fill($pdf);
  cpdf_restore($pdf);
  /* aiguille des minutes */
  cpdf_save($pdf);
  cpdf_rotate($pdf, -(($ltime['seconds']/60.0) + $ltime['minutes'] - 15.0) * 6.0);
  cpdf_moveto($pdf, -$radius/10, -$radius/20);
  cpdf_lineto($pdf, $radius * 0.8, 0.0);
  cpdf_lineto($pdf, -$radius/10, $radius/20);
  cpdf_closepath($pdf);
  cpdf_fill($pdf);
  cpdf_restore($pdf);
  /* aiguille des secondes */
  cpdf_setrgbcolor($pdf, 1.0, 0.0, 0.0);
  cpdf_setlinewidth($pdf, 2);
  cpdf_save($pdf);
  cpdf_rotate($pdf, -(($ltime['seconds'] - 15.0) * 6.0));
  cpdf_moveto($pdf, -$radius/5, 0.0);
  cpdf_lineto($pdf, $radius, 0.0);
  cpdf_stroke($pdf);
  cpdf_restore($pdf);
  /* Un petit cercle au centre */
  cpdf_circle($pdf, 0, 0, $radius/30);
  cpdf_fill($pdf);
  cpdf_restore($pdf);
  cpdf_finalize_page($pdf, $pagecount+1);
}
cpdf_finalize($pdf);
header("Content-type: application/pdf");
cpdf_output_buffer($pdf);
```

```
cpdf_close($pdf);
?>
```

cpdf_global_set_document_limits (PHP 4 >= 4.0b4)

Fixe les limites d'un document PDF.

```
void cpdf_global_set_document_limits  (int maxpages, int maxfonts, int maximages, int
maxannotations, int maxobjects)
```

cpdf_global_set_document_limits() permet de fixer plusieurs limites au document PDF.
cpdf_global_set_document_limits() doit être appelé avant **cpdf_open()** pour être effective. Elle fixe les limites de tous les documents ouverts après.

Voir aussi **cpdf_open()**.

cpdf_set_creator (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Fixe le créateur d'un document PDF.

```
void cpdf_set_creator (string creator)
```

cpdf_set_creator() fixe le créateur d'un document PDF.

Voir aussi **cpdf_set_subject()**, **cpdf_set_title()** et **cpdf_set_keywords()**.

cpdf_set_title (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Fixe le titre d'un document PDF.

```
void cpdf_set_title (string title)
```

cpdf_set_title() fixe le titre d'un document PDF.

Voir aussi **cpdf_set_subject()**, **cpdf_set_creator()** et **cpdf_set_keywords()**.

cpdf_set_subject (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Fixe le sujet d'un document PDF.

```
void cpdf_set_subject (string subject)
```

cpdf_set_subject() fixe le sujet d'un document PDF.

Voir aussi **cpdf_set_title()**, **cpdf_set_creator()** et **cpdf_set_keywords()**.

cpdf_set_keywords (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Fixe les mot clés d'un document PDF.

```
void cpdf_set_keywords (string keywords)
```

cpdf_set_keywords() fixe les mots-clé d'un document PDF.

Voir aussi **cpdf_set_title()**, **cpdf_set_creator()** et **cpdf_set_subject()**.

cpdf_open (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Ouvre un nouveau document PDF.

```
resource cpdf_open (int compression, string filename)
```

cpdf_open() ouvre un nouveau document PDF. Le premier paramètre *compression* active ou pas la compression, suivant qu'il vaut 0 ou 1. Le deuxième paramètre, optionnel, choisit le fichier de destination du document. Si il est omis, le document sera écrit en mémoire, et pourra être écrit dans un fichier avec **cpdf_save_to_file()** ou envoyé à l'affichage avec **cpdf_output_buffer()**.

> **Note :** La valeur retournée sera nécessaire pour les autres fonctions de ClibPDF comme premier paramètre.
>
> La librairie ClibPDF prend le nom de fichier "-" comme synonyme de stdout. Si PHP est compilé comme un module apache, cela ne fonctionnera pas, car la méthode d'envoie des données de ClibPDF ne fonctionne pas avec Apache. Vous pouvez résoudre ce problème en ne fournissant pas de nom de fichier, et en utilisant la fonction **cpdf_output_buffer()** pour afficher le document PDF.

Voir aussi **cpdf_close()** et **cpdf_output_buffer()**.

cpdf_close (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Ferme un fichier PDF.

```
void cpdf_close (resource pdf_document)
```

cpdf_close() ferme un fichier PDF. Ce doit être la dernière fonction appelée, et elle apparaît même après **cpdf_finalize()**, **cpdf_output_buffer()** et **cpdf_save_to_file()**.

Voir aussi **cpdf_open()**.

cpdf_page_init (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Commence une nouvelle page.

```
void cpdf_page_init (resource pdf_document, int page_number, int orientation, double
height, double width [, double unit])
```

cpdf_page_init() commence une nouvelle page, avec la hauteur *height* et la largeur *width*. La page a le numéro *page_number* et l'orientation *orientation*. *orientation* vaut 0 pour portrait et 1 pour paysage. Le dernier paramètre, optionnel, *unit*, fixe l'unité pour le système de coordonnées. Cette valeur doit être un nombre de points postscript, par unité. Etant donné que un pouce (inch) vaut 72 points, une valeur de 72 vaudra un pouce (inch). Par défaut, cette valeur vaut 72.

Voir aussi **cpdf_set_current_page()**.

cpdf_finalize_page (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Termine une page.

```
void cpdf_finalize_page (resource pdf_document, int page_number)
```

cpdf_finalize_page() termine la page de numéro *page_number*. **cpdf_finalize_page()** ne fait qu'une sauvegarde mémoire. Les pages terminées prennent moins de place, mais ne peuvent plus être modifiées.

Voir aussi **cpdf_page_init()**.

cpdf_finalize (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Termine un document.

```
void cpdf_finalize (resource pdf_document)
```

cpdf_finalize() termine un document. Vous devez toujours appeler **cpdf_close()** après.

Voir aussi **cpdf_close()**.

cpdf_output_buffer (PHP 3>= 3.0.9, PHP 4 >= 4.0b4)

Envoie le document PDF dans un buffer mémoire.

```
void cpdf_output_buffer (resource pdf_document)
```

cpdf_output_buffer() envoie le document PDF dans un buffer mémoire de stdout. Le document doit avoir été créé en mémoire, ce qui est le cas si **cpdf_open()** a été appelée dans paramètre de nom de fichier.

Voir aussi **cpdf_open()**.

cpdf_save_to_file (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Ecrit un document PDF dans un fichier.

```
void cpdf_save_to_file (resource pdf_document, string filename)
```

cpdf_save_to_file() écrit un document PDF dans un fichier, s'il a été créé en mémoire. **cpdf_save_to_file()** n'est pas nécessaire si un nom de fichier a été fourni lors de l'appel à **cpdf_open()**.

Voir aussi **cpdf_output_buffer()** et **cpdf_open()**.

cpdf_set_current_page (PHP 3>= 3.0.9, PHP 4 >= 4.0b4)

Fixe la page courante.

```
void cpdf_set_current_page (resource pdf_document, int page_number)
```

cpdf_set_current_page() fixe la page courante, oú toutes les prochaînes opérations vont avoir lieu. On peut changer de page jusqu'à ce qu'une page soit terminée avec **cpdf_finalize_page()**.

Voir aussi **cpdf_finalize_page()**.

cpdf_begin_text (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Démarre une section de texte.

```
void cpdf_begin_text (resource pdf_document)
```

cpdf_begin_text() démarre une section de texte. Elle doit être terminée avec **cpdf_end_text()**.

Exemple 1. Affichage de texte

```
<?php
cpdf_begin_text($pdf);
cpdf_set_font($pdf, 16, "Helvetica", "WinAnsiEncoding");
cpdf_text($pdf, 100, 100, "Some text");
cpdf_end_text($pdf)
?>
```

Voir aussi **cpdf_end_text()**.

cpdf_end_text (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Termine une section de texte.

```
void cpdf_end_text (resource pdf_document)
```

cpdf_end_text() termine une section de texte, commencée avec **cpdf_begin_text()**.

Exemple 1. Affichage de texte

```
<?php
cpdf_begin_text($pdf);
cpdf_set_font($pdf, 16, "Helvetica", "WinAnsiEncoding");
cpdf_text($pdf, 100, 100, "Some text");
cpdf_end_text($pdf)
?>
```

Voir aussi **cpdf_begin_text()**.

cpdf_show (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Imprime un texte à la position courante.

```
void cpdf_show (resource pdf_document, string text)
```

cpdf_show() imprime la chaîne *text*, à la position courante.

Voir aussi **cpdf_text()**, **cpdf_begin_text()** et **cpdf_end_text()**.

cpdf_show_xy (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Affiche un texte à une position.

```
void cpdf_show_xy (resource pdf_document, string text, double x-coor, double y-coor [, int mode])
```

cpdf_show_xy() imprime la chaîne *text*, à la position de coordonnées (*x-koor*, *y-koor*). Le dernier paramètre optionnel est une unité de longueur. Si il prend la valeur de 0 (ou si il est omis), c'est la valeur par défaut (72) qui est utilisé.

> **Note :** The function **cpdf_show_xy()** est identique à **cpdf_text()** sans les options.

Voir aussi **cpdf_text()**.

cpdf_text (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Imprime un texte avec des options.

```
void cpdf_text (resource pdf_document, string text, double x-coor, double y-coor [, int mode [, double orientation [, int alignmode]]])
```

cpdf_text() imprime le text *text* à la position de coordonnées (*x-koor*, *y-koor*). Le paramètre *mode* est une unité de longueur. Si il prend la valeur de 0 (ou si il est omis), c'est la valeur par défaut (72) qui est utilisé.

Le paramètre optionel *orientation* est un angle de rotation du texte, en degrés. Le paramètre optionnel *alignmode* détermine l'alignement du texte. Reportez vous à la doc de ClibPDF, pour les valeurs possibles.

Voir aussi **cpdf_show_xy()**.

cpdf_set_font (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Sélectionne la police courante et sa taille.

```
void cpdf_set_font (resource pdf_document, string font name, double size, string encoding)
```

cpdf_set_font() selectionne la police courante, sa taille et l'encodage. Actuellement, seules les polices postscript sont supportées.

Le dernier paramètre *encoding* peut prendre les valeurs suivantes : "MacRomanEncoding", "MacExpertEncoding", "WinAnsiEncoding", et "NULL". "NULL" signifie qu'il faut utiliser l'encodage par défaut.

Reportez vous à la doc de ClibPDF, pour plus d'informations, notamment sur les polices asiatiques.

cpdf_set_leading (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Fixe la distance entre deux lignes.

```
void cpdf_set_leading (resource pdf_document, double distance)
```

cpdf_set_leading() fixe la distance entre deux lignes. Cela servira si le texte est affiché par **cpdf_continue_text()**.

Voir aussi **cpdf_continue_text()**.

cpdf_set_text_rendering (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Détermine le rendu du texte.

```
void cpdf_set_text_rendering (resource pdf_document, int mode)
```

cpdf_set_text_rendering() détermines le rendu du texte.

Les valeurs possibles pour *mode* sont : 0=texte plein, 1=texte stroke, 2=texte plein et stroke, 3=invisible, 4=texte plein et ajouté au chemin, 5=texte stroke et ajouté au chemin, 6=texte plein et stroke et ajouté au chemin, 7=et ajouté au chemin.

cpdf_set_horiz_scaling (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Fixe l'échelle horizontale du texte.

```
void cpdf_set_horiz_scaling (resource pdf_document, double scale)
```

cpdf_set_horiz_scaling() fixe l'échelle horizontale du texte à *scale* %.

cpdf_set_text_rise (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Fixe l'élévation du texte.

```
void cpdf_set_text_rise (resource pdf_document, double value)
```

cpdf_set_text_rise() fixe l'élévation du texte à *value* unités.

cpdf_set_text_matrix (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Fixe la matrice du texte.

```
void cpdf_set_text_matrix (resource pdf_document, array matrix)
```

cpdf_set_text_matrix() fixe la matrice du texte, qui décrit la transformation appliquée à police.

cpdf_set_text_pos (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Fixe la position du texte.

```
void cpdf_set_text_pos (resource pdf_document, double x-koor, double y-koor, int mode)
```

cpdf_set_text_pos() Fixe la position du texte pour le prochain appel à **cpdf_show()**.

Le paramètre *mode* est une unité de longueur. Si il prend la valeur de 0 (ou si il est omis), c'est la valeur par défaut (72) qui est utilisé.

Voir aussi **cpdf_show()**, **cpdf_text()**.

cpdf_set_char_spacing (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Fixe l'espacement des caractères.

```
void cpdf_set_text_pos (resource pdf_document, double space)
```

cpdf_set_char_spacing() fixe l'espacement des caractères.

Voir aussi **cpdf_set_word_spacing()**, **cpdf_set_leading()**.

cpdf_set_word_spacing (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Fixe l'espacement des mots.

```
void cpdf_set_word_spacing (resource pdf_document, double space)
```

cpdf_set_word_spacing() fixe l'espacement des caractères.

Voir aussi **cpdf_set_char_spacing()**, **cpdf_set_leading()**.

cpdf_continue_text (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Imprime le texte à la ligne suivante.

```
void cpdf_continue_text (resource pdf_document, string text)
```

cpdf_continue_text() imprime le texte *text* à la ligne suivante.

Voir aussi **cpdf_show_xy()**, **cpdf_text()**, **cpdf_set_leading()**, **cpdf_set_text_pos()**.

cpdf_stringwidth (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Retourne la taille de la chaîne.

```
double cpdf_stringwidth (resource pdf_document, string text)
```

cpdf_stringwidth() retourne la taille de la chaîne *text*. Une police doit avoir déjà été choisie.

Voir aussi **cpdf_set_font()**.

cpdf_save (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Sauve l'environnement courant.

```
void cpdf_save (resource pdf_document)
```

cpdf_save() sauve l'environnement courant. **cpdf_save()** est similaire à la commande postscript gsave. Très pratique quand vous devez faire des translations et rotations sur un objet, mais sans affecter les autres.

Voir aussi **cpdf_restore()**.

cpdf_restore (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Restaure un environnement.

```
void cpdf_restore (resource pdf_document)
```

cpdf_restore() restaure l'environnement sauvé par **cpdf_save()**. Cette fonction est similaire à la commande postscript grestore. Très pratique quand vous devez faire des translations et rotations sur un objet, mais sans affecter les autres.

Exemple 1. Sauver/Restaurer

```
<?php
cpdf_save($pdf);
// plein de transformations, translations, ...
cpdf_restore($pdf)
?>
```

Voir aussi **cpdf_save()**.

cpdf_translate (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Modifie l'origine du système de coordonées.

```
void cpdf_translate (resource pdf_document, double x-koor, double y-koor, int mode)
```

cpdf_translate() modifie l'origine du système de coordonées en placant l'origine aux coordonnées (*x-koor*, *y-koor*).

Le paramètre *mode* est une unité de longueur. S'il prend la valeur de 0 (ou s'il est omis), c'est la valeur par défaut (72) qui est utilisé.

cpdf_scale (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Modifie l'échelle.

```
void cpdf_scale (resource pdf_document, double x-scale, double y-scale)
```

cpdf_scale() modifie l'echelle dans les deux directions.

cpdf_rotate (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Effectue une rotation.

```
void cpdf_rotate (resource pdf_document, double angle)
```

cpdf_rotate() effectue une rotation, d'un angle de *angle* degrés.

cpdf_setflat (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Fixe la platitude (flatness).

```
void cpdf_setflat (resource pdf_document, double value)
```

cpdf_setflat() fixe la platitude (flatness), entre 0 et 100.

cpdf_setlinejoin (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Fixe le paramètre linejoin.

```
void cpdf_setlinejoin (resource pdf_document, long value)
```

cpdf_setlinejoin() fixe le paramètre linejoin à une valeur *value*, entre 0 et 2. 0 = miter, 1 = round, 2 = bevel.

cpdf_setlinecap (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Fixe le paramètre linecap.

```
void cpdf_setlinecap (resource pdf_document, int value)
```

cpdf_setlinecap() fixe le paramètre linecap à une valeur *value* entre 0 et 2. 0 = butt end, 1 = round, 2 = projecting square.

cpdf_setmiterlimit (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Fixe le paramètre miter limit.

```
void cpdf_setmiterlimit (resource pdf_document, double value)
```

cpdf_setmiterlimit() fixe le paramètre "miter limit" à une valeur supérieure ou égale à 1.

cpdf_setlinewidth (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Fixe la largeur de ligne.

```
void cpdf_setlinewidth (resource pdf_document, double width)
```

cpdf_setlinewidth() fixe la largeur de ligne à la valeur de *width*.

cpdf_setdash (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Fixe le motif de pointillé.

```
void cpdf_setdash (resource pdf_document, double white, double black)
```

cpdf_setdash() fixe le motif de pointillé à *white* unité de blanc et *black* unités de noir. Si les deux sont à 0, une ligne pleine est affichée.

cpdf_newpath (PHP 3>= 3.0.9, PHP 4 >= 4.0b4)

Commence un nouveau chemin

```
void cpdf_newpath  (int pdf_document)
```

cpdf_newpath() commence un nouveau chemin dans le document *pdf_document*.

cpdf_moveto (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Fixe le point courant.

```
void cpdf_moveto (resource pdf_document, double x-koor, double y-koor, int mode)
```

cpdf_moveto() fixe le point courant aux coordonnées (*x-koor*, *y-koor*).

Le paramètre *mode* est une unité de longueur. Si il prend la valeur de 0 (ou si il est omis), c'est la valeur par défaut (72) qui est utilisée.

cpdf_rmoveto (PHP 3>= 3.0.9, PHP 4 >= 4.0b4)

Fixe le point courant relativement.

```
void cpdf_rmoveto (resource pdf_document, double x-koor, double y-koor, int mode)
```

cpdf_rmoveto() fixe le point courant aux coordonnées (*x-koor*, *y-koor*), relativement.

Le paramètre *mode* est une unité de longueur. Si il prend la valeur de 0 (ou si il est omis), c'est la valeur par défaut (72) qui est utilisée.

Voir aussi **cpdf_moveto()**.

cpdf_curveto (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Dessine une courbe.

```
void cpdf_curveto (resource pdf_document, double x1, double y1, double x2, double y2,
double x3, double y3, int mode)
```

cpdf_curveto() dessine une courbe de Bezier, entre le point courant et le point (*x3*, *y3*), en utilisant les points de contrôle (*x1*, *y1*) et (*x2*, *y2*).

Le paramètre *mode* est une unité de longueur. Si il prend la valeur de 0 (ou si il est omis), c'est la valeur par défaut (72) qui est utilisée.

Voir aussi **cpdf_moveto()**, **cpdf_rmoveto()**, **cpdf_rlineto()**, **cpdf_lineto()**.

cpdf_lineto (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Dessine une ligne.

```
void cpdf_lineto (resource pdf_document, double x-koor, double y-koor, int mode)
```

cpdf_lineto() dessine une ligne entre le point courant et le point de coordonnées (*x-koor*, *y-koor*).

Le paramètre *mode* est une unité de longueur. Si il prend la valeur de 0 (ou si il est omis), c'est la valeur par défaut (72) qui est utilisée.

Voir aussi **cpdf_moveto()**, **cpdf_rmoveto()**, **cpdf_curveto()**.

cpdf_rlineto (PHP 3>= 3.0.9, PHP 4 >= 4.0b4)

Dessine une ligne, relativement.

```
void cpdf_rlineto (resource pdf_document, double x-koor, double y-koor, int mode)
```

cpdf_rlineto() dessine une ligne entre le point courant et le point de coordonnées relatives (*x-koor*, *y-koor*).

Le paramètre *mode* est une unité de longueur. Si il prend la valeur de 0 (ou si il est omis), c'est la valeur par défaut (72) qui est utilisée.

Voir aussi **cpdf_moveto()**, **cpdf_rmoveto()** et **cpdf_curveto()**.

cpdf_circle (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Dessine un cercle.

```
void cpdf_circle (resource pdf_document, double x-koor, double y-koor, double radius, int
mode)
```

cpdf_circle() dessine un cercle de centre (*x-koor*, *y-koor*) et de rayon *radius*.

Le paramètre *mode* est une unité de longueur. Si il prend la valeur de 0 (ou si il est omis), c'est la valeur par défaut (72) qui est utilisée.

Voir aussi **cpdf_arc()**.

cpdf_arc (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Dessine un arc de cercle.

```
void cpdf_arc (resource pdf_document, double x-koor, double y-koor, double radius, double
start, double end, int mode)
```

cpdf_arc() dessine un arc de cercle, dont le centre est au point (*x-koor*, *y-koor*) et l'angle est *radius*, commencant à l'angle *start* et finissant à l'angle *end*.

Le paramètre *mode* est une unité de longueur. Si il prend la valeur de 0 (ou si il est omis), c'est la valeur par défaut (72) qui est utilisée.

Voir aussi **cpdf_circle()**.

cpdf_rect (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Dessine un rectangle.

```
void cpdf_rect (resource pdf_document, double x-koor, double y-koor, double width, double height, int mode)
```

cpdf_rect() dessine un rectangle dont le coin inférieur droit est au point (*x-koor*, *y-koor*). La largeur est *width*. La hauteur est*height*.

Le paramètre *mode* est une unité de longueur. Si il prend la valeur de 0 (ou si il est omis), c'est la valeur par défaut (72) qui est utilisée.

cpdf_closepath (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Ferme le chemin.

```
void cpdf_closepath (resource pdf_document)
```

cpdf_closepath() ferme le chemin courant.

cpdf_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Dessine une ligne le long du chemin.

```
void cpdf_stroke (resource pdf_document)
```

cpdf_stroke() dessine une ligne le long du chemin.

Voir aussi **cpdf_closepath()**, **cpdf_closepath_stroke()**.

cpdf_closepath_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Ferme le fichier et dessine une ligne le long du chemin.

```
void cpdf_closepath_stroke (resource pdf_document)
```

cpdf_closepath_stroke() est une combinaison de **cpdf_closepath()** et **cpdf_stroke()**.

Voir aussi **cpdf_closepath()**, **cpdf_stroke()**.

cpdf_fill (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Remplit le chemin courant.

```
void cpdf_fill (resource pdf_document)
```

cpdf_fill() remplit l'intérieur du chemin courant avec la couleur courante.

Voir aussi **cpdf_closepath()**, **cpdf_stroke()**, **cpdf_setgray_fill()**, **cpdf_setgray()**, **cpdf_setrgbcolor_fill()** et **cpdf_setrgbcolor()**.

cpdf_fill_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Remplit le chemin, et dessine le bord.

```
void cpdf_fill_stroke (resource pdf_document)
```

cpdf_fill_stroke() remplit l'intérieur du chemin avec la couleur courante, et dessine le chemin.

Voir aussi **cpdf_closepath()**, **cpdf_stroke()**, **cpdf_fill()**, **cpdf_setgray_fill()**, **cpdf_setgray()**, **cpdf_setrgbcolor_fill()** et **cpdf_setrgbcolor()**.

cpdf_closepath_fill_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Remplit le chemin, dessine le bord et ferme le chemin.

```
void cpdf_closepath_fill_stroke (resource pdf_document)
```

cpdf_closepath_fill_stroke() remplit le chemin, dessine le bord et ferme le chemin.

Voir aussi **cpdf_closepath()**, **cpdf_stroke()**, **cpdf_fill()**, **cpdf_setgray_fill()**, **cpdf_setgray()**, **cpdf_setrgbcolor_fill()** et **cpdf_setrgbcolor()**.

cpdf_clip (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Aligne les dessins sur le chemin courant.

```
void cpdf_clip (resource pdf_document)
```

cpdf_clip() aligne les dessins sur le chemin courant.

cpdf_setgray_fill (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Modifie le niveau de gris comme couleur de remplissage.

```
void cpdf_setgray_fill (resource pdf_document, double value)
```

cpdf_setgray_fill() remplace le niveau de gris, couleur de remplissage courante, par *value*.

Voir aussi **cpdf_setrgbcolor_fill()**.

cpdf_setgray_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Choisit un niveau de gris comme couleur de dessin.

```
void cpdf_setgray_stroke (resource pdf_document, double gray_value)
```

cpdf_setgray_stroke() remplace le niveau de gris, couleur de dessin courante, par *value*.

Voir aussi **cpdf_setrgbcolor_stroke()**.

cpdf_setgray (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Modifie un niveau de gris comme couleur de dessin et de remplissage.

```
void cpdf_setgray (resource pdf_document, double gray_value)
```

cpdf_setgray_stroke() remplace le niveau de gris, couleur de dessin et de remplissage, par *value*.

Voir aussi **cpdf_setrgbcolor_stroke()**, **cpdf_setrgbcolor_fill()**.

cpdf_setrgbcolor_fill (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Choisit une couleur rgb comme couleur de remplissage.

```
void cpdf_setrgbcolor_fill (resource pdf_document, double red_value, double green_value,
double blue_value)
```

cpdf_setrgbcolor_fill() remplace la couleur de remplissage, par la couleur rgb (*red_value*, *green_value*, *blue_value*).

Voir aussi **cpdf_setrgbcolor_stroke()**, **cpdf_setrgbcolor()**.

cpdf_setrgbcolor_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Choisit une couleur rgb comme couleur de dessin.

```
void cpdf_setrgbcolor_stroke (resource pdf_document, double red_value, double green_value,
double blue_value)
```

cpdf_setrgbcolor_stroke() remplace la couleur de dessin, par la couleur rgb (*red_value*, *green_value*, *blue_value*).

Voir aussi **cpdf_setrgbcolor_fill()** et **cpdf_setrgbcolor()**.

cpdf_setrgbcolor (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Choisit une couleur rgb comme couleur de dessin et de remplissage.

```
void cpdf_setrgbcolor (resource pdf_document, double red_value, double green_value, double
blue_value)
```

cpdf_setrgbcolor_stroke() remplace la couleur de remplissage et de dessin, par la couleur rgb (*red_value*, *green_value*, *blue_value*).

Voir aussi **cpdf_setrgbcolor_stroke()** et **cpdf_setrgbcolor_fill()**.

cpdf_add_outline (PHP 3>= 3.0.9, PHP 4 >= 4.0b4)

Ajoute un signet à la page courante.

```
void cpdf_add_outline (resource pdf_document, string text)
```

cpdf_add_outline() ajoute un signet à la page courante, avec le texte *text* qui pointe sur la page courante.

Exemple 1. Ajouter une mise en relief

```
<?php
$cpdf = cpdf_open(0);
cpdf_page_init($cpdf, 1, 0, 595, 842);
cpdf_add_outline($cpdf, 0, 0, 0, 1, "Page 1");
// ...
// quelques dessins
// ...
cpdf_finalize($cpdf);
Header("Content-type: application/pdf");
cpdf_output_buffer($cpdf);
cpdf_close($cpdf);
?>
```

cpdf_set_page_animation (PHP 3>= 3.0.9, PHP 4 >= 4.0b4)

Fixe l'animation de la transition entre les pages.

```
void cpdf_set_page_animation (resource pdf_document, int transition, double duration)
```

cpdf_set_page_animation() fixe l'animation de la transition entre les pages.

La valeur du paramètre de transition *transition* peut être :

0 pour aucune,
1 pour deux lignes en travers de l'écran, qui révèlent la prochaîne page,
2 pour plusieurs lignes en travers de l'écran, qui révèlent la prochaîne page,
3 pour une boîte qui révèle la prochaîne page,
4 pour une seule ligne en travers de l'écran, qui révèle la prochaîne page,
5 pour l'ancienne page qui se dissout
6 pour un effet de dissolution d'un angle à l'autre
7 pour le remplacement simple (par défaut)

La valeur de *duration* est le nombre de secondes avant le passage à la page suivante.

cpdf_import_jpeg (PHP 3>= 3.0.9, PHP 4 >= 4.0b4)

Ouvre une image JPEG.

```
int cpdf_import_jpeg (resource pdf_document, string file name, double x-koor, double
y-koor, double angle, double width, double height, double x-scale, double y-scale, int
mode)
```

cpdf_import_jpeg() ouvre une image JPG, enregistré dans le fichier *file name*. Le format de l'image doit être JPEG. L'image est placée dans la page courante, aux coordonnées (*x-koor*, *y-koor*). L'image subira une rotation d'un angle de *angle* degrés.

Le paramètre *mode* est une unité de longueur. Si il prend la valeur de 0 (ou si il est omis), c'est la valeur par défaut (72) qui est utilisée.

Voir aussi **cpdf_place_inline_image()**.

cpdf_place_inline_image (PHP 3>= 3.0.9, PHP 4 >= 4.0b4)

Place une image dans la page.

```
void cpdf_place_inline_image (resource pdf_document, int image, double x-koor, double
y-koor, double angle, double width, double height, int mode)
```

cpdf_place_inline_image() places une image créée par un script PHP, dans la page, à la position (*x-koor*, *y-koor*). L'image peut être mise à l'échelle, en même temps.

Le paramètre *mode* est une unité de longueur. Si il prend la valeur de 0 (ou si il est omis), c'est la valeur par défaut (72) qui est utilisé.

Voir aussi **cpdf_import_jpeg()**.

cpdf_add_annotation (PHP 3>= 3.0.12, PHP 4 >= 4.0b4)

Ajoute une annotation.

```
void cpdf_add_annotation (resource pdf_document, double llx, double lly, double urx,
double ury, string title, string content, int mode)
```

cpdf_add_annotation() ajoute une note, dont le coin inférieur droit est (*llx*, *lly*) et le coin supérieur droit est (*urx*, *ury*).

Le paramètre *mode* est une unité de longueur. Si il prend la valeur de 0 (ou si il est omis), c'est la valeur par défaut (72) qui est utilisé.

XI. CURL

Introduction à CURL

PHP supporte libcurl, une librairie créée par Daniel Stenberg, qui vous permet de vous connecter de communiquer avec de nombreux serveurs, grâce à de nombreux protocoles. libcurl supporte actuellement les protocoles suivants : http, https, ftp, gopher, telnet, dict, file, et ldap. libcurl supporte aussi les certificats HTTPS, les POST HTTP, PUT HTTP, le chargement par FTP (ce qui peut être fait par l'extension FTP), les chargement par formulaire HTTP, les proxies, les cookies et l'authentification par mot de passe et nom de compte.

Pour pouvoir utiliser les fonctions CURL, vous devez installer le package CURL (http://curl.haxx.se/). PHP requiert la version CURL 7.0.2-beta ou plus récente. PHP ne fonctionnera pas avec une version inférieure à la version 7.0.2-beta.

Pour utiliser CURL depuis les scripts PHP, vous devez aussi compiler PHP avec l'option `--with-curl[=DIR]` où DIR est le chemin jusqu'au dossier contenant les dossiers `lib` et `include`. Dans le dossier `include` il doit se trouver un dossier appelé `curl`, qui contient notamment les fichiers `easy.h` et `curl.h`. Il doit aussi se trouver un fichier nommé `libcurl.a` dans le dossier `lib`.

Une fois que vous avez compilé PHP avec le support CURL, vous pouvez commencer à l'exploiter avec vos scripts PHP. Le principe de fonctionnement est d'initialiser une session CURL avec **curl_init()**, puis de choisir toutes vos options de transfert avec **curl_exec()** et de finir votre session avec **curl_close()**. Voici un exemple d'utilisation des fonctions CURL, qui récupère la page principale de PHP :

Exemple 1. Utilisation de CURL et PHP pour récupérer une page

```
<?php
$ch = curl_init ("http://www.php.net/");
$fp = fopen ("php_homepage.txt", "w");
curl_setopt ($ch, CURLOPT_INFILE, $fp);
curl_setopt ($ch, CURLOPT_HEADER, 0);
curl_exec ($ch);
curl_close ($ch);
fclose ($fp);
?>
```

curl_init (PHP 4 >= 4.0.2)

Initialise une session CURL

```
resource curl_init  (string [url])
```

curl_init() initialise une nouvelle session et retourne un identifiant de session CURL, à utiliser avec les fonctions **curl_setopt()**, **curl_exec()** et **curl_close()**. Si le paramètre optionnel *url* est fourni, alors CURLOPT_URL prendra cette valeur. Vous pouvez manuellement fixer cette valeur avec la fonction **curl_setopt()**.

Exemple 1. Initialiser une session CURL et récupèration d'une page web.

```
<?php
$ch = curl_init();
curl_setopt ($ch, CURLOPT_URL, "http://www.zend.com/");
curl_setopt ($ch, CURLOPT_HEADER, 0);
curl_exec ($ch);
curl_close ($ch);
?>
```

Voir aussi : **curl_close()**, **curl_setopt()**.

curl_init (PHP 4 >= 4.0.2)

Modifie une option de transfert CURL

```
boolean curl_setopt  (resource ch, string option, mixed value)
```

curl_setopt() fixe les options de transfert de la session CURL identifiée par *ch*. *option* est le nom de l'option à fixer, et *value* est sa valeur.

value doit être de type "long" pour les options suivantes (spécifiée par *option*) :

- *CURLOPT_INFILESIZE*: Lorsque vous téléchargez un fichier sur un site distant, cette option sert à indiquer à PHP la taille maximale du fichier attendu.
- *CURLOPT_VERBOSE*: Choisissez une valeur non nulle pour que CURL vous affiche tous les événements.
- *CURLOPT_HEADER*: Choisissez une valeur non nulle pour que CURL inclut l'en-tête dans la valeur de retour.
- *CURLOPT_NOPROGRESS*: Choisissez une valeur non nulle pour que PHP n'affiche pas l'état des transferts CURL.

  Note : PHP choisit automatiquement une valeur non nulle. Ne changez cette valeur que le temps du débuggage.

- *CURLOPT_NOBODY*: Choisissez une valeur non nulle pour que le corps du transfert ne soit pas inclus dans la valeur de retour.
- *CURLOPT_FAILONERROR*: Choisissez une valeur non nulle pour que PHP traite silencieusement les codes HTTP supérieurs à 300. Le comportement par défaut est de retourner la page normalement, en ignorant ce code.
- *CURLOPT_UPLOAD*: Choisissez une valeur non nulle pour que PHP prépare un chargement.
- *CURLOPT_POST*: Choisissez une valeur non nulle pour que PHP fasse un HTTP POST. Un POST est un encodage normal "application/x-www-from-url", utilisé couramment par les formulaires HTML.
- *CURLOPT_FTPLISTONLY*: Choisissez une valeur non nulle pour que PHP ne fasse que lister les noms d'un dossier FTP.
- *CURLOPT_FTPAPPEND*: Choisissez une valeur non nulle pour que PHP concatène le fichier distant, plutôt que de l'écraser.

- *CURLOPT_NETRC*: Choisissez une valeur non nulle pour que PHP scanne votre fichier ~./netrc et utilise votre nom de compte et mot de passe sur le site distant que vous souhaitez contacter.
- *CURLOPT_FOLLOWLOCATION*: Choisissez une valeur non nulle pour suivre toutes les en-têtes "Location: " que le serveur envoie dans les en-têtes HTTP (notez que cette fonction est récursive, et que PHP suivra toutes les en-têtes "Location: " qu'il trouvera).
- *CURLOPT_PUT*: Choisissez une valeur non nulle pour que pour chargement se fasse par HTTP PUT. Le fichier à charger doit être fixé avec les options CURLOPT_INFILE et CURLOPT_INFILESIZE.
- *CURLOPT_MUTE*: Choisissez une valeur non nulle pour que PHP soit totalement silencieux concernant toutes les fonctions CURL.
- *CURLOPT_TIMEOUT*: Passez un entier "long" comme paramètre qui représente le temps maximum d'exécution de la fonction CURL.
- *CURLOPT_LOW_SPEED_LIMIT*: Passez un entier long qui représente la vitesse minimale en octets par secondes en dessous de laquelle, et pendant CURLOPT_LOW_SPEED secondes, PHP considèrera qu'elle est trop lente, et annulera le transfert.
- *CURLOPT_LOW_SPEED_TIME*: Passez un entier "long" qui représente le temps en secondes, qui, si la vitesse de transfert reste en dessous de CURLOPT_LOW_SPEED_LIMIT, PHP considèrera que la connexion est trop lente, et l'annulera.
- *CURLOPT_RESUME_FROM*: Passez un entier "long", qui représente l'offset, en octets, à partir duquel vous voulez commencer le transfert.
- *CURLOPT_SSLVERSION*: Passez un entier "long" qui contient la version de SSL (2 ou 3) à utiliser. Par défaut, PHP essaiera de le déterminer par lui-même, bien que dans certains cas, il vous faudra le faire manuellement.
- *CURLOPT_TIMECONDITION*: Passez un entier "long" qui définit comment CURLOPT_TIMEVALUE est utilisé. Vous pouvez choisir entre les valeurs TIMECOND_IFMODSINCE ou TIMECOND_ISUNMODSINCE. C'est une fonctionnalité HTTP.
- *CURLOPT_TIMEVALUE*: Passez un entier "long" qui représente le temps en secondes depuis le 1er janvier 1970. Cette valeur sera utilisée comme spécifié dans l'option CURLOPT_TIMEVALUE. Par défaut, TIMECOND_IFMODSINCE sera utilisé.

value doit être une chaîne de caractères pour les valeurs suivantes de *option*

- *CURLOPT_URL*: L'URL que PHP va récupérer. Vous pouvez aussi choisir cette valeur lors de l'appel à **curl_init()**.
- *CURLOPT_USERPWD*: Passez une chaîne de caractères au format [nom]:[mot de passe], pour que PHP l'utilise lors de la connexion.
- *CURLOPT_PROXYUSERPWD*: Passez une chaîne de caractères au format [nom]:[mot de passe], pour que PHP l'utilise lors de la connexion à un proxy HTTP.
- *CURLOPT_RANGE*: Passez une chaîne de caractères qui représente la plage de valeur que vous désirez. Elle est au format "X-Y", où les valeurs de X ou Y peuvent être omises. Le transfert HTTP supporte aussi plusieurs intervalles, séparé par des virgules : X-Y,N-M.
- *CURLOPT_POSTFIELDS*: Passez une chaîne de caractères qui contient toutes les données à passer lors d'une opération de HTTP POST.
- *CURLOPT_REFERER*: Passez une chaîne de caractères qui contient l'en-tête de "REFERER", utilisé lors d'une requête HTTP.
- *CURLOPT_USERAGENT*: Passez une chaîne de caractères qui contient l'en-tête "user-agent" utilisé dans une requête HTTP.
- *CURLOPT_FTPPORT*: Passez une chaîne de caractères qui désignera l'adresse IP utilisée pour l'instruction FTP "PORT". L'instruction POST indique au serveur distant de se connecter cette adresse IP. La chaîne peut être une adresse IP, un nom d'hôte, un nom d'interface réseau (sous UNIX), ou juste '-', pour utiliser les IP par défaut du système.
- *CURLOPT_COOKIE*: Passez une chaîne de caractères qui contiendra le contenu du cookie, à transmettre dans l'en-tête HTTP.
- *CURLOPT_SSLCERT*: Passez une chaîne de caractères qui contiendra le nom de fichier du certificat, au format PEM.

- *CURLOPT_SSLCERTPASSWD*: Passez une chaîne de caractères qui contient le mot de passe nécessaire pour utiliser le certificat CURLOPT_SSLCERT.
- *CURLOPT_COOKIEFILE*: Passez une chaîne de caractères qui contiendra le nom du fichier contenant les données de cookie. Le fichier de cookie peut être au format Netscape, ou simplement des en-têtes HTTP écrites dans un fichier.
- *CURLOPT_CUSTOMREQUEST*: Passez une chaîne de caractères qui sera utilisé à la place de GET ou HEAD lors des requêtes HTTP. Cette commande est pratique pour effectuer un DELETE, ou une autre commande HTTP exotique.

  Note : N'utilisez pas cette commande sans vous assurer que le serveur l'accepte.

Les options suivantes requièrent un pointeur de fichier, qui est obtenu avec la fonction **fopen()** :

- *CURLOPT_FILE*: Le fichier de sortie de votre transfert. Par défaut, STDOUT.
- *CURLOPT_INFILE*: Le fichier d'entrée de votre transfert.
- *CURLOPT_WRITEHEADER*: Le fichier de destination de l'en-tête de la sortie du transfert.
- *CURLOPT_STDERR*: Le fichier d'erreurs.

curl_exec (PHP 4 >= 4.0.2)

Exécute une session CURL

```
boolean curl_exec  (resource ch)
```

Cette fonction doit être appelée après l'initialisation et le paramètrage d'une session CURL. Son but est simplement d'éxécuter la session *ch*.

curl_close (PHP 4 >= 4.0.2)

Ferme une session CURL

```
void curl_close  (int ch)
```

curl_close() ferme une session CURL et libère toutes les ressources réservées. L'identifiant CURL, *ch*, est aussi effacé.

curl_version (PHP 4 >= 4.0.2)

Retourne la version courante de CURL

```
string curl_version  (void)
```

curl_version() retourne une chaîne avec la version courante de la librairie CURL.

XII. Paiement Cybercash

Ces fonctions ne sont disponibles que si PHP a été compilé avec l'option `--with-cybercash=[DIR]`. Ces fonctions ont été ajoutées dans PHP 4.

cybercash_encr (PHP 4 >= 4.0b4)

```
array cybercash_encr (string wmk, string sk, string inbuff)
```

cybercash_encr() retourne un tableau associatif, contenant les éléments "errcode" et, si "errcode" vaut FALSE, "outbuff" (string), "outLth" (long) et "macbuff" (string).

cybercash_decr (PHP 4 >= 4.0b4)

```
array cybercash_decr (string wmk, string sk, string inbuff)
```

cybercash_decr() retourne un tableau associatif, contenant les éléments "errcode" et, si "errcode" vaut FALSE, "outbuff" (string), "outLth" (long) et "macbuff" (string).

cybercash_base64_encode (PHP 4 >= 4.0b4)

```
string cybercash_base64_encode  (string inbuff)
```

cybercash_base64_decode (PHP 4 >= 4.0b4)

```
string cybercash_base64_decode  (string inbuff)
```

XIII. CyberMUT : Crédit Mutuel

Cette extension vous permet de traiter des transactions de cartes de crédits, avec le système due Crédit Mutuel : CyberMUT (http://www.creditmutuel.fr/centre_commercial/vendez_sur_internet.html).

CynerMUT est un système de paiement français, proposé par le Crédit Mutuel. Si vous n'êtes pas résidants français, ces fonctions ne vous seront pas utiles.

Cette extension n'est disponible que si PHP a été compilé par l'option `--with-cybermut[=DIR]`, où DIR est le dossier qui contient les fichiers `libcm-mac.a` et `cm-mac.h`. Vous aurez besoin du SDK approprié, qui vous est fournit après vous êtes inscrit à CyberMUT (via le web, ou à votre agence la plus proche).

L'utilisation de ces fonctions est presque identique aux fonctions originales, hormis le fait que les fonctions **cybermut_creerformulairecm()** et **cybermut_creerreponsecm()**, qui sont retournées directement par des fonctions PHP, au lieu d'être passées par référence aux fonctions originales.

Ces fonctions ont été ajoutée en 4.0.6.

> **Note :** Ces fonctions ne font que fournis un moyen d'utiliser le SDK CyberMUT. Lisez attentivement le "CyberMUT Developers Guide" pour plus de détails sur les parmètres nécessaires.

cybermut_creerformulairecm (PHP 4 >= 4.0.5)

Génère un formulaire HTML de paiement

```
string cybermut_creerformulairecm (string url_CM, string version, string TPE, string
montant, string ref_commande, string texte_libre, string url_retour, string url_retour_ok,
string url_retour_err, string langue, string code_societe, string texte_bouton)
```

cynermut_creerformulairecm() génère un formulaire HTML, de demande de paiement.

Exemple 1. Première étape du paiment (équivalent à cgi1.c)

```
<?php
// Dossier contenant les clés de chiffrement
putenv("CMKEYDIR=/var/creditmut/cles");
// Numéro de version
$VERSION="1.2";
  $retour =  creditmut_creerformulairecm(
  "https://www.creditmutuel.fr/test/telepaiement/paiement.cgi",
  $VERSION,
  "1234567890",
  "300FRF",
  $REFERENCE,
  $TEXTE_LIBRE,
  $URL_RETOUR,
  $URL_RETOUR_OK,
  $URL_RETOUR_ERR,
  "francais",
  "company",
  "Paiement par carte bancaire");
  echo $retour;
?>
```

Voir aussi **cybermut_testmac()** et **cybermut_creerreponsecm()**.

cybermut_testmac (PHP 4 >= 4.0.5)

Vérifie le message de confirmation

```
bool cybermut_testmac (string code_MAC, string version, string TPE, string cdate, string
montant, string ref_commande, string texte_libre, string code-retour)
```

cybermut_testmac() s'assure qu'il n'y a pas de données parasites dans le message de confirmation recu. Attention aux paramètres `code-retour` and `texte-libre`, qui ne peuvent pas être utilisés directement, car ils contienent des tirets dans leur nom. Vous devez utilisez la syntaxe suivante :

```
<?php
  $code_retour=$HTTP_GET_VARS["code-retour"];
  $texte_libre=$HTTP_GET_VARS["texte-libre"];
?>
```

Exemple 1. Deuxième étape de paiement (équivalent à cgi2.c)

```
<?php
// Assurez vous que l'option Enable Track Vars est active.
// Dossier qui contient les clés de paiement
putenv("CMKEYDIR=/var/creditmut/cles");
// Numéro de version
$VERSION="1.2";
$texte_libre = $HTTP_GET_VARS["texte-libre"];
$code_retour = $HTTP_GET_VARS["code-retour"];
$mac_ok = creditmut_testmac($MAC,$VERSION,$TPE,$date,$montant,$reference,$texte_libre,$code_retour
if ($mac_ok) {
  //
  // Gestion d'un paiement réussi
  //
  //
  $result=creditmut_creerreponsecm("OK");
  } else {
  $result=creditmut_creerreponsecm("Document Falsifié");}
?>
```

Voir aussi **cybermut_creerformulairecm()** et **cybermut_creerreponsecm()**.

cybermut_creerreponsecm (PHP 4 >= 4.0.5)

Génère un accusé de réception de confirmation de paiement

`string` **`cybermut_creerreponsecm`** `(string` *`phrase`*`)`

cybermut_creerreponsecm() retourne une chaîne contenant le message d'accusé de reception.

Le paramètre vaut "OK" si le message de confirmation du paiement a été correctement indentifié par **cybermut_testmac()**. Tout autre chaîne doit être considéré comme une erreur de traitement.

Voir aussi **cybermut_creerformulairecm()** et **cybermut_testmac()**.

XIV. Caractères

> **Avertissement**
>
> Ce module est *EXPERIMENTAL*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

Ces fonctions vérifient si un caractère ou une chaîne de caractères font partie d'une certaine classe de caractères, en fonction de la configuration locale.

Appelée avec un argument de type entier, ces fonctions se comportent exactement comme le équivalent en langage C.

Appelée avec un argument de type chaîne, elles vérifieront chaque caractère de la chaîne, et ne retourneront `TRUE` que si chaque caractère de la chaîne satisfait les critères requis.

Tout autre type d'argument (autre que chaîne ou entier) génère une erreur, et retourne `FALSE` immédiatement.

> **Avertissement**
>
> Ces fonctions ont été ajoutée en PHP 4.0.4, et leur nom peut changer dans un futur proche. Les suggestions actuelles sont : **ctype_issomething()** au lieu de **ctype_somthing()** ou encore d'en faire une partie `ext/standard` et utiliser ainsi leur nom en langage C, même si cela peut conduire à des confusions entre **isset()** et **is_sometype()**.

ctype_alnum (PHP 4 >= 4.0.4)

Vérifie qu'un caractère est alpha-numérique

```
bool ctype_alnum (string c)
```

> **Avertissement**
>
> Ce module est *EXPERIMENTAL*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

Voir aussi **setlocale()**.

ctype_alpha (PHP 4 >= 4.0.4)

Vérifie qu'un caractère est alphabétique

```
bool ctype_alpha (string c)
```

> **Avertissement**
>
> Ce module est *EXPERIMENTAL*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

ctype_cntrl (PHP 4 >= 4.0.4)

Vérifie qu'un caractère est un caractère de contrôle

```
bool ctype_cntrl (string c)
```

> **Avertissement**
>
> Ce module est *EXPERIMENTAL*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

ctype_digit (PHP 4 >= 4.0.4)

Vérifie qu'un caractère est numérique

```
bool ctype_digit (string c)
```

> **Avertissement**
>
> Ce module est *EXPERIMENTAL*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

ctype_lower (PHP 4 >= 4.0.4)

Vérifie qu'un caractère est en minuscule

```
bool ctype_lower (string c)
```

Avertissement

Ce module est *EXPERIMENTAL*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

ctype_graph (PHP 4 >= 4.0.4)

Vérifie qu'un caractère est imprimable (sauf " ", espace)

```
bool ctype_graph (string c)
```

Avertissement

Ce module est *EXPERIMENTAL*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

ctype_print (PHP 4 >= 4.0.4)

Vérifie qu'un caractère est imprimable

```
bool ctype_print (string c)
```

Avertissement

Ce module est *EXPERIMENTAL*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

ctype_punct (PHP 4 >= 4.0.4)

Vérifie qu'un caractère est imprimable, sans être ni un espace, ni un caractère alpha-numérique

```
bool ctype_punct (string c)
```

Avertissement

Ce module est *EXPERIMENTAL*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

ctype_space (PHP 4 >= 4.0.4)

Vérifie qu'un caractère est caractère blanc (espace, tabulation...)

```
bool ctype_space (string c)
```

> **Avertissement**
>
> Ce module est *EXPERIMENTAL*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

ctype_upper (PHP 4 >= 4.0.4)

Vérifie qu'un caractère est en majuscule

```
bool ctype_upper (string c)
```

> **Avertissement**
>
> Ce module est *EXPERIMENTAL*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

ctype_xdigit (PHP 4 >= 4.0.4)

Vérifie qu'un caractère représente un nombre héxadécimal

```
bool ctype_xdigit (string c)
```

> **Avertissement**
>
> Ce module est *EXPERIMENTAL*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

XV. DBA

Ces fonctions sont l'interface avec les bases de type Berkeley.

C'est une couche générale pour plusieurs bases de données sur fichiers. En tant que tel, les fonctionnalités sont limitées à une partie des fonctionnalités des bases de données modernes, comme Sleepycat Software's DB2 (http://www.sleepycat.com/) (A ne pas confondre avec IBM's DB2 software, qui fonctionne avec ODBC).

Le comportement de certaines fonctions dépends de la base de données utilisée. Par exemple **dba_optimize()** et **dba_sync()** n'auront pas le même effet d'une base à l'autre.

Lors de l'utilisation de la fonction **dba_open()** ou de **dba_popen()**, une des librairies suivante doit être fournie comme argument. La liste complète des librairies supportées par votre configuration est disponible avec la fonction **phpinfo()**. (Pour ajouter le support de l'une de ces librairies, ajouter l'option de configuration `--with-XXXX`).

Tableau 1. Liste des librairies DBA

Librairie	Notes
`dbm`	Dbm est la plus ancienne des base de données de type Berkeley. Il vaut mieux l'éviter si possible. Les fonctions de compatibilités codées dans DB2 et gdbm ne sont pas supportées, car elles ne sont compatibles qu'au niveau du code source, et ne peuvent pas gérer le format dbm originel. (`--with-dbm`).
`ndbm`	ndbm est un nouveau type de dbm plus flexible. Il a cependant la majorité des limitations du genre. (`--with-ndbm`).
`gdbm`	gdbm est la base dbm GNU (ftp://ftp.gnu.org/pub/gnu/gdbm/). (`--with-gdbm`).
`db2`	db2 est DB2 de Sleepycat Software (http://www.sleepycat.com/). Elle se décrit comme un "ensemble d'outils qui fournissent une base de données performante, tant pour les applications indépendantes que pour le client/serveur". (`--with-db2`).
`db3`	DB3 est le DB3 de Sleepycat Software (http://www.sleepycat.com/). (`--with-db3`).
`cdb`	cdb est "un package rapide, robuste, léger, pour créer et lire des bases de données constantes". C'est l'auteur de qmail qui l'a écrit, et elle est disponible ici (http://cr.yp.to/cdb.html). Puisque c'est une base constante, elle ne supporte que la lecture. (--with-cdb).

Exemple 1. Exemple DBA

```
<?php
$id = dba_open("/tmp/test.db", "n", "db2");
if(!$id) {
    echo "dba_open a échoué\n";
    exit;
}
dba_replace("key", "Ceci est un exemple!", $id);
if(dba_exists("key", $id)) {
    echo dba_fetch("key", $id);
    dba_delete("key", $id);
}
dba_close($id);
?>
```

DBA gère les données binaires, et n’a pas de limites arbitraires. Elle hérite de toutes les limites de la base sous jacentes.

Toutes les bases de données sur fichiers doivent fournir un moyen de changer le mode d’accès au fichier d’une base, et si possible, de toutes les bases. Le mode d’accès est généralement passé en 4ème argument à **dba_open()** ou **dba_popen()**.

Vous pouvez accéder à toutes les entrées d’une base d’une manière linéaire, avec les fonctions **dba_firstkey()** et **dba_nextkey()**. Vous ne devez pas modifier une base lorsque vous la traversez ainsi.

Exemple 2. Passer en revue une base

```
<?php
// ...ouverture de la base...
$key = dba_firstkey($id);
while($key != FALSE) {
    if(...) {
// conserver la clé pour faire d’autres opérations plus tard
        $handle_later[] = $key;
    }
    $key = dba_nextkey($id);
}
for($i = 0; $i < count($handle_later); $i++)
    dba_delete($handle_later[$i], $id);
?>
```

dba_close (PHP 3>= 3.0.8, PHP 4)

Ferme une base.

```
void dba_close (resource handle)
```

dba_close() ferme le lien établi avec la base et libère toutes les ressources de *handle*.

handle est un identifiant de base, retourné par **dba_open()**.

dba_close() ne retourne aucune valeur.

Voir aussi **dba_open()** et **dba_popen()**.

dba_delete (PHP 3>= 3.0.8, PHP 4)

Efface une entrée.

```
bool dba_delete (string key, resource handle)
```

dba_delete() efface l'entrée spécifiée par la clé *key*, dans la base identifiée par *handle*.

key est la clé de l'entrée à effacer.

handle est un identifiant de lien, retourné par **dba_open()**.

dba_delete() retourne TRUE ou FALSE, si l'entrée est effacée, ou pas effacée, respectivement.

Voir aussi **dba_exists()**, **dba_fetch()**, **dba_insert()** et **dba_replace()**

dba_exists (PHP 3>= 3.0.8, PHP 4)

Vérifie qu'une clé existe.

```
boolean dba_exists (string key, resource handle)
```

dba_exists() vérifie si la clé *key* existe dans la base identifiée par *handle*.

key est la clé qui doit être vérifiée.

handle est un identifiant de base, retourné par **dba_open()**.

dba_exists() retourne TRUE ou FALSE, si la clé est trouvée, ou pas, respectivement.

Voir aussi **dba_fetch()**, **dba_delete()**, **dba_insert()** et **dba_replace()**.

dba_fetch (PHP 3>= 3.0.8, PHP 4)

Lit les données liées à une clé.

```
string dba_fetch (string key, resource handle)
```

dba_fetch() lit les données spécifiée par la clé *key* dans la base identifiée par *handle*.

key est la clé dont on veut lire les données.

handle est un identifiant de base, retourné par **dba_open()**.

dba_fetch() retourne la chaîne associée ou FALSE, si la paire clé/valeur n'a pas été trouvée.

Voir aussi **dba_exists()**, **dba_delete()**, **dba_insert()** et **dba_replace()**.

dba_firstkey (PHP 3>= 3.0.8, PHP 4)

Lit la première clé.

```
string dba_firstkey (resource handle)
```

dba_firstkey() retourne la première clé de la base de données spécifiée par *handle* et y place le pointeur interne de clé. Cela permettra de traverser la base.

handle est un identifiant de base, retourné par **dba_open()**.

dba_firstkey() retourne la clé, ou FALSE, suivant que la première clé existe ou pas.

Voir aussi **dba_nextkey()** et l'exemple 2 de l'introduction DBA.

dba_insert (PHP 3>= 3.0.8, PHP 4)

Insère une entrée.

```
boolean dba_insert (string key, string value, resource handle)
```

dba_insert() insère l'entrée décrite par la clé *key* et la valeur *value* dans la base spécifiée par *handle*. Si une entrée aveec la même clé *key* existe déjà, l'insertion échouera.

key est la clé de la valeur à insérer.

value est la valeur à insérer.

handle est un identifiant de base, retourné par **dba_open()**.

dba_insert() retourne TRUE ou FALSE, suivant que l'insertion a réussi ou échoué.

Voir aussi **dba_exists()**, **dba_delete()**, **dba_fetch()** et **dba_replace()**.

dba_nextkey (PHP 3>= 3.0.8, PHP 4)

Lit la clé suivante.

```
string dba_nextkey (resource handle)
```

dba_nextkey() retourne la clé suivante, dans la base identifiée par *handle* et incrémente le pointeur de clé.

handle est un identifiant de base, retourné par **dba_open()**.

dba_nextkey() retourne la clé, ou FALSE en cas d'échec.

Voir aussi **dba_firstkey()**.

dba_popen (PHP 3>= 3.0.8, PHP 4)

Ouvre une connexion persistante à une base de données.

```
int dba_popen (string path, string mode, string handler [, mixed ...])
```

dba_popen() établit une connexion persistante à la base repérée par *path* avec le mode *mode*, en utilisant l'identifiant *handler*.

path est le chemin sur votre machine.

mode vaut "r" pour lecture seule, "w" pour lecture/écriture, "c" pour lecture/écriture, et création si la base n'existe pas, et "n" pour création, écrasement, et accès en lecture/écriture.

handler est le nom de l'identifiant qui sera utilisé pour accéder à *path*. Il est passé à **dba_popen()**.

dba_popen() retourne un identifiant positif, ou FALSE, suivant que la base a été ouverte, ou que l'accès a échoué.

Voir aussi **dba_open()** et **dba_close()**.

dba_open (PHP 3>= 3.0.8, PHP 4)

Ouvre une base de données.

```
int dba_open (string path, string mode, string handler [, mixed ...])
```

dba_open() établit une connexion à la base repérée par *path* avec le mode *mode* et l'identifiant *handler*.

*path*est le chemin sur votre machine.

mode vaut "r" pour lecture seule, "w" pour lecture/écriture, "c" pour lecture/écriture, et création si la base n'existe pas, et "n" pour création, écrasement, et accès en lecture/écriture.

handler est le nom de l'identifiant qui sera utilisé pour accéder à *path*. Il est passé à **dba_popen()**.

Voir aussi **dba_popen()** et **dba_close()**.

dba_optimize (PHP 3>= 3.0.8, PHP 4)

Optimise une base.

```
boolean dba_optimize (resource handle)
```

dba_optimize() optimise la base de données identifiée par *handle*.

handle est un identifiant de base retourné par **dba_open()**.

dba_optimize() retourne TRUE ou FALSE, suivant que l'optimisation a réussi ou échoué.

Voir aussi **dba_sync()**.

dba_replace (PHP 3>= 3.0.8, PHP 4)

Remplace ou insère une entrée.

```
boolean dba_replace (string key, string value, resource handle)
```

dba_replace() remplaces ou insère une entrée, pour la clé *key* et avec la valeur *value* dans la base identifiée par *handle*.

key est la clé qui va être insérée.

value est la valeur qui va être insérée.

handle est un identifiant de base retourné par **dba_open()**.

dba_replace() retourne TRUE ou FALSE, suivant que l'opération réussit ou échoue.

Voir aussi **dba_exists()**, **dba_delete()**, **dba_fetch()** et **dba_insert()**.

dba_sync (PHP 3>= 3.0.8, PHP 4)

Synchronise une base de données.

```
boolean dba_sync (resource handle)
```

dba_sync() synchronise la base de données spécifiée par *handle*. Si accepté, cela va probablement lancer une opération de réécriture physique du fichier.

handle est un identifiant de base retourné par **dba_open()**.

dba_sync() retourne TRUE ou FALSE, si la synchronisation réussi, ou échoue, respectivement.

Voir aussi **dba_optimize()**.

XVI. Dates et heures

checkdate (PHP 3, PHP 4 >= 4.0b1)

Valide une date/heure.

```
int checkdate (int month, int day, int year)
```

checkdate() retourne TRUE si la date *day*/*month*/*year* est valide, et sinon FALSE. Notez bien que l'ordre des arguments n'est pas l'ordre français. La date est considérée comme valide si :

- L'année est comprise entre entre 1 et 32767 inclus. (pour les versions antérieures à PHP 4.0.3, les années inférieures à 1 étaient aussi valides).
- Le mois est compris entre 1 et 12 inclus
- Le jour est compris dans l'intervalle de dates du mois. Les années bissextiles sont prises en compte.

date (PHP 3, PHP 4 >= 4.0b1)

Formate une date/heure locale

```
string date (string format, int [timestamp])
```

date() retourne une date sous forme d'une chaîne, au format donné par la chaîne format. La date est fournie sous la forme d'un *timestamp*. Par défaut, la date courante est utilisée.

Les caractères suivants sont utilisés pour spécifier le format :

- a - "am" (matin) ou "pm" (après-midi)
- A - "AM" (matin) ou "PM" (après-midi)
- B - Heure Internet Swatch
- d - Jour du mois, sur deux chiffres (éventuellement avec un zéro) : "01" à "31"
- D - Jour de la semaine, en trois lettres (et en anglais) : par exemple "Fri" (pour Vendredi)
- F - Mois, textuel, version longue; en anglais, i.e. "January" (pour Janvier)
- g - Heure, au format 12h, sans les zéros initiaux i.e. "1" à "12"
- G - Heure, au format 24h, sans les zéros initiaux i.e. "0" à "23"
- h - Heure, au format 12h, "01" à "12"
- H - heure, au format 24h, "00" à "23"
- i - Minutes; "00" à "59"
- I (i majuscule) - "1" si l'heure d'été est activée, "0" si l'heure d'hiver .
- j - Jour du mois sans les zéros initiaux: "1" à "31"
- l - ('L' minuscule) - Jour de la semaine, textuel, version longue; en anglais, i.e. "Friday" (pour Vendredi)
- L - Booléen pour savoir si l'année est bissextile ("1") ou pas ("0")
- m - Mois; i.e. "01" à "12"
- M - Mois, en trois lettres (et en anglais) : par exemple "Apr" (pour Avril)
- n - Mois sans les zéros initiaux; i.e. "1" à "12"
- r - Format de date RFC 822; i.e. "Thu, 21 Dec 2000 16:01:07 +0200" (ajouté en PHP 4.0.4)
- s - Secondes; i.e. "00" à "59"
- S - Suffixe ordinal d'un nombre, en anglais, sur deux lettres : i.e. "th", "nd"

- t - Nombre de jours dans le mois donné, i.e. "28" à "31"
- T - Fuseau horaire de la machine ; i.e. "MET"
- U - Secondes depuis une époque
- w - Jour de la semaine, numérique, i.e. "0" (Dimanche) to "6" (Samedi)
- Y - Année, 4 chiffres; i.e. "1999"
- y - Année, 2 chiffres; i.e. "99"
- z - Jour de l'année; i.e. "0" à "365"
- Z - Décalage horaire en secondes (i.e. "-43200" à "43200")

Les caractères non reconnus seront imprimés tels quel. "Z" retournera toujours "0" lorsqu'il est utilisé avec **gmdate()**.

Exemple 1. Exemple avec date()

```
<?php
  print (date("l dS of F Y h:i:s A"));
  print ("July 1, 2000 is on a " . date("l", mktime(0,0,0,7,1,2000)));
?>
```

Il est possible d'utiliser **date()** et **mktime()** ensemble pour générer des dates dans le futur ou dans le passé.

Exemple 2. Exemples avec date() et mktime()

```
<?php
  $tomorrow  = mktime(0,0,0,date("m")  ,date("d") + 1,date("Y")    );
  $lastmonth = mktime(0,0,0,date("m")-1,date("d")    ,date("Y")    );
  $nextyear  = mktime(0,0,0,date("m"),  date("d")    ,date("Y") + 1);
?>
```

Voici maintenant quelques exemples de formatages avec **date()**. Notez que vous devriez échapper tous les autres caractères, car s'ils ont une signification spéciale, ils risquent de produire des effets secondaires indésirables. Notez aussi que les versions futures de PHP peuvent attribuer une signification à des lettres qui sont actuellement inertes. Lorsque vous échappez les caractères, pensez à utiiser des guillemets simples, pour que les séquences \n ne deviennent pas des nouvelles lignes.

Exemple 3. Formatage avec date()

```
<?php
/* Aujourd'hui, le 12 Mars 2001, 10:16:18 pm */
  $aujourdhui = date("F j, Y, g:i a");                 // March 12, 2001, 10:16 pm
  $aujourdhui = date("m.d.y");                         // 03.12.01
  $aujourdhui = date("j, m, Y");                       // 12, 3, 2001
  $aujourdhui = date("Ymd");                           // 20010312
  $aujourdhui = date('h-i-s, j-m-y, it is w Day z ');  // 05-16-17, 12-03-
01, 1631 1618 6 Monpm01
  $aujourdhui = date('\C\'\e\s\t\ \l\e\ jS \j\o\u\r\.');   // C'est le 12th jour.
  $aujourdhui = date("D M j G:i:s T Y");               // Mon Mar 12 15:16:08 MST 2001
  $aujourdhui = date('H:m:s \m \e\s\t\ \l\e\ \m\o\i\s');     // 17:03:18 m est le mois
  $aujourdhui = date("H:i:s");                         // 10:16:18
// notation française
$aujourdhui = date("d/m/y");                          // 12/03/01
$aujourdhui = date("d/m/Y");                          // 12/03/2001
?>
```

Pour formater des dates dans d'autres langues, utilisez les fonctions **setlocale()** et **strftime()**.

Voir aussi **gmdate()** et **mktime()**.

getdate (PHP 3, PHP 4 >= 4.0b1)

Retourne la date/heure

```
array getdate ([int timestamp])
```

getdate() retourne un tableau associatif contenant les informations de date et d'heure du timestamp *timestamp* (lorsqu'il est fourni), avec les champs suivants :

- "seconds" - secondes
- "minutes" - minutes
- "hours" - heures
- "mday" - jour du mois
- "wday" - jour de la semaine, numérique. 0: dimanche jusqu'à 6: samedi
- "mon" - mois, numérique
- "year" - année, numérique
- "yday" - jour de l'année, numérique; i.e. "299"
- "weekday" - jour de la semaine, texte complet (en anglais); i.e. "Friday"
- "month" - mois, texte complet (en anglais); i.e. "January"

Exemple 1. Exemple avec getdate()

```
<?php
  $aujourdhui = getdate();
  $mois = $aujourdhui['month'];
  $mjour = $aujourdhui['mday'];
  $annee = $aujourdhui['year'];
  echo "$mjour/$mois/$annee";
?>
```

gettimeofday (PHP 3>= 3.0.7, PHP 4 >= 4.0b4)

Retourne l'heure actuelle

```
array gettimeofday (void)
```

gettimeofday() est une interface vers gettimeofday(2). Elle retourne un tableau associatif qui contient les informations retournées par le système :

- "sec" - secondes
- "usec" - microsecondes
- "minuteswest" - minutes de décalage par rapport à Greenwich, vers l'Ouest.
- "dsttime" - type de correction dst

gmdate (PHP 3, PHP 4 >= 4.0b1)

Formate une date/heure GMT/CUT.

```
string gmdate (string format [, int timestamp])
```

gmdate() est identique à la fonction **date()**, hormis le fait que le temps retourné est GMT (Greenwich Mean Time) Par exemple, en Finlande (GMT +0200), la première ligne ci-dessous affiche "Jan 01 1998 00:00:00", tandis que la seconde "Dec 31 1997 22:00:00".

Exemple 1. Exemple avec gmdate()

```
<?php
  echo date("M d Y H:i:s", mktime (0,0,0,1,1,1998));
  echo gmdate("M d Y H:i:s", mktime (0,0,0,1,1,1998));
?>
```

Voir aussi **date()**, **mktime()** et **gmmktime()**.

gmmktime (PHP 3, PHP 4 >= 4.0b1)

Retourne le timestamp UNIX d'une date GMT.

```
int gmmktime (int hour, int minute, int second, int month, int day, int year [, int
is_dst])
```

Identique à **mktime()** hormis le fait que les paramètres passés sont GMT.

gmstrftime (PHP 3>= 3.0.12, PHP 4 >= 4.0RC2)

Formate une date/heure GMT/CUT en fonction des paramétrages locaux.

```
string gmstrftime (string format [, int timestamp])
```

gmstrftime() se comporte exactement comme **strftime()** hormis le fait que l'heure utilisée est celle de Greenwich (Greenwich Mean Time (GMT)). Par exemple, dans la zone Eastern Standard Time (est des USA) (GMT -0500), la première ligne de l'exemple ci-dessous affiche "Dec 31 1998 20:00:00", tandis que la seconde affiche "Jan 01 1999 01:00:00".

Exemple 1. Exemple avec gmstrftime()

```
<?php
  setlocale('LC_TIME', 'en_US');
  echo strftime("%b %d %Y %H:%M:%S", mktime (20,0,0,12,31,98))."\n";
  echo gmstrftime("%b %d %Y %H:%M:%S", mktime (20,0,0,12,31,98))."\n";
?>
```

Voir aussi **strftime()**.

localtime (PHP 4 >= 4.0RC2)

Lit l'heure locale

```
array localtime (int [timestamp], boolean [is_associative])
```

localtime() retourne un tableau identique à la structure retournée par la fonction C localtime. Le premier argument *timestamp* est un timestamp UNIX. S'il n'est pas fourni, l'heure courante est utilisée. Le second argument *is_associative*, s'il est mis à 0 ou ignoré, force **localtime()** à retourner un tableau à index numérique. S'il est mis à 1, **localtime()** retourne un tableau associatif, avec tous les éléments de la structure C, accessible avec les clés suivantes :

- "tm_sec" - secondes
- "tm_min" - minutes
- "tm_hour" - heure
- "tm_mday" - jour du mois
- "tm_mon" - mois de l'année
- "tm_year" - Année, incompatible an 2000
- "tm_wday" - Jour de la semaine
- "tm_yday" - Jour de l'année
- "tm_isdst" - Est-ce que l'heure d'hiver a pris effet?

microtime (PHP 3, PHP 4 >= 4.0b1)

Retourne le timestamp UNIX actuel avec microsecondes.

```
string microtime (void)
```

microtime() retourne la chaîne "msec sec" avec sec qui est mesurée en secondes depuis le début de l'époque UNIX, (1er janvier 1970 00:00:00 GMT), et msec qui est le nombre de microsecondes de cette heure. Cette fonction est seulement disponible sur les systèmes d'exploitation qui supportent la fonction système gettimeofday().

Les deux parties de la chaîne sont exprimées en secondes.

Exemple 1. Exemple avec microtime()

```
<?php
  function getmicrotime(){
    list($usec, $sec) = explode(" ",microtime());
    return ((float)$usec + (float)$sec);
  }
  $time_start = getmicrotime();
  for ($i=0; $i < 1000; $i++){
    //ne rien faire, pendant un millier de fois...
  }
  $time_end = getmicrotime();
  $time = $time_end - $time_start;
  echo "Rien fait durant $time secondes";
?>
```

Voir aussi **time()**.

mktime (PHP 3, PHP 4 >= 4.0b1)

Retourne le timestamp UNIX d'une date.

```
int mktime (int hour, int minute, int second, int month, int day, int year [, int is_dst])
```

ATTENTION : l'ordre des arguments est différent de celui de la commande UNIX habituelle mktime(), et fournit des résultats aléatoires si on oublie cet ordre. C'est une erreur très commune que de se tromper de sens.

mktime() retourne un timestamp UNIX correspondant aux arguments fournis. Ce timestamp est un entier long, contenant le nombre de secondes entre le début de l'époque UNIX (1er Janvier 1970) et le temps spécifié.

Les arguments peuvent être omis, de gauche à droite, et tous les arguments manquants sont utilisés avec la valeur courante de l'heure et du jour.

`is_dst` peut être mis à 1 si l'heure d'hiver est appliquée, 0 si elle ne l'est pas, et -1 (par défaut) si on ne sait pas.

> **Note :** `is_dst` a été ajouté à partir de la version 3.0.10.

mktime() est pratique pour faire des calculs de dates et des validations, car elle va automatiquement corriger les valeurs invalides. Par exemple, toutes les lignes suivantes vont retourner la même date : "Jan-01-1998".

Exemple 1. Exemple mktime()

```
<?php
  echo date("M-d-Y", mktime (0,0,0,12,32,1997));
  echo date("M-d-Y", mktime (0,0,0,13,1,1997));
  echo date("M-d-Y", mktime (0,0,0,1,1,1998));
  echo date("M-d-Y", mktime (0,0,0,1,1,98));
?>
```

`year` peut prendre deux ou quatre chiffres, avec les valeurs entre 0-69 qui correspondent à 2000-2069 et 70-99 à 1970-1999 (sur les systèmes où time_t sont sur des entiers 32bit signés, comme cela se fait le plus souvent de nos jours, `year` est valide dans l'intervalle 1902 et 2037.

Le dernier jour d'un mois peut être décrit comme le jour "0" du mois suivant, et non pas le jour -1. Les deux exemples suivants vont donner : "Le dernier jour de Février 2000 est: 29".

Exemple 2. Dernier jour du mois

```
<?php
  $lastday = mktime (0,0,0,3,0,2000);
  echo strftime ("Le dernier jour de Février 2000 est: %d", $lastday);
  $lastday = mktime (0,0,0,4,-31,2000);
  echo strftime ("Le dernier jour de Février 2000 est: %d", $lastday);
?>
```

Voir aussi **date()** et **time()**.

strftime (PHP 3, PHP 4 >= 4.0b1)

Formate une date/heure locale avec les options locales.

```
string strftime (string format, int [timestamp])
```

strftime() retourne la date sous la forme d'une chaîne formatée conformément au format *format*, en utilisant le timestamp *timestamp* donné. Si le *timestamp* est omis, la date actuelle est utilisée. Les mois et jours de la semaine, et toutes les chaînes dépendantes de la langue sont fixées avec la commande **setlocale()**.

Les caractères suivants sont utilisés pour spécifier le format de la date :

- %a - nom abrégé du jour de la semaine (local).
- %A - nom complet du jour de la semaine (local).
- %b - nom abrégé du mois (local).
- %B - nom complet du mois (local).
- %c - représentation préférée pour les dates et heures, en local.
- %C - numéro de siècle (l'année, divisée par 100 et arrondie entre 00 et 99)
- %d - jour du mois en numérique (intervalle 01 à 31)
- %D - identique à %m/%d/%y
- %e - numéro du jour du mois. Les chiffres sont précédés d'un espace (de ' 1' à '31')
- %h - identique à %b
- %H - heure de la journée en numérique, et sur 24-heures (intervalle de 00 à 23)
- %I - heure de la journée en numérique, et sur 12- heures (intervalle 01 à 12)
- %j - jour de l'année, en numérique (intervalle 001 à 366)
- %m - mois en numérique (intervalle 1 à 12)
- %M - minute en numérique
- %n - newline character
- %p - soit 'am' ou 'pm' en fonction de l'heure absolue, ou en fonction des valeurs enregistrées en local.
- %r - l'heure au format a.m. et p.m.
- %R - l'heure au format 24h
- %S - secondes en numérique
- %t - tabulation
- %T - l'heure actuelle (égal à %H:%M:%S)
- %u - le numéro de jour dans la semaine, de 1 à 7. (1 représente Lundi)
- %U - numéro de semaine dans l'année, en considérant le premier dimanche de l'année comme le premier jour de la première semaine.
- %V - le numéro de semaine comme défini dans l'ISO 8601:1988, sous forme décimale, de 01 à 53. La semaine 1 est la première semaine qui a plus de 4 jours dans l'année courante, et dont Lundi est le premier jour.
- %W - numéro de semaine dans l'année, en considérant le premier lundi de l'année comme le premier jour de la première semaine
- %w - jour de la semaine, numérique, avec Dimanche = 0
- %x - format préféré de représentation de la date sans l'heure
- %X - format préféré de représentation de l'heure sans la date
- %y - l'année, numérique, sur deux chiffres (de 00 à 99)
- %Y - l'année, numérique, sur quatre chiffres
- %Z - fuseau horaire, ou nom ou abréviation
- %% - un caractère '%' littéral

> **Note :** Tous les caractères suivants ne sont pas toujours supportés par toutes les librairies C. Dans ce cas, ils ne seront pas supportés par PHP non plus.

Exemple 1. Exemple strftime()

```
<?php
  setlocale ("LC_TIME", "C");
  print(strftime("%A en Finlandais est "));
  setlocale ("LC_TIME", "fi");
  print(strftime("%A, en Français "));
  setlocale ("LC_TIME", "fr");
  print(strftime("%A est en Allemand "));
  setlocale ("LC_TIME", "de");
  print(strftime("%A.\n"));
?>
```

Cet exemple ne fonctionnera que si vous avez les configurations respectives installées sur votre système.

Voir aussi **setlocale()**, **mktime()** et le groupe de spécifications de strftime() (http://www.opengroup.org/onlinepubs/7908799/xsh/strftime.html).

time (PHP 3, PHP 4 >= 4.0b1)

Retourne le timestamp UNIX actuel.

```
int time (void)
```

time() retourne l'heure courante, mesurée en secondes depuis le début de l'époque UNIX, (1er janvier 1970 00:00:00 GMT).

Voir aussi **date()**.

strtotime (PHP 3>= 3.0.12, PHP 4)

Transforme un texte anglais en timestamp

```
int strtotime (string time [, int now])
```

strtotime() essaye de lire une date au format anglais dans la chaîne *time*, et de la transformer en timestamp UNIX.

Exemple 1. Exemple avec strtotime()

```
<?php
// l'exemple n'est pas traduit, car cela ne fonctionne qu'en anglais
  echo strtotime("now") . "\n";
  echo strtotime("10 September 2000") . "\n";
  echo strtotime("+1 day") . "\n";
  echo strtotime("+1 week") . "\n";
  echo strtotime("+1 week 2 days 4 hours 2 seconds")."\n";
?>
```

XVII. dBase

Ces fonctions vous permettront d’accéder aux enregistrements d’une base au format dBase (.dbf).

dBase ne permet pas l’utilisation d’index, de "memo fields", ni le blocage de la base. Deux processus de serveurs web différents modifiant le même fichier dBase risque de rendre votre base de données incohérente.

Les fichiers dBase sont de simples fichiers séquentiels d’enregistrements de longueur fixe. Les enregistrements sont ajoutés à la fin du fichier et les enregistrements supprimés sont conservés jusqu’à l’appel de **dbase_pack()**.

Nous vous recommandons de ne pas utiliser les fichiers dBase comme base de données de production. Choisissez n’importe quel serveur SQL à la place. MySQL et PostgresSQL sont des choix classiques avec PHP. Le support de dBase ne se justifie ici que pour vous permettre d’importer et d’exporter des données depuis et vers votre base des données issues du web, car ce format de fichier est communément accepté par les feuilles et organiseurs Windows. L’import et l’export de données est l’unique chose pour laquelle l’utilisation de dBase est recommandée.

dbase_create (PHP 3, PHP 4 >= 4.0b1)

Crée une base de données dBase.

```
int dbase_create (string filename, array fields)
```

fields est un tableau de tableaux. Chaque tableau décrit le format d'un fichier de la base. Chaque champs est constitué d'un nom, d'un caractère de type de champs, d'une longueur et d'une précision.

Les types de champs disponibles sont :

L

Boolean (booléen). Pas de longueur ou de précision pour ces valeurs.

M

Memo. (Note importante : les Memos ne sont pas supportés par PHP.) Elles n'ont pas de longueur ou de précision.

D

Date (enregistrée au format 'YYYYMMDD'). Elles n'ont pas de longueur ou de précision.

N

Number (nombre). Possède une longueur et un précision (le nombre de chiffres après la virgule).

C

String (chaîne).

Si la base de données a été créée, un identifiant de base dbase_identifier est retourné, sinon, FALSE est retourné.

Exemple 1. Création d'une base dBase

```
<?php
// "database" name
  $dbname = "/tmp/test.dbf";
// database "definition"
  $def =
    array(
        array("date",     "D"),
        array("name",     "C",  50),
        array("age",      "N",   3, 0),
        array("email",    "C", 128),
        array("ismember", "L")
    );
// création
  if (!dbase_create($dbname, $def))
    print "<strong>Error!</strong>";
?>
```

dbase_open (PHP 3, PHP 4 >= 4.0b1)

Ouverture d'une base dBase.

```
int dbase_open (string filename, int flags)
```

flags est un flag, comme pour la fonction open(). (Typiquement; 0 signifie lecture seule, 1 signifie écriture seule, et 2 écriture/lecture).

dbase_open() retourne un identifiant de base de données, ou FALSE si la base n'a pas pu être sélectionnée.

dbase_close (PHP 3, PHP 4 >= 4.0b1)

Ferme une base dBase.

```
boolean dbase_close (resource dbase_identifier)
```

dbase_close() ferme la base associée à *dbase_identifier*.

dbase_pack (PHP 3, PHP 4 >= 4.0b1)

Compacte une base dBase.

```
boolean dbase_pack (resource dbase_identifier)
```

dbase_pack() compacte la base de données *dbase_identifier* (effacement définitif de tous les enregistrements marqués pour l'effacement, avec la fonction **dbase_delete_record()**).

dbase_add_record (PHP 3, PHP 4 >= 4.0b1)

Ajoute un enregistrement dans une base dBase.

```
boolean dbase_add_record (resource dbase_identifier, array record)
```

dbase_add_record() ajoute les données de *record* dans la base spécifiée par *dbase_identifier*. Si le nombre de colonnes fourni n'est pas celui du nombre de champs dans la base, l'opération échouera, et FALSE sera retourné.

dbase_replace_record (PHP 3>= 3.0.11, PHP 4 >= 4.0b1)

Remplace un enregistrement dans une base dBase.

```
boolean dbase_replace_record (resource dbase_identifier, array record, int
dbase_record_number)
```

dbase_replace_record() remplace les données associées à l'enregistrement *record_number* par les données enregistrées dans *record*. Si le nombre de colonnes fourni n'est pas celui du nombre de champs dans la base, l'opération échouera, et FALSE sera retourné.

dbase_record_number est un entier qui peut aller de 1 jusqu'au nombre maximal d'enregistrement de la base (retourné par **dbase_numrecords()**).

dbase_delete_record (PHP 3, PHP 4 >= 4.0b1)

Efface un enregistrement dans une base dBase.

```
boolean dbase_delete_record (resource dbase_identifier, int record)
```

dbase_delete_record() marque l'enregistrement *record* pour l'effacement, dans la base *dbase_identifier*. Pour effacer réellement l'enregistrement, il faut utiliser aussi **dbase_pack()**.

dbase_get_record (PHP 3, PHP 4 >= 4.0b1)

Lit un enregistrement dans une base dBase.

```
array dbase_get_record (resource dbase_identifier, int record)
```

dbase_get_record() retourne les données de l'enregistrement *record* dans un tableau. Le tableau est indexé à partir de 0, et inclus un membre nommé 'deleted' (effacé), qui sera mis à 1 si l'enregistrement a été marqué pour l'effacement (voir **dbase_delete_record()**).

Chaque champs est converti au format approprié PHP. (Les dates sont laissées au format chaîne).

dbase_get_record_with_names (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Lit un enregistrement dans une base, sous la forme d'un tableau associatif.

```
array dbase_get_record_with_names  (resource dbase_identifier, int record)
```

dbase_identifier retourne les données de l'enregistrement *record* dans un tableau associatif. Le tableau inclus un membre nommé 'deleted' (effacé), qui sera mis à 1 si l'enregistrement a été marqué pour l'effacement (voir **dbase_delete_record()**).

Chaque champs est converti au format approprié PHP. (Les dates sont laissées au format chaîne).

dbase_numfields (PHP 3, PHP 4 >= 4.0b1)

Compte le nombre de champs d'une base dBase.

```
int dbase_numfields (resource dbase_identifier)
```

dbase_numfields() retourne le nombre de champs (colonnes) de la base de données *dbase_identifier*. Les numéros de champs sont numérotés de 0 à dbase_numfields($db)-1, tandis que les numéros d'enregistrements sont numérotés de 1 à dbase_numrecords($db).

Exemple 1. Utiliser dbase_numfields()

```
<?php
  $rec = dbase_get_record($db, $recno);
  $nf  = dbase_numfields($db);
  for ($i=0; $i < $nf; $i++) {
    print $rec[$i]."<br>\n";
  }
?>
```

dbase_numrecords (PHP 3, PHP 4 >= 4.0b1)

Compter le nombre d’enregistrements dans une base dBase.

```
int dbase_numrecords (resource dbase_identifier)
```

dbase_numrecords() retourne le nombre d’enregistrements (lignes) dans la base *`dbase_identifier`*. Les numéros de champs sont numérotés de 0 à `dbase_numfields($db)-1`, tandis que les numéros d’enregistrements sont numérotés de 1 à `dbase_numrecords($db)`.

XVIII. DBM

Ces fonctions vous permettent d'écrire des lignes dans une base de données de type dbm. Ce type de base (supporté par Berkeley db, gdbm, et quelques librairies systèmes, ou certaines librairies du système d'exploitation) enregistre les paires clés/valeurs, (contrairement aux enregistrements par ligne, utilisé par les autres bases de données relationnelles).

Exemple 1. Présentation de dbm

```
<?php
  $dbm = dbmopen("dernier", "w");
  if (dbmexists($dbm, $userid)) {
    $last_seen = dbmfetch($dbm, $userid);
  } else {
    dbminsert($dbm, $userid, time());
  }
  faire_quelquechose();
  dbmreplace($dbm, $userid, time());
  dbmclose($dbm);
?>
```

dbmopen (PHP 3, PHP 4 >= 4.0b1)

Ouvre une base de données dbm

```
resource dbmopen (string filename, string flags)
```

Le premier argument est le chemin absolu jusqu'au fichier dbm à ouvrir. Le deuxième argument est le mode d'ouverture du fichier, qui peut prendre les valeurs suivantes : "r", "n", "c" ou "w" qui représentent respectivement lecture seule, nouveau (ce qui implique lecture/écriture, et qui, probablement, va écraser une base existante), création(ce qui implique lecture/écriture, et qui, probablement, va écraser une base existante), et lecture/écriture.

dbmopen() retourne un identifiant, qui sera passé à toutes les autres fonctions dbm, en cas de succès, ou `FALSE` en cas d'échec.

Si ndbm est utilisé, ndbm va créer les fichiers `filename.dir` et `filename.pag`. gdbm n'utilise qu'un fichier, tout comme les librairies internes, et Berkeley db crée le fichier `filename.db`. Notez que PHP dispose de son propre système de verrouillage des fichiers, qui s'additionne à celui éventuellement utilisé par les librairies. PHP n'efface jamais les fichiers `.lck` qu'il crée. Il les utilise comme inode fixe, sur lequel faire le verrouillage. Pour plus d'informations sur les fichiers dbm, reportez-vous à vos pages de manuel Unix (man) , ou bien chargez gdbm : `ftp://prep.ai.mit.edu/pub/gnu`.

dbmclose (PHP 3, PHP 4 >= 4.0b1)

Ferme une base de données dbm.

```
boolean dbmclose (resource dbm_identifier)
```

dbmclose() déverrouille et ferme la base de données *`dbm_identifier`*.

dbmexists (PHP 3, PHP 4 >= 4.0b1)

Indique si une valeur existe.

```
boolean dbmexists (resource dbm_identifier, string key)
```

dbmexists() retourne `TRUE` s'il y a une valeur associée à la clé *`key`*.

dbmfetch (PHP 3, PHP 4 >= 4.0b1)

Lit une valeur.

```
string dbmfetch (resource dbm_identifier, string key)
```

dbmfetch() retourne la valeur associée à la clé *`key`*.

dbminsert (PHP 3, PHP 4 >= 4.0b1)

Insère une valeur.

```
int dbminsert (resource dbm_identifier, string key, string value)
```

dbminsert() ajoute la valeur *`value`* dans la base de données, avec la clé *`key`*.

dbminsert() retourne -1 si la base a été ouverte en mode lecture seule, 0 si l'insertion a été réussie, et 1 si la clé key existe déjà. (Pour remplacer la valeur, utilisez **dbmreplace()**.)

dbmreplace (PHP 3, PHP 4 >= 4.0b1)

Remplace une valeur.

```
boolean dbmreplace (resource dbm_identifier, string key, string value)
```

dbmreplace() remplace la valeur courante par la valeur *value* pour la clé *key*, dans une base *dbm_identifier*.

dbmreplace() crée la clé, si elle n'existe pas dans la base.

dbmdelete (PHP 3, PHP 4 >= 4.0b1)

Efface une valeur.

```
boolean dbmdelete (resource dbm_identifier, string key)
```

dbmdelete() efface la valeur de la clé *key*, dans la base *dbm_identifier*.

dbmdelete() retourne FALSE si la clé n'existe pas dans cette base.

dbmfirstkey (PHP 3, PHP 4 >= 4.0b1)

Lit la première clé.

```
string dbmfirstkey (resource dbm_identifier)
```

dbmfirstkey() retourne la première clé de la base de données. Notez bien que les clés ne sont pas dans un ordre défini, étant donné que la table est construite comme un tableau associatif.

dbmnextkey (PHP 3, PHP 4 >= 4.0b1)

Lit la clé suivante.

```
string dbmnextkey (resource dbm_identifier, string key)
```

dbmnextkey() retourne la clé après la clé *key*. En appelant **dbmfirstkey()**, puis successivement **dbmnextkey()**, il est possible de passer en revue toute les paires clé/valeur de la base de données dbm. Par exemple :

Exemple 1. Passer en revue une base de données.

```
<?php
  $cle = dbmfirstkey($dbm_id);
  while ($cle){
    echo "$cle = " . dbmfetch($dbm_id, $cle) . "\n";
    $cle = dbmnextkey($dbm_id, $cle);
  }
?>
```

dblist (PHP 3, PHP 4 >= 4.0b1)

Décrit la librairie dbm utilisée.

```
string dblist (void)
```

XIX. dbx

Avertissement

Ce module est *EXPERIMENTAL*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

Le module dbx est un module d'abstraction de base de données (db pour database (base de données) et 'X' pour toutes les bases supportées). Les fonctions dbx vous permettent d'accéder à toutes les bases supportées, avec la même convention. Pour cela il vous faut avoir ces fonctions compilées avec PHP (option de configuration `--enable-dbx` et toutes les bases que vous souhaitez utiliser. Par exemple, si vous voulez accéder à MySQL depuis dbx, vous devez aussi configurer PHP avec `--with-mysql`. Les fonctions dbx n'interface pas directement les bases de données, mais utilise les modules de support de ces bases. Pour pouvoir utiliser une base avec le module dbx, il vous faut l'avoir configuré avec PHP, ou bien la charger dynamiquement. Actuellement les bases MySQL, PostgreSQL et ODBC sont supportées, mais d'autres suivront bientôt (j'espère).

La documentation complémentaire (ajout de nouvelles bases à dbx) est accessible à http://www.guidance.nl/php/dbx/doc/.

dbx_close (PHP 4 >= 4.0.6)

Ferme une connexion à une base

```
boolean dbx_close (resource link_identifier)
```

> **Avertissement**
>
> Ce module est *EXPERIMENTAL*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbx_close() retourne TRUE en cas de succès, et FALSE en cas d'erreur.

Exemple 1. Exemple avec dbx_close()

```
<?php
$link = dbx_connect("mysql", "localhost", "base", "utilisateur", "mot de passe")
    or die ("Impossible de se connecter");
print("Connexion réussie");
dbx_close($link);
?>
```

Note : Reportez vous aussi à documentation de la base de données que vous utilisez.

Voir aussi **dbx_connect()**.

dbx_connect (PHP 4 >= 4.0.6)

Ouvre une connexion à une base de données

```
resource dbx_connect (string module, string host, string database, string username, string
password [, int persistent])
```

> **Avertissement**
>
> Ce module est *EXPERIMENTAL*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbx_connect() retourne une resource dbx_link_object en cas de succès, FALSE sinon. Si la connexion a put être établie, mais que la base de données n'a pas pu être selectionnée, la fonction retournera quand même une ressource. Le paramètre *persistent* peut prendre la valeur DBX_PERSISTENT, pour créer une connexion persitante.

Les valeurs possibles de *module* sont les suivantes (n'oubliez pas que cela fonctionnera que si le module associé est chargé):

- module 1: "mysql"
- module 2: "odbc"
- module 3: "pgsql"

Le support de pgsql est au stade expérimental, et vous devez compiler vous-même le module pgsql après avoir modifié un des fichiers sources. Sinon, vous aurez une alerte affichée à chaque requête.

La ressource dbx_link_object a trois membres : ’handle’, ’module’ et ’database’. Le membre ’database’ contient le nom de la base de données actuellement selectionnée. Le membre ’module’ est à usage interne à dbx, et contient le numéro de module sus-cité. Le membre ’handle’ est une resource valide de connexion à la base de données, et peut être utilisé en tant que tel dans les autres fonctions spécifiques à cette base de données.

```
<?php
$link = dbx_connect("mysql", "localhost", "base de données", "utilisa-
teur", "mot de passe");
mysql_close($link->handle);
// dbx_close($link) est beaucoup plus adapté ici
?>
```

Les paramètres *`host`*, *`database`*, *`username`* et *`password`* sont attendus, mais ne sont pas toujours utiles, suivant la fonction de connexion de la base de données utilisée.

Exemple 1. Exemple avec dbx_connect()

```
<?php
$link = dbx_connect("odbc", "", "base de données", "utilisa-
teur", "mot de passe", DBX_PERSISTENT)
    or die ("Impossible de se connecter");
print ("Connexion réussie");
dbx_close($link);
?>
```

Note : Reportez vous aussi à documentation de la base de données que vous utilisez.

Voir aussi **dbx_close()**.

dbx_error (PHP 4 >= 4.0.6)

Rapporte le message d’erreur du dernier appel de fonction

```
string dbx_error (resource link_identifier)
```

Avertissement

Ce module est *EXPERIMENTAL*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbx_error() retourne une chaîne contenant le message d’erreur du module sélectionné. S’il y a des connexions multiples sur le même module, seule la dernière erreur est fournie. S’il a des des connexions sur différents modules, la dernière erreur du module est retourné (le module est alors représenté par *`link_identifier`*). Notez que le module ODBC ne supporte pas encore cette fonction.

Exemple 1. Exemple avec dbx_error()

```
<?php
$link = dbx_connect("mysql", "localhost", "base de données", "utilisateur", "mot de passe")
    or die ("Impossible de se connecter");
$result = dbx_query($link, "select id from nonexistingtbl");
if ($result==0) {
    echo dbx_error($link);
```

```
}
dbx_close($link);
?>
```

Note : Reportez vous aussi à documentation de la base de données que vous utilisez.

dbx_query (PHP 4 >= 4.0.6)

Envoie une requête et lit tous les résultats

```
resource dbx_query (resource link_identifier, string sql_statement [, long flags])
```

Avertissement

Ce module est *EXPERIMENTAL*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbx_query() retourne une resource dbx_result_object ou 1 en cas de succès (un objet de résultat ne sera retourné que pour les requêtes SQL qui retournent un résultat), ou 0 en cas d'erreur. Le paramètre `flags` sert à contrôler la quantité d'informations retournée. Il peut être n'importe quelle combinaisons par OR des constantes : DBX_RESULT_INFO, DBX_RESULT_INDEX, DBX_RESULT_ASSOC. DBX_RESULT_INFO fournit des informations sur les colonnes, comme les noms des champs et leur type. DBX_RESULT_INDEX retourne le résultat dans un tableau indexé (par exemple, data[2][3], où 2 est le numéro de ligne et 3 est le numéro de colonne), dont la première colonne est indexée à 0. DBX_RESULT_ASSOC associe les noms de colonnes avec leurs indices. Notez que DBX_RESULT_INDEX est toujours retourné, indépendamment de la valeur de `flags`. Si DBX_RESULT_ASSOC est spécifié, DBX_RESULT_INFO est aussi retourné, même s'il n'a pas été spécifié. Ce qui signifie que les seules combinaisons envisageables sont DBX_RESULT_INDEX, DBX_RESULT_INDEX | DBX_RESULT_INFO et DBX_RESULT_INDEX | DBX_RESULT_INFO | DBX_RESULT_ASSOC. La dernière combinaison est la valeur par défaut de `flags`. Les résultats associés sont en fait des références, ce qui fait que modifier `data[0][0]`, revient à modifier `data[0]['fieldnameforfirstcolumn']`.

Un objet dbx_result_object a 5 membres (éventuellement 4, suivants les valeurs de `flags`) : 'handle', 'cols', 'rows', 'info' (optionnel) et 'data'. Handle est un identifiant de résultat, qui peut être utilisé avec les fonctions spécifiques à son module. Par exemple :

```
<?php
$result = dbx_query($link, "SELECT id FROM tbl");
mysql_field_len($result->handle, 0);
?>
```

Les membres cols et rows contiennent les numéros de colonne et de champs.

```
<?php
$result = dbx_query($link, "SELECT id FROM tbl");
echo "Taille du résultat: " . $result->rows . " x " . $result->cols . "<br>\n";
?>
```

Le membre info n'est retourné que si DBX_RESULT_INFO et/ou DBX_RESULT_ASSOC sont spécifié dans le paramètre `flags`. C'est un deuxième tableau, qui possède deux lignes ("name" and "type"), pour connaître les informations sur les colonnes.

```
<?php
$result = dbx_query($link, "SELECT id FROM tbl");
echo "Nom de la colonne : " . $result->info["name"][0] . "<br>\n";
echo "Type de la colonne: " . $result->info["type"][0] . "<br>\n";
?>
```

Le membre data contient les données effectivement lues, éventuellement associées à des noms de colonnes. Si DBX_RESULT_ASSOC est utilisé, il est possible d'utiliser `$result->data[2]["fieldname"]`.

Exemple 1. Exemple avec dbx_query()

```
<?php
$link = dbx_connect("odbc", "", "base de données", "utilisateur", "mot de passe")
    or die ("Impossible de se connecter");
$result = dbx_query($link, "SELECT id, parentid, description FROM tbl");
if ($result==0) echo "La requête a échoué\n<br>";
elseif ($result==1) {
    echo "La requête a réussie\n<br>";
} else {
    $rows=$result->rows;
    $cols=$result->cols;
    echo "<p>table dimension: {$result->rows} x {$result->cols}<br><table border=1>\n";
    echo "<tr>";
    for ($col=0; $col<$cols; ++$col) {
        echo "<td>-{$result->info["name"][$col]}-<br>-{$result->info["type"][$col]}-</td>";
    }
    echo "</tr>\n";
    for ($row=0; $row<$rows; ++$row){
        echo "<tr>";
        for ($col=0; $col<$cols; ++$col) {
            echo "<td>-{$result->data[$row][$col]}-</td>";
        }
        echo "</tr>\n";
    }
    echo "</table><p>\n";
    echo "table dimension: {$result-
>rows} x id, parentid, description<br><table border=1>\n";
    for ($row=0; $row<$rows; ++$row) {
        echo "<tr>";
        echo "<td>-{$result->data[$row]["id"]}-</td>";
        echo "<td>-{$result->data[$row]["parentid"]}-</td>";
        echo "<td>-{$result->data[$row]["description"]}-</td>";
        echo "</tr>\n";
    }
    echo "</table><p>\n";
}
dbx_close($link);
?>
```

Note : Reportez vous aussi à documentation de la base de données que vous utilisez.

Voir aussi **dbx_connect()**.

dbx_sort (PHP 4 >= 4.0.6)

Tri un résultat avec une fonction utilisateur

```
boolean dbx_sort (dbx_result_object result, string user_compare_function)
```

> **Avertissement**
>
> Ce module est *EXPERIMENTAL*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbx_sort() retourne TRUE en cas de succès, et FALSE sinon.

Exemple 1. Exemple avec dbx_sort()

```
<?php
function user_re_order ($a, $b) {
    $rv = dbx_cmp_asc($a, $b, "parentid");
    if (!$rv) $rv = dbx_cmp_asc($a, $b, "id");
    return $rv;
}

$link = dbx_connect("odbc", "", "base de données", "utilisateur", "mot de passe")
    or die ("Impossible de se connecter");
$result = dbx_query($link, "SELECT id, parentid, description FROM tbl ORDER BY id");
echo "Les données sont maintenant triées par id<br>";
dbx_query($result, "user_re_order");
echo "Les données sont maintenant triées par parentid, puis par id<br>";
dbx_close($link);
?>
```

Voir aussi **dbx_cmp_asc()** et **dbx_cmp_desc()**.

dbx_cmp_asc (PHP 4 4.0.6 only)

Compare deux lignes pour tri croissant

```
int dbx_cmp_asc (array row_a, array row_b, string columnname_or_index)
```

> **Avertissement**
>
> Ce module est *EXPERIMENTAL*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbx_cmp_asc() retourne 0 si row_a[$columnname_or_index] est égal à row_b[$columnname_or_index], 1 si elle est plus grande et -1 si elle est plus petite.

Exemple 1. Exemple avec dbx_cmp_asc()

```
<?php
function user_re_order($a, $b) {
    $rv = dbx_cmp_asc($a, $b, "parentid");
    if (!$rv) {
        $rv = dbx_cmp_asc($a, $b, "id");
        return $rv;
    }
}
```

```
$link = dbx_connect("odbc", "", "base de données", "utilisateur", "mot de passe")
    or die ("Impossible de se connecter");
$result = dbx_query($link, "SELECT id, parentid, description FROM tbl ORDER BY id");
echo "Les données sont maintenant triées par id<br>";
dbx_query($result, "user_re_order");
echo "Les données sont maintenant triées par parentid, puis par id<br>";
dbx_close($link);
?>
```

Voir aussi **dbx_sort()** et **dbx_cmp_desc()**.

dbx_cmp_desc (PHP 4 4.0.6 only)

Compare deux lignes pour tri décroissant

```
int dbx_cmp_desc (array row_a, array row_b, string columnname_or_index)
```

dbx_cmp_desc() retourne 0 si row_a[$columnname_or_index] est égal à row_b[$columnname_or_index], 1 si elle est plus grande et -1 si elle est plus petite.

Exemple 1. Exemple avec dbx_cmp_desc()

```
<?php
function user_re_order ($a, $b) {
    $rv = dbx_cmp_asc($a, $b, "parentid");
    if (!$rv) {
        $rv = dbx_cmp_asc($a, $b, "id");
        return $rv;
    }
}

$link = dbx_connect("odbc", "", "base de données", "utilisateur", "mot de passe")
    or die ("Impossible de se connecter");
$result = dbx_query($link, "SELECT id, parentid, description FROM tbl ORDER BY id");
echo "Les données sont maintenant triées par id<br>";
dbx_query($result, "user_re_order");
echo "Les données sont maintenant triées par parentid, puis par id<br>";
dbx_close($link);
?>
```

Voir aussi **dbx_sort()** et **dbx_cmp_asc()**.

XX. DB++ functions

Avertissement

Ce module est *EXPERIMENTAL*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

Experimental support for db++ database

This paper describes the the db++ extension which enables PHP to access db++ relation files through all of the search and update methods available in the client/server "C" library interface of db++ and to read and process the output of a db++ query.

Requirenments

??? Download where

Installation

Creation and installation of this extension requires the db++ client libraries and header files to be installed on your system as described above. You have to run **configure** with option `--with-dbplus` to build this extension.

configure looks for the client libraries and header files under the default path`/usr/dbplus/`. If you have installed db++ in a different place you have add the installation path to the **configure** option like this:
`--with-dbplus=/your/installation/path`.

dbplus_add (unknown)

Add a tuple to a relation

```
int dbplus_add (int relation, array tuple)
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbplus_aql (unknown)

Perform AQL query

```
int dbplus_aql (string query [, string server [, string dbpath]])
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbplus_chdir (unknown)

Get/Set database virtual current directory

```
string dbplus_chdir ([string newdir])
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbplus_close (unknown)

Close a relation

```
int dbplus_close (int relation)
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbplus_curr (unknown)

Get current tuple from relation

```
int dbplus_curr (int relation, array tuple)
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbplus_errcode (unknown)

Get error string for given errorcode or last error

```
string dbplus_errcode (int err)
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbplus_first (unknown)

Get first tuple from relation

```
int dbplus_first (int relation, array tuple)
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbplus_flush (unknown)

???

```
int dbplus_flush (int relation)
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbplus_freealllocks (unknown)

Free all locks held by this client

```
int dbplus_freealllocks ()
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbplus_freerlocks (unknown)

Free all locks on given relation

```
int dbplus_freerlocks (int relation)
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbplus_info (unknown)

???

```
int dbplus_info (int relation, string key, array )
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbplus_last (unknown)

Get last tuple from relation

```
int dbplus_last (int relation, array tuple)
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbplus_lockrel (unknown)

Request read-lock on relation

```
int dbplus_lockrel (int relation)
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbplus_next (unknown)

Get next tuple from relation

```
int dbplus_next (int relation, array )
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbplus_open (unknown)

Open relation file

```
int dbplus_open (string name)
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

The relation file *name* will be opened. *name* can be either a file name or a relative or absolute path name. This will be mapped in any case to an absolute relation file path on a specific host machine and server.

On success a relation file handle (cursor) is returned which must be used in any subsequent commanads referencing the relation.

dbplus_prev (unknown)

Get previous tuple from relation

```
int dbplus_prev (int relation, array tuple)
```

Avertissement

Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbplus_restorepos (unknown)

???

```
int dbplus_restorepos (int relation, array tuple)
```

Avertissement

Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbplus_ropen (unknown)

Open relation file ... ???

```
int dbplus_ropen (string name)
```

Avertissement

Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbplus_runlink (unknown)

Remove relation from filesystem

```
int dbplus_runlink (int relation)
```

Avertissement

Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbplus_rzap (unknown)

Remove all tuples from relation

```
int dbplus_rzap (int relation, int truncate)
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbplus_savepos (unknown)

???

```
int dbplus_savepos (int relation)
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbplus_setindex (unknown)

???

```
int dbplus_setindex (int relation, string idx_name)
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbplus_setindexbynumber (unknown)

???

```
int dbplus_setindexbynumber (int relation, int idx_number)
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbplus_sql (unknown)

Perform SQL query

```
int dbplus_sql (string query, string server, string dbpath)
```

Avertissement

Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbplus_tremove (unknown)

Remove tuple and return new current tuple

```
int dbplus_tremove (int relation, array old [, array current])
```

Avertissement

Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbplus_undo (unknown)

???

```
int dbplus_undo (int relation)
```

Avertissement

Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbplus_undoprepare (unknown)

???

```
int dbplus_undoprepare (int relation)
```

Avertissement

Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbplus_unlockrel (unknown)

Give up read-lock on relation

```
int dbplus_unlockrel (int relation)
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbplus_unselect (unknown)

???

```
int dbplus_unselect (int relation)
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbplus_update (unknown)

Update specified tuple in relation

```
int dbplus_update (int relation, array old, array new)
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbplus_xlockrel (unknown)

Request exclusive write lock on relation

```
int dbplus_xlockrel (int relation)
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbplus_xunlockrel (unknown)

Free exclusive write lock on relation

```
int dbplus_xunlockrel (int relation)
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbplus_change (unknown)

```
int dbplus_change (int handle, string domain)
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

Not implemented yet.

dbplus_find (unknown)

```
int dbplus_find (int handle, string constr, string tname)
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

Not implemented yet.

dbplus_freelock (unknown)

```
int dbplus_freelock (int handle, string tname)
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

Not implemented yet.

dbplus_getlock (unknown)

```
int dbplus_getlock (int handle, string tname)
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

Not implemented yet.

dbplus_getunique (unknown)

```
int dbplus_getunique (int handle, string varname, int flush)
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

Not implemented yet.

dbplus_rchperm (unknown)

```
int dbplus_rchperm (int handle, int mask, string user, string group)
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

Not implemented yet.

dbplus_rcreate (unknown)

```
int dbplus_rcreate (string name, string domlist, int flag)
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

Not implemented yet.

dbplus_rcrtexact (unknown)

```
int dbplus_rcrtexact (string name, int handle, int flag)
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

Not implemented yet.

dbplus_rcrtlike (unknown)

```
int dbplus_rcrtlike (string name, int handle, int flag)
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

Not implemented yet.

dbplus_resolve (unknown)

```
int dbplus_resolve (string name, string vname)
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

Not implemented yet.

dbplus_rkeys (unknown)

```
int dbplus_rkeys (int handle, string domlist)
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

Not implemented yet.

dbplus_rquery (unknown)

```
int dbplus_rquery ()
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

dbplus_rrename (unknown)

```
int dbplus_rrename (int handle, string name, int flag)
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

Not implemented yet.

dbplus_rsecindex (unknown)

```
int dbplus_rsecindex (int handle, string domlist, int compact)
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

Not implemented yet.

dbplus_tcl (unknown)

```
int dbplus_tcl (int sid, string script)
```

> **Avertissement**
>
> Cette fonction est *EXPERIMENTALE*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

Not implemented yet.

XXI. Accès aux dossiers

chroot (PHP 4 >= 4.0.5)

Change la racine

```
int chroot (string directory)
```

chroot() change la racine du script en cours, et la remplace par *directory*. **chroot()** retourne FALSE s'il n'a pas pu modifier la racine, et TRUE sinon.

> **Note :** Il n'est pas conseillé d'utiliser cette fonction sur un site web, car il n'est pas possible de restaurer la racine à sa valeur initiale à la fin de la requête. Cette fonction n'est viable que lorsque PHP est utilisé en CGI.

chdir (PHP 3, PHP 4 >= 4.0b1)

Change de dossier

```
int chdir (string directory)
```

chdir() change le dossier courant de PHP en *directory*. **chdir()** retourne FALSE si l'opération échoue, et TRUE sinon.

dir (PHP 3, PHP 4 >= 4.0b1)

Classe dossier

```
new dir (string directory)
```

Un mécanisme pseudo-objet permet la lecture d'un dossier. L'argument *directory* doit être ouvert. Deux propriétés sont disponibles une fois le dossier ouvert : le pointeur peut être utilisé avec d'autres fonctions telles que **readdir()**, **rewinddir()** et **closedir()**. Le chemin du dossier est le chemin fourni lors de la construction de l'objet. Trois méthodes permettent de lire, remettre à zéro et fermer le dossier.

Exemple 1. Exemple avec dir()

```
<?php
$d = dir("/etc");
echo "Pointeur: ".$d->handle."<br>\n";
echo "Chemin: ".$d->path."<br>\n";
while($entry=$d->read()) {
    echo $entry."<br>\n";
}
$d->close();
?>
```

closedir (PHP 3, PHP 4 >= 4.0b1)

Ferme le pointeur sur le dossier.

```
void closedir (resource dir_handle)
```

closedir() ferme le pointeur de dossier *dir_handle*. Le dossier devait avoir été ouvert avec **opendir()**.

getcwd (PHP 4 >= 4.0b4)

Retourne le dossier de travail

```
string getcwd (void)
```

getcwd() retourne le nom du dossier courant.

opendir (PHP 3, PHP 4 >= 4.0b1)

Ouvre un dossier, et récupère un pointeur dessus.

```
int opendir (string path)
```

opendir() retourne un pointeur sur un dossier pour être utilisé avec les fonctions **closedir()**, **readdir()** et **rewinddir()**.

Si le paramètre *path* n'est pas un dossier valide, ou si le dossier ne peut être accédé pour des raisons de permissions ou des erreurs liées au système de fichiers, **opendir()** retourne FALSE et génère une erreur PHP. Vous pouvez supprimer cette erreur en ajoutant @ avant le nom de la fonction.

Exemple 1. Exemple opendir()

```
<?php
if ($dir = @opendir("/tmp")) {
  while($file = readdir($dir)) {
    echo "$file\n";
  }
  closedir($dir);
}
?>
```

readdir (PHP 3, PHP 4 >= 4.0b1)

Lit une entrée du dossier.

```
string readdir (resource dir_handle)
```

readdir() retourne le nom du fichier suivant dans le dossier identifié par *dir_handle*. Les noms sont retournés dans n'importe quel ordre.

Exemple 1. Liste tous les fichiers du dossier courant

```
<?php
$handle=opendir('.');
echo "Pointeur de dossier: $handle\n";
echo "Fichiers:\n";
while ($file = readdir($handle)) {
    echo "$file\n";
}
closedir($handle);
```

```
?>
```

Notez que **readdir()** retournera aussi les dossiers "." et "..". Si vous ne les voulez pas, supprimez les simplement :

Exemple 2. Liste tous les fichiers du dossier courant, sauf "." et ".."

```
<?php
$handle=opendir('.');
while ($file = readdir($handle)) {
    if ($file != "." && $file != "..") {
        echo "$file\n";
    }
}
closedir($handle);
?>
```

rewinddir (PHP 3, PHP 4 >= 4.0b1)

Retourne à la première entrée du dossier.

```
void rewinddir (resource dir_handle)
```

rewinddir() retourne à la première entrée du dossier identifié par *`dir_handle`* : le prochain fichier lu sera le premier.

XXII. DOM XML

Note importante : cette documentation est en cours de rédaction, et n'est pas encore finie. Elle souffre naturellement d'un manque de détails et de relecture. Soyez en prévenu. (Damien Seguy).

Ces fonctions ne sont disponibles que si PHP a été configuré avec l'option `--with-dom=[DIR]`, et utilise la librairie GNOME xml library. Vous aurez aussi besoin de la librairie libxml-2.2.7 (la version beta ne fonctionne pas). Ces fonctions ont été ajoutées en PHP 4.

Cette extension vous permet de générer des documents XML avec les API DOM. Elle fournit aussi une fonction **xmltree()** qui transforme un fichier XML en tableau PHP. Actuellement, ce tableau est accessible uniquement en lecture. Cela ne siginifie pas que vous ne pouvez pas le modifier, mais cela n'aurait aucun sens car **domxml_dumpmem()** ne pourra pas prendre ces modifications en considération. Par conséquent, si vous voulez lire un fichier XML et écrire sa version modifiée, utilisez les fonctions **domxml_add_node()**, **domxml_set_attribute()**, etc... et finalement, **domxml_dumpmem()**.

Ce module définit les constantes suivantes :

Tableau 1. Constantes XML

Constante	Valeur	Description
XML_ELEMENT_NODE	1	Le noeud est un élément
XML_ATTRIBUTE_NODE	2	Le noeud est un attribut
XML_TEXT_NODE	3	Le noeud est un texte
XML_CDATA_SECTION_NODE	4	
XML_ENTITY_REF_NODE	5	
XML_ENTITY_NODE	6	Le noeud est une entité telle que
XML_PI_NODE	7	Le noeud est une instruction
XML_COMMENT_NODE	8	Le noeud est un commentaire
XML_DOCUMENT_NODE	9	Le noeud est un document
XML_DOCUMENT_TYPE_NODE	10	
XML_DOCUMENT_FRAG_NODE	11	
XML_NOTATION_NODE	12	
XML_GLOBAL_NAMESPACE	1	
XML_LOCAL_NAMESPACE	2	

Chaque fonction de cette extension peut être utilisée de deux manières différentes. Dans un contexte procédural, il faut passer l'objet en premier argument; dans un contexte objet, vous pouvez appeler la fonction comme une méthode de cet objet. Cette documentation présente les fonctions dans leur contexte procédural. Vous pouvez connaître la méthode objet en supprimant le préfixe "`domxml_`". Les tables suivantes listes toutes les classes, leurs attributs et leurs méthodes.

Ce module définit un ensemble de classes, qui sont listées ci-dessous (y compris leur attributs et leur méthodes).

Tableau 2. classe DomDocument (méthodes)

Nom de la méthode	Nom de la fonction	Description
root	**domxml_root()**	
children	**domxml_children()**	
add_root	**domxml_add_root()**	
dtd	**domxml_intdtd()**	
dumpmem	**domxml_dumpmem()**	
xpath_init	xpath_init	
xpath_new_context	xpath_new_context	
xptr_new_context	xptr_new_context	

Tableau 3. Classe DomDocument (attributs)

Nom	Type	Description
doc	class DomDocument	L'objet lui-même
name	string	
url	string	
version	string	Version de XML
encoding	string	
standalone	long	1 si le fichier est complet
type	long	Une des constantes de la table ...
compression	long	1 si le fichier est compressé
charset	long	

Tableau 4. classe DomNode (méthodes)

Nom	Nom en PHP	Description
lastchild	**domxml_last_child()**	
children	**domxml_children()**	
parent	**domxml_parent()**	
new_child	**domxml_new_child()**	
get_attribute	**domxml_get_attribute()**	
set_attribute	**domxml_set_attribute()**	
attributes	**domxml_attributes()**	
node	**domxml_node()**	
set_content()	domxml_set_content	

Tableau 5. classe DomNode (attributs)

Nom	Type	Description
node	class DomNode	L'objet lui-même
type	long	
name	string	
content	string	

xmldoc (PHP 4 >= 4.0b4)

Crée un objet DOM pour un document XML.

```
object xmldoc (string str)
```

xmldoc() analyse le document XML *str* et retourne un objet de classe "Dom document", avec les propriétés de "doc" (ressources), "version" (string) et "type" (long).

xmldocfile (PHP 4 >= 4.0b4)

Crée un objet DOM à partir d'un fichier XML

```
object xmldocfile (string filename)
```

xmldocfile() analyse le fichier XML *filename* et retourne un objet "Dom document", avec les propriétées de "doc" (ressources) et "version" (string).

xmltree (PHP 4 >= 4.0b4)

Crée un arbre d'objet PHP, à partir d'un document XML.

```
object xmltree (string str)
```

xmltree() analyse le document XML *str* et retourne un arbre d'objets PHP qui représente le document analysé. **xmltree()** est différentes des autres fonctions, car vous ne pouvez accéder à cet arbre avec aucune des autres fonctions. Modifier cet arbre n'a pas de sens, car il n'y a pas moyen de sauver ces modifications. Cette fonction a tout de même des applications en lecture seule.

domxml_root (PHP 4 >= 4.0b4)

Retourne l'élément racine

```
object domxml_root (object doc)
```

domxml_root() prend en argument *doc*, un objet de la classe "Dom document", et retourn l'élément racine de ce document. Les autres noeuds qui peuvent être considérés comme racine (tels que les commentaires) sont ignorés.

L'exemple suivant retourne simplement l'élément CHAPTER et l'affiche. Les autres racines (des commentaires) ne sont pas retournés.

Exemple 1. Lecture de l'élément principal

```
<?php
  $xmlstr = "<?xml version='1.0' standalone='yes'>
  <!DOCTYPE chapter SYSTEM '/share/sgml/Norman_Walsh/db3xml10/db3xml10.dtd'
  [ <!ENTITY sp \"spanish\">
  ]>
  <!- lsfj  -->
  <chapter language='en'><title language='en'>Title</title>
   <para language='ge'>
    &sp;
    <!- comment -->
    <informaltable language='&sp;'>
```

```
    <tgroup cols='3'>
     <tbody>
      <row><entry>a1</entry><entry
  morerows='1'>b1</entry><entry>c1</entry></row>
  <row><entry>a2</entry><entry>c2</entry></row>
      <row><entry>a3</entry><entry>b3</entry><entry>c3</entry></row>
     </tbody>
    </tgroup>
   </informaltable>
  </para>
  </chapter>";
  if(!$dom = xmldoc($xmlstr)) {
    echo "Erreur lors de l'analyse du document\n";
    exit;
  }
  $root = $dom->root();
  /* ou $root = domxml_root($dom); */
  print_r($root);
?>
```

domxml_add_root (PHP 4 >= 4.0b4)

Ajoute une autre racine

```
resource domxml_add_root (resource doc, string name)
```

domxml_add_root() ajoute la racine *name* au document *doc*.

Exemple 1. Création d'une en-tête HTML simple

```
<?php
  $root = $doc->add_root("HTML");
  $head = $root->new_child("HEAD", "");
  $head->new_child("TITLE", "Ici, le titre");
  echo $doc->dumpmem();
?>
```

domxml_dumpmem (PHP 4 >= 4.0b4)

Ecrit le document XML interne dans une chaîne

```
string domxml_dumpmem (resource doc)
```

domxml_dumpmem() crée un document XML à partir de la représentation interne. **domxml_dumpmem()** est généralement appelée avec avoir construit un nouveau document XML, comme dans l'exemple **domxml_add_root()**.

Voir aussi **domxml_add_root()**.

domxml_attributes (PHP 4 >= 4.0b4)

Retourne les attributs d'un noeud

```
array domxml_attributes (resource node)
```

domxml_attributes() retourne tous les attributs du noeud *node* sous forme d'un tableau d'objets "dom attribute".

domxml_get_attribute (PHP 4 >= 4.0.5)

Retourne un attribut d'un noeud

```
object domxml_get_attribute (resource node, string name)
```

domxml_get_attribute() retourn l'attribut *name* du noeud *node*.

Voir aussi **domxml_set_attribute()**.

domxml_set_attribute (PHP 4 >= 4.0.5)

Modifie un attribut

```
object domxml_set_attribute (resource node, string name, string value)
```

domxml_set_attribute() modifie l'attribut *name* du noeud *node* en lui attribuant la valeur *value*.

En partant de l'exemple proposé à la fonction **domxml_add_root()**, il est simple d'ajouter un attribut à l'élément HEAD en appelant simplement **set_attribute()**.

Exemple 1. Ajouter un attribut à un élément

```
<?php
  $doc = new_xmldoc("1.0");
  $root = $doc->add_root("HTML");
  $head = $root->new_child("HEAD", "");
  $head->new_child("TITLE", "Ici, le titre");
  $head->set_attribute("Language", "fr");
  $head->new_child("TITLE", "Hier der Titel");
  $head->set_attribute("Language", "ge");
  echo $doc->dumpmem();
?>
```

domxml_children (PHP 4 >= 4.0b4)

Retourne les fils d'un noeud

```
array domxml_children (object doc|node)
```

domxml_children() retourne tous les fils du noeud *doc|node*, sous forme d'un tableau de noeuds.

Dans l'exemple ci-dessous, la variable `children` contiendra un tableau avec les noeuds de type XML_ELEMENT. Ce noeud est l'élément TITLE.

Exemple 1. Lire les fils d'un noeud

```
<?php
  $doc = new_xmldoc("1.0");
  $root = $doc->add_root("HTML");
  $head = $root->new_child("HEAD", "");
  $head->new_child("TITLE", "Hier der Titel");
  $head->set_attribute("Language", "ge");
  $children = $head->children()
?>
```

domxml_new_child (PHP 4 >= 4.0b4)

Ajoute un nouveau fils

```
resource domxml_new_child (string name, string content)
```

domxml_new_child() ajoute un nouveau fils. (NDtraducteur : cette documentation n'est pas encore finie...)

domxml_new_xmldoc (PHP 4 >= 4.0b4)

Crée un document XML vide

```
object domxml_new_xmldoc (string version)
```

domxml_new_xmldoc() crée un nouveau document XML vide, et le retourne.

Voir aussi **domxml_add_root()**.

xpath_new_context (4.0.4 - 4.0.6 only)

Crée un nouveau contexte xpath

```
object xpath_new_context (object dom_document)
```

Pas de documentation encore (22/2/2201).

xpath_eval (PHP 4 >= 4.0.4)

Evalue une expression xpath

```
array xpath_eval (object xpath_context)
```

Pas de documentation encore (22/2/2201).

XXIII. Gestion des erreurs

Ces fonctions permettent de gérer les erreurs, et de les enregistrer. Vous pouvez définir les règles de traitement des erreurs et choisir la manière de les enregistrer : vous pouvez adapter le rapport d'erreur à vos besoins.

Avec les fonctions d'enregistrements, vous pouvez envoyer directement les rapport à d'autres machines (ou même les envoyer par email à un pager), à l' historique système, ou encore selectionner les erreurs les plus importantes et ne pas enregistrer les autres.

La fonction de niveau d'erreur vous permet de personnaliser le niveau et le type d'erreur noté : depuis les inoffensives alertes jusqu'au erreurs personnalisées retournées par les fonctions.

error_log (PHP 3, PHP 4 >= 4.0b1)

Envoie un message d'erreur quelque part

```
int error_log (string message, int message_type [, string destination [, string
extra_headers]])
```

error_log() envoie un message d'erreur à l'historique du serveur web, à un port TCP ou un fichier. *message* est le message d'erreur qui doit être enregistré. *message_type* indique où le message doit être envoyé :

Tableau 1. Types de error_log()

0	*message* est envoyé à l'historique PHP, qui est basé sur l'historique système ou un fichier, en fonction de la configuration de error_log.
1	*message* est envoyé par email à l'adresse *destination*. C'est le seul type qui utilise le quatrième paramètre *extra_headers*. Ce message utilise la même fonction interne que **mail()**.
2	*message* est envoyé par la connexion de debuggage PHP. Cette option n'est disponible que si l'option remote debugging a été désactivé. Dans ce cas, le parmètre *destination* spécifie l'hôte ou l'adresse IP, et optionnellement le numéro de port, de la socket qui recevra les informations de débuggage.
3	*message* est ajouté au fichier *destination*.

Exemple 1. Exemples avec error_log()

```
<?php
// Envoi une notification par l'historique du serveur, si la connexion à la base
// de données est impossible.
if (!Ora_Logon ($username, $password)) {
    error_log ("Base Oracle indisponible!", 0);
}
// Indiquer à l'administrateur, par email, qu'il n'y a plus de FOO
if (!($foo = allocate_new_foo()) {
    error_log ("Aya!, Il ne reste plus de FOO disponibles!", 1,
               "operateur@mondomaine.com");
}
// D'autres manières d'appeler error_log():
error_log ("Grosse bourde!", 2, "127.0.0.1:7000");
error_log ("Grosse bourde!", 2, "loghost");
error_log ("Grosse bourde!", 3, "/var/tmp/my-errors.log");
?>
```

error_reporting (PHP 3, PHP 4 >= 4.0b1)

Fixe le niveau de rapport d'erreurs PHP

```
int error_reporting ([int level])
```

error_reporting() fixe le niveau de rapport d'erreur PHP et retourne l'ancienne valeur. Le niveau d'erreur peut être un champs de bits, ou une constante. L'utilisation des constantes est vivement recommandé, pour assurer une compatiblité maximale avec les futures versions. Au fur et à mesure que de nouveaux niveaux d'erreurs sont créés, l'intervalle de validité des niveaux évolue, et les anciennes valeurs n'ont plus les mêmes significations.

Exemple 1. Exemple de modification de niveau d'erreur

```
error_reporting (55);   // En PHP 3, équivalent à E_ALL ^ E_NOTICE
/* ...en PHP 4, '55' signifie (E_ERROR | E_WARNING | E_PARSE |
E_CORE_ERROR | E_CORE_WARNING) */
error_reporting (2039); // PHP 4 équivalent à E_ALL ^ E_NOTICE
error_reporting (E_ALL ^ E_NOTICE); // La même signification en PHP 3 et 4
```

Suivez les liens de chaque valeur interne pour connaître leur signification :

Tableau 1. Constantes avec error_reporting()

constante	valeur
1	E_ERROR
2	E_WARNING
4	E_PARSE
8	E_NOTICE
16	E_CORE_ERROR
32	E_CORE_WARNING
64	E_COMPILE_ERROR
128	E_COMPILE_WARNING
256	E_USER_ERROR
512	E_USER_WARNING
1024	E_USER_NOTICE

Exemple 2. Exemples avec error_reporting()

```
error_reporting(0);
/* Empêche tout affichage d'erreur */
error_reporting(7); // Ancienne syntaxe PHP 2/3
error_reporting(E_ERROR | E_WARNING | E_PARSE); // Nouvelle syntaxe PHP 3/4
/* Utilisation appropriée pour les erreurs courantes d'exécution */
error_reporting(15); // Ancienne syntaxe, PHP 2/3
error_reporting(E_ERROR | E_WARNING | E_PARSE | E_NOTICE); // Nouvelle syntaxe PHP 3/4
/*  Utilisation appropriée pour les erreurs courantes de développement
 (variables non initialisées..)*/
error_reporting(63); // Ancienne syntaxe, PHP 2/3
error_reporting(E_ALL); // Nouvelle syntaxe PHP 3/4
/* rapporte toutes les erreurs PHP*/
```

restore_error_handler (PHP 4 >= 4.0.1)

Réactive l'ancienne fonction de gestion des erreurs

```
void restore_error_handler (void)
```

Utilisée après avoir modifié la fonction de gestion des erreurs, grâce à **set_error_handler()**, **restore_error_handler()** permet de réutiliser l'ancienne version de gestion des erreurs (qui peut être la fonction PHP par défaut, ou une autre fonction utilisateur).

Voir aussi **error_reporting()**, **set_error_handler()**, **trigger_error()** et **user_error()**

set_error_handler (PHP 4 >= 4.0.1)

Choisi une fonction utilisateur comme gestionnaire d'erreurs

```
string set_error_handler (string error_handler)
```

set_error_handler() choisit la fonction utilisateur *error_handler* pour gérer les erreurs dans un script. Retourne un pointeur sur l'ancienne fonction de gestion des erreurs (si il y en avait une), ou FALSE, en cas d'erreur. **set_error_handler()** sert à définir votre propre gestionnaire d'erreur, qui prendra en charge leur traitement durant l'exécution d'un script. Cela peut être utile lorsque vous devez repérer des erreurs critiques lors d'un nettoyage de bases, ou bien si vous souhaitez générer une erreur dans certaines conditions (avec **trigger_error()**).

La fonction utilisateur doit accepter deux arguments : le code de l'erreur, et une chaîne décrivant l'erreur. L'exemple ci dessous montre le traitement d'exceptions en déclenchant des erreurs, et en les gérant avec une fonction utilisateur :

Exemple 1. Traitement des erreurs avec set_error_handler() et trigger_error()

```
<?php
// redéfinit les constantes utilisateurs - PHP 4 seulement
define (FATAL,E_USER_ERROR);
define (ERROR,E_USER_WARNING);
define (WARNING,E_USER_NOTICE);
// Fixe le niveau de rapport d'erreur pour ce script
error_reporting  (FATAL + ERROR + WARNING);
// Fonction de traitement des erreurs
function myErrorHandler ($errno, $errstr) {
    switch ($errno) {
    case FATAL:
    echo "<B>FATAL</B> [$errno] $errstr<br>\n";
    echo "  Erreur fatale à la ligne ".__LINE__." du fichier ".__FILE__;
    echo ", PHP ".PHP_VERSION." (".PHP_OS.")<br>\n";
    echo "Aborting...<br>\n";
    exit -1;
    break;
    case ERROR:
    echo "<B>ERREUR</B> [$errno] $errstr<br>\n";
    break;
    case WARNING:
    echo "<B>ALERTE</B> [$errno] $errstr<br>\n";
    break;
    default:
    echo "Erreur inconnue de type : [$errno] $errstr<br>\n";
    break;
    }
}
// fonction qui teste la gestion d'erreur
function scale_by_log ($vect, $scale) {
    if ( !is_numeric($scale) || $scale <= 0 )
    trigger_error("log(x) pour x <= 0 est indéfini, vous avez passé: scale = $scale",
      FATAL);
    if (!is_array($vect)) {
    trigger_error("Vecteur d'entrée incorrect : un tableau de valeurs est at-
tendu : ", ERROR);
    return null;
    }
    for ($i=0; $i<count($vect); $i++) {
    if (!is_numeric($vect[$i]))
```

```
    trigger_error("La valeur à la position $i n'est pas un nombre. On uti-
lise 0 (zéro) à la place",
      WARNING);
    $temp[$i] = log($scale) * $vect[$i];
    }
    return $temp;
}
// Ancienne fonction de traitement des erreurs
$old_error_handler = set_error_handler("myErrorHandler");
// Génération de quelques erreurs : définition d'un tableau avec des élé-
ments non numériques
echo "vector a\n";
$a = array(2,3,"foo",5.5,43.3,21.11);
print_r($a);
// définition d'un deuxième table à problème
echo "---\nvector b - a alerte (b = log(PI) * a)\n";
$b = scale_by_log($a, M_PI);
print_r($b);
// Ceci est un problème, on passe une chaîne à la place d'un tableau
echo "---\nvector c - une erreur\n";
$c = scale_by_log("not array",2.3);
var_dump($c);
// Ceci est critique : le tableau contient des valeurs négatives
echo "---\nvector d - fatal error\n";
$d = scale_by_log($a, -2.5);
?>
```

L'éxécution du script devrait donner ceci :

```
vector a
Array
(
    [0] => 2
    [1] => 3
    [2] => foo
    [3] => 5.5
    [4] => 43.3
    [5] => 21.11
)
--
vector b - une alerte (b = log(PI) * a)
<B>WARNING</B> [1024] La valeur à la position 2 n'est pas un nombre. On uti-
lise 0 (zéro) à la place<br>
Array
(
    [0] => 2.2894597716988
    [1] => 3.4341896575482
    [2] => 0
    [3] => 6.2960143721717
    [4] => 49.566804057279
    [5] => 24.165247890281
)
--
vector c - an error
<B>ERROR</B> [512] Vecteur d'entrée incorrect : un tableau de valeur est attendu<br>
NULL
--
vector d - fatal error
<B>FATAL</B> [256] log(x) de x <= 0 est indéfini : scale = -2.5<br>
Erreur fatale à la ligne 16 du fichier trigger_error.php, PHP 4.0.1pl2 (Linux)<br>
Annulation du script....<br>
```

Il faut se rappeler que la fonction standard de traitement des erreurs de PHP est alors complètement ignorée. **error_reporting()** n'aura plus d'effet, et votre fonction de gestion des erreurs sera toujours appelée. Vous pourrez toujours lire la valeur de l'erreur courante de **error_reporting()** et faire réagier la fonction de gestion des erreurs en fonction. Cette remarque est notamment valable si la commande a été préfixée par @ (0 sera retourné).

Notez aussi qu'il est alors confié à cette fonction de terminer le script (**die()**) si nécessaire. Si la fonction de gestion des erreurs se termine normalement, l'exécution du script se poursuivra avec l'exécution de la prochaine commande.

Voir aussi **error_reporting()**, **restore_error_handler()**, **trigger_error()**, et **user_error()**

trigger_error (PHP 4 >= 4.0.1)

Déclenche une erreur utilisateur

```
void trigger_error (string error_msg [, int error_type])
```

trigger_error() est utilisé pour déclencher une erreur utilisateur. Elle peut aussi être utilisée en conjonction avec un gestionnaire d'erreur interne, ou un gestionnaire d'erreurs utilisateur qui a été choisi comme gestionnaire d'erreur avec **set_error_handler()**.

trigger_error() est pratique lorsque vous devez générer une réponse particulière lors de l'exécution. Par exemple

```
<?php
if (assert ($divisor == 0))
   trigger_error ("Impossible de diviser par zéro", E_USER_ERROR);
?>
```

Note : Voir **set_error_handler()** pour illustration.

Voir aussi **error_reporting()**, **set_error_handler()**, **restore_error_handler()**, **user_error()**

user_error (PHP 4 >= 4.0RC2)

Génére un message d'erreur utilisateur

```
void user_error (string error_msg [, int error_type])
```

user_error() est un alias de la fonction **trigger_error()**.

Voir aussi **error_reporting()**, **set_error_handler()**, **restore_error_handler()** et **trigger_error()**.

XXIV. FrontBase

Avertissement

Ce module est *EXPERIMENTAL*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

Ces fonctions vous permettent d'accéder aux serveurs SQL FrontBase. Pour pouvoir les utiliser, vous devez compiler PHP avec le support fbsql en utilisant l'option `--with-fbsql`. Si vous utilisez cette option sans spécifier le chemin jusqu'à l'installation fbsql, PHP recherchera les librairies du client fbsql dans les dossiers habituels, sur votre système. Les utilisateurs qui ont installé FrontBase dans un dossier non standard doivent spécifier le chemin comme ceci : `--with-fbsql=/path/to/fbsql`. Cela va indiquer à PHP le bon emplacement des librairies de FrontBase, et éviter les conflits.

Plus d'informations sur FrontBase sont disponibles à http://www.frontbase.com/.

La documentation complète de FrontBase est disponible à http://www.frontbase.com/cgi-bin/WebObjects/FrontBase.woa/wa/productsPage?currentPage=Documentation.

Le support de Frontbase a été ajouté en PHP 4.0.6.

fbsql_affected_rows (PHP 4 >= 4.0.6)

Lit le nombre de ligne affectées par la dernière requête

```
int fbsql_affected_rows (resource [link_identifier])
```

fbsql_affected_rows() retourne le nombre de lignes affectées par la dernière requête INSERT, UPDATE ou DELETE, effectuée avec la connexion représentée par *link_identifier*. Si ce dernier n'est pas spécifié, c'est la dernière connexion ouverte par **fbsql_connect()** qui sera utilisée.

> **Note :** Si vous utilisez les transactions, vous devez appeler **fbsql_affected_rows()** après votre requête INSERT, UPDATE ou DELETE, mais pas après la validation.

Si la dernière requête DELETE ne contenait pas de clause WHERE, toutes les lignes seront effacées, mais **fbsql_affected_rows()** retournera 0.

> **Note :** Lors d'une requête UPDATE, FrontBase ne modifie pas les lignes dont les anciennes valeurs sont égales aux nouvelles. Cela fait que **fbsql_affected_rows()** ne retournera pas le nombre de ligne traitées, mais le nombre de lignes affectées (modifiées) par la requête.

Si la dernière requête échoue, **fbsql_affected_rows()** retourne -1.

Voir aussi **fbsql_num_rows()**.

fbsql_autocommit (PHP 4 >= 4.0.6)

Active ou désactive la validation automatique.

```
boolean fbsql_autocommit (resource link_identifier, boolean [OnOff])
```

fbsql_autocommit() retourne l'état courant de la validation automatique, pour la connexion *link_identifier*. Si le paramètre *OnOff* est fourni, la validation automatique sera remplacée par sa valeur (un booléen).

fbsql_change_user (unknown)

Change le nom d'utilisateur de la session active

```
resource fbsql_change_user (string user, string password, string [database], resource
[link_identifier])
```

fbsql_change_user() change le nom de l'utilisateur courant sur la session active courante, ou sur *link_identifier*. Si une base de données est spécifiée avec le paramètre *database*, elle deviendra la base par défaut du nouvel utilisateur. Le nouvel utilisateur doit être spécifié par son login (*user*), et son mot de passe (*password*). Si l'authentification échoue, la session courante restera ouverte.

fbsql_close (PHP 4 >= 4.0.6)

Ferme la connexion FrontBase

```
boolean fbsql_close (resource [link_identifier])
```

fbsql_close() retourne `TRUE` en cas de succès et `FALSE` en cas d'erreur.

fbsql_close() ferme la connexion au serveur FrontBase associé à la ressource *`link_identifier`*. Si *`link_identifier`* est omis, c'est la dernière connexion ouverte qui sera fermée.

Utiliser **fbsql_close()** n'est pas nécessaire, car les liens non persistants seront automatiquement fermé à la fin du script.

Exemple 1. Exemple avec fbsql_close()

```
<?php
    $link = fbsql_connect("localhost", "_SYSTEM", "secret")
        or die("Could not connect");
    print("Connecté!");
    fbsql_close($link);
?>
```

Voir aussi **fbsql_connect()** et **fbsql_pconnect()**.

fbsql_connect (PHP 4 >= 4.0.6)

Ouvre une connexion à un serveur FrontBase

```
resource fbsql_connect (string [hostname], string [username], string [password])
```

fbsql_connect() retourne une ressource de connexion positive en cas de succès, ou un message d'erreur en cas d'échec.

fbsql_connect() établit une connexion avec un serveur FrontBase. Les valeur suivantes sont utilisées, en cas d'omission : *`hostname`* = 'NULL', *`username`* = '_SYSTEM' et *`password`* = "" (pas de mot de passe).

Si un deuxième appel est fait à **fbsql_connect()** avec les mêmes arguments, une nouvelle connexion ne sera pas générée, mais la connexion déjà ouverte sera reutilisée, et retournée.

La connexion au serveur sera fermée dès la fin du scrpit, à moins qu'elle ne soit explicitement terminée plus tôt, avec la fonction **fbsql_close()**.

Exemple 1. Exemple avec fbsql_connect()

```
<?php
    $link = fbsql_connect("localhost", "_SYSTEM", "secret")
        or die("Could not connect");
    print("Connected successfully");
    fbsql_close($link);
?>
```

Voir aussi **fbsql_pconnect()** et **fbsql_close()**.

fbsql_create_db (PHP 4 >= 4.0.6)

Crée une base de données

```
boolean fbsql_create_db (string database_name, resource [link_identifier])
```

fbsql_create_db() crée une nouvelle base de données, nommée *`database_name`*, sur le serveur repéré par la ressource *`link_identifier`*.

Exemple 1. Exemle avec fbsql_create_db()

```
<?php
    $link = fbsql_pconnect("localhost", "_SYSTEM", "secret")
        or die ("Impossible de se connecter");
    if (fbsql_create_db("my_db")) {
        print("Base de données créée!\n");
    } else {
        printf("Erreur de création de la base de données : %s\n", fbsql_error());
    }
?>
```

Voir aussi **fbsql_drop_db()**.

fbsql_data_seek (PHP 4 >= 4.0.6)

Déplace le pointeur interne de résultat

```
int fbsql_data_seek (int result_identifier, int row_number)
```

fbsql_data_seek() retourne `TRUE` en cas de succès et `FALSE` en cas d'erreur.

fbsql_data_seek() déplace le pointeur interne de ligne dans le résultat de requête *`result_identifier`* jusqu'à la ligne *`row_number`*. Le prochain appel à **fbsql_fetch_row()** retournera cette ligne.

Les lignes sont numérotées à partir de 0.

Exemple 1. Exemple avec fbsql_data_seek()

```
<?php
    $link = fbsql_pconnect("localhost", "_SYSTEM", "secret")
        or die ("Impossible de se connecter");
    fbsql_select_db("samp_db")
        or die ("Impossible de sélectionner une base");
    $query = "SELECT last_name, first_name FROM friends;";
    $result = fbsql_query($query)
        or die ("Query failed");
// Lecture des lignes en ordre inverse
    for ($i = fbsql_num_rows($result) - 1; $i >=0; $i--) {
        if (!fbsql_data_seek($result, $i)) {
            printf ("Impossible d'accéder à la ligne %d\n", $i);
            continue;
        }
        if(!($row = fbsql_fetch_object($result)))
            continue;
        printf ("%s %s<BR>\n", $row->last_name, $row->first_name);
    }
    fbsql_free_result($result);
?>
```

fbsql_db_query (PHP 4 >= 4.0.6)

Envoie une requête à la base FrontBase

```
resource fbsql_db_query (string database, string query, resource [link_identifier])
```

fbsql_db_query() retourne une ressource positive représentant un résultat de requête en cas de succès, et `FALSE` en cas d'erreur.

fbsql_db_query() sélectionne la base *database* et y exécute la requête *query*. Si le paramètre optionnel *link_identifier* est spécifié, **fbsql_db_query()** travaillera sur cette connexion. S'il est omis, **fbsql_db_query()** essaiera d'utiliser la dernière connexion ouverte. Si aucune connexion n'a été ouverte, **fbsql_db_query()** essaiera de se connecter automatiquement en appelant la fonction **fbsql_connect()**, sans arguments.

Voir aussi **fbsql_connect()**.

fbsql_drop_db (PHP 4 >= 4.0.6)

Supprime une base de données FrontBase

```
boolean fbsql_drop_db (string database_name, resource [link_identifier])
```

fbsql_drop_db() retourne TRUE en cas de succès et FALSE en cas d'erreur.

fbsql_drop_db() essaie de supprimer la base de données *database_name*, sur la connexion représentée par *link_identifier*.

fbsql_errno (PHP 4 >= 4.0.6)

Retourne le code d'erreur FrontBase

```
int fbsql_errno (resource [link_identifier])
```

fbsql_errno() retourne le code d'erreur de la dernière connexion FrontBase, ou bien 0 (zéro) si aucune erreur n'est survenue.

Les erreurs générées par FrontBase ne sont pas automatiquement affichées comme alertes. Il faut utiliser la fonction **fbsql_errno()** pour connaître leur code d'erreur. Notez que cette fonction ne retourne que le code d'erreur généré par la dernière fonction FrontBase (hormis **fbsql_error()** et **fbsql_errno()**) : si vous voulez repérer les erreurs, faites le dès que les fonctions ont été appelées.

```
<?php
  fbsql_connect("marliesle");
  echo fbsql_errno().": ".fbsql_error()."<br>";
  fbsql_select_db("nonexistentdb");
  echo fbsql_errno().": ".fbsql_error()."<br>";
  $conn = fbsql_query("SELECT * FROM nonexistenttable;");
  echo fbsql_errno().": ".fbsql_error()."<br>";
?>
```

Voir aussi **fbsql_error()** et **fbsql_warnings()**

fbsql_error (PHP 4 >= 4.0.6)

Retourne le message d'erreur FrontBase

```
string fbsql_error (resource [link_identifier])
```

fbsql_error() retourne le dernier message d'erreur généré par le serveur FrontBase, ou bien " (chaîne vide) si aucune erreur n'est survenue.

Les erreurs générées par FrontBase ne sont pas automatiquement affichées comme alertes. Il faut utiliser la fonction **fbsql_errno()** pour connaître leur code d'erreur. Notez que cette fonction ne retourne que le code d'erreur généré par la

dernière fonction FrontBase (hormis **fbsql_error()** et **fbsql_errno()**) : si vous voulez repérer les erreurs, faites le dès que les fonctions ont été appelées.

```
<?php
  fbsql_connect("marliesle");
  echo fbsql_errno().": ".fbsql_error()."<br>";
  fbsql_select_db("nonexistentdb");
  echo fbsql_errno().": ".fbsql_error()."<br>";
  $conn = fbsql_query("SELECT * FROM nonexistenttable;");
  echo fbsql_errno().": ".fbsql_error()."<br>";
?>
```

Voir aussi **fbsql_errno()** et **fbsql_warnings()**

fbsql_fetch_array (PHP 4 >= 4.0.6)

Lit toute une ligne de résultat dans un tableau.

```
array fbsql_fetch_array (resource result, int [result_type])
```

fbsql_fetch_array() retourne un tableau contenant la ligne courante du résultat *result*, ou FALSE s'il n'y a plus de lignes.

fbsql_fetch_array() est une version améliorée de **fbsql_fetch_row()**. En plus de stocker les données dans un tableau à indice numérique, elle les stocke aussi sous forme de tableau associatif, dont les indices sont les noms des colonnes.

SI deux colonnes (ou plus) on le même nom, la dernière colonne sera utilisée. Pour accéder aux autres colonnes de même nom, vous devez absolument utiliser les indices numériques.

```
select t1.f1 as foo t2.f1 as bar from t1, t2;
```

Il est important de noter que **fbsql_fetch_array()** n'est pas significativement plus lent que **fbsql_fetch_row()**, tandis qu'elle apporte un confort d'utilisation notable.

Le second argument optionnel *result_type* de **fbsql_fetch_array()** est une constante qui peut prendre l'une des valeurs suivantes : FBSQL_ASSOC, FBSQL_NUM et FBSQL_BOTH.

Pour plus de détails, reportez-vous à **fbsql_fetch_row()** et **fbsql_fetch_assoc()**.

Exemple 1. Exemple avec fbsql_fetch_array()

```
<?php
  fbsql_connect($host, $user, $password);
  $result = fbsql_db_query("database","select user_id, fullname from table");
  while ($row = fbsql_fetch_array($result)) {
    echo "user_id: ".$row["user_id"]."<br>\n";
    echo "user_id: ".$row[0]."<br>\n";
    echo "fullname: ".$row["fullname"]."<br>\n";
    echo "fullname: ".$row[1]."<br>\n";
  }
  fbsql_free_result($result);
?>
```

fbsql_fetch_assoc (PHP 4 >= 4.0.6)

Lit toute une ligne de résultat dans un tableau associatif

```
array fbsql_fetch_assoc (resource result)
```

fbsql_fetch_assoc() retourne un tableau associatif contenant la ligne courante du résultat *result*, ou FALSE s'il n'y a plus de lignes.

fbsql_fetch_assoc() est équivalent à **fbsql_fetch_array()** avec l'option FBSQL_ASSOC. Elle ne retourne qu'un tableau associatif. C'est le comportement initial de **fbsql_fetch_array()**. Si vous avez aussi besoin des indices numériques, utilisez **fbsql_fetch_array()**.

SI deux colonnes (ou plus) on le même nom, la dernière colonne sera utilisée. Pour accéder aux autres colonnes de même nom, vous devez absolument utiliser la fonction **fbsql_fetch_array()**.

Il est important de noter que **fbsql_fetch_assoc()** n'est pas significativement plus lent que **fbsql_fetch_row()**, tandis qu'elle apporte un confort d'utilisation notable.

Pour plus de détails, reportez-vous à **fbsql_fetch_row()** et **fbsql_fetch_array()**.

Exemple 1. Exemple avec fbsql_fetch_assoc()

```
<?php
  fbsql_connect($host, $user, $password);
  $result = fbsql_db_query("database","select * from table;");
  while ($row = fbsql_fetch_assoc($result)) {
    echo $row["user_id"];
    echo $row["fullname"];
  }
  fbsql_free_result($result);
?>
```

fbsql_fetch_field (PHP 4 >= 4.0.6)

Lit des informations sur une colonne dans un résultat, et retourne un objet

```
object fbsql_fetch_field (resource result, int [field_offset])
```

fbsql_fetch_field() retourne une objet contenant les informations sur un champs, dans le résultat *result*.

fbsql_fetch_field() sert à lire des informations sur les champs dans le résultat *result*. Si le second paramètre *field_offset* n'est pas spécifié, le champs suivant est lu.

Les propriétés de l'objet sont :

- name - Nom de colonne
- table - Nom de la table d'origine
- max_length - Taille maximale de la colonne
- not_null - 1 si la colonne ne peut être nulle
- type - Type de la colonne

Exemple 1. Exemple avec fbsql_fetch_field()

```
<?php
  fbsql_connect($host, $user, $password)
    or die ("Impossible de se connecter");
  $result = fbsql_db_query("database", "select * from table;")
    or die ("Query failed");
// lire les données de colonnes
  $i = 0;
  while ($i < fbsql_num_fields($result)) {
    echo "Information de la colonne $i:<br>\n";
    $meta = fbsql_fetch_field($result);
    if (!$meta) {
        echo "Aucune information disponible<br>\n";
    }
    echo "<PRE>
max_length:   $meta->max_length
name:         $meta->name
not_null:     $meta->not_null
table:        $meta->table
type:         $meta->type
</PRE>";
    $i++;
  }
  fbsql_free_result($result);
?>
```

Voir aussi **fbsql_field_seek()**.

fbsql_fetch_lengths (PHP 4 >= 4.0.6)

Lit la taille de chaque colonne d'un résultat

```
array fbsql_fetch_lengths ([resource result])
```

fbsql_fetch_lengths() retourne un tableau contenant les tailles maximales de chaque champs, dans la dernière ligne lue par **fbsql_fetch_row()** ou FALSE en cas d'erreur.

fbsql_fetch_lengths() stocke les tailles de chaque ligne de résultat retourné par **fbsql_fetch_row()**, **fbsql_fetch_array()** et **fbsql_fetch_object()** dans un tableau à indices numériques, commençant à 0.

Voir aussi **fbsql_fetch_row()**.

fbsql_fetch_object (PHP 4 >= 4.0.6)

Lit une ligne de résultat sous forme d'objet

```
object fbsql_fetch_object (resource result, int [result_type])
```

fbsql_fetch_object() retourne un objet dont les propriétés représentent les colonnes de la ligne à lire, dans le résultat *result*, ou FALSE s'il n'y a pas de ligne à lire.

fbsql_fetch_object() est similaire à **fbsql_fetch_array()**, à la différence qu'elle retourne un objet. Nous ne pouvez alors accéder aux données qu'avec les noms des colonnes, et sous la forme de membre d'objets, et non plus avec leurs offset (les nombres ne peuvent représenter un membre d'objet).

Le second argument optionnel *result_type* de **fbsql_fetch_array()** est une constante qui peut prendre l'une des valeurs suivantes : FBSQL_ASSOC, FBSQL_NUM et FBSQL_BOTH.

En terme de vitesse, cette fonction est identique à **fbsql_fetch_array()** et presque aussi rapide que **fbsql_fetch_row()** (la différence n'est pas significative).

Exemple 1. Exemple avec fbsql_fetch_object()

```
<?php
  fbsql_connect($host, $user, $password);
  $result = fbsql_db_query("database", "select * from table;");
  while ($row = fbsql_fetch_object($result)) {
    echo $row->user_id;
    echo $row->fullname;
  }
  fbsql_free_result($result);
?>
```

Voir aussi **fbsql_fetch_array()** et **fbsql_fetch_row()**.

fbsql_fetch_row (PHP 4 >= 4.0.6)

Lit une ligne de résultat sous forme de tableau numérique

```
array fbsql_fetch_row (resource result)
```

fbsql_fetch_row() retourne un tableau représentant la ligne courant dans le résultat *result*, ou bien FALSE s'il n'y a plus de lignes à lire.

fbsql_fetch_row() lit une ligne de données dans le résultat *result*, et crée un tableau numérique. Chaque colonne est stockés dans un élément du tableau, dans le même ordre que dans le résultat. Les indices commencent à 0.

Le prochain appel à **fbsql_fetch_row()** va lire la prochaine ligne, ou bien retourner FALSE s'il n'y a plus de lignes à lire.

Voir aussi **fbsql_fetch_array()**, **fbsql_fetch_object()**, **fbsql_data_seek()**, **fbsql_fetch_lengths()** et **fbsql_result()**.

fbsql_field_flags (PHP 4 >= 4.0.6)

Lit les options associé à une colonne de résultat

```
string fbsql_field_flags (resource result, int field_offset)
```

fbsql_field_flags() retourne les options du champs *field_offset*, dans le résultat *field_offset*. Les options sont retournées sous la forme d'un seul mot par option, séparées par des espaces, de façons à faciliter la manipulation avec **explode()**.

fbsql_field_name (PHP 4 >= 4.0.6)

Lit le nom d'un champs

```
string fbsql_field_name (int result, int field_index)
```

fbsql_field_name() retourne le nom du champs numéro *field_index* dans le résultat *result*. *field_index* est le résultat de **fbsql_query()** et *field_index* est l'offset numérique du champs.

> **Note :** *field_index* commence à 0.
>
> e.g. L'index du troisème champs est 2, et l'index du quatrième champs est 3...

Exemple 1. Exemple avec fbsql_field_name()

```
<?php
// La tablea utilisateur est constituée de trois colonnes
//   user_id
//   username
//   password.
  $res = fbsql_db_query("users", "select * from users;", $link);
  echo fbsql_field_name($res, 0) . "\n";
  echo fbsql_field_name($res, 2);
?>
```

L'exemple ci-dessus va afficher :

```
user_id
password
```

fbsql_field_len (PHP 4 >= 4.0.6)

Retourne la taille d'un champs

```
int fbsql_field_len (resource result, int field_offset)
```

fbsql_field_len() retourne la taille du champs *`field_offset`* dans le résultat *`result`*.

fbsql_field_seek (PHP 4 >= 4.0.6)

Déplace le pointeur de résultat

```
boolean fbsql_field_seek (int result, int field_offset)
```

fbsql_field_seek() place le pointeur de colonne à la colonne *`field_offset`*. Si ce paramètre est omis, **fbsql_fetch_field()** retourne le numéro de colonne courant.

Voir aussi **fbsql_fetch_field()**.

fbsql_field_table (PHP 4 >= 4.0.6)

Lit le nom de la table d'origine d'un champs

```
string fbsql_field_table (resource result, int field_offset)
```

fbsql_field_table() retourn le nom de la table d'où est issue le champs d'offset *`field_offset`*. Les numéros de colonne commencent à 0.

fbsql_field_type (PHP 4 >= 4.0.6)

Lit le type d'une colonne

```
string fbsql_field_type (resource result, int field_offset)
```

fbsql_field_type() est similaire à la fonction **fbsql_field_name()**. Les arguments sont identiques, mais le type du champs est retourné. Il peut valoir "int", "real", "string", "blob" ou d'autres valeurs, comme décrit dans la documentation FrontBase (http://www.frontbase.com/cgi-bin/WebObjects/FrontBase.woa/wa/productsPage?currentPage=Documentation).

Exemple 1. Exemple avec fbsql_field_type()

```
<?php
  fbsql_connect("localhost:3306");
  fbsql_connect("localhost", "_SYSTEM", "");
  $result = fbsql_query("SELECT * FROM onek;");
  $fields = fbsql_num_fields($result);
  $rows   = fbsql_num_rows($result);
  $i = 0;
  $table = fbsql_field_table($result, $i);
  echo "Votre table '".$table."' a ".$fields." colonnes et ".$rows." lignes <br>";
  echo "La table dispose des champs suivants <br>";
  while ($i < $fields) {
    $type  = fbsql_field_type ($result, $i);
    $name  = fbsql_field_name ($result, $i);
    $len   = fbsql_field_len  ($result, $i);
    $flags = fbsql_field_flags($result, $i);
    echo $type." ".$name." ".$len." ".$flags."<br>";
    $i++;
  }
  fbsql_close();
?>
```

fbsql_free_result (PHP 4 >= 4.0.6)

Libère le résultat de la mémoire

```
bool fbsql_free_result (resource result)
```

fbsql_free_result() va libérer toute la mémoire utilisée par le résultat associé à la ressource *result*.

fbsql_free_result() n'a besoin d'être appelé que si vous craignez que votre script ne va consommer trop de mémoire, lorsqu'une requête retourne de très grand résultats. Toutes les ressources mémoire utilisées par le script sont de toutes manières libérées à la fin du script.

fbsql_insert_id (PHP 4 >= 4.0.6)

Lit le dernier identifiant généré par une requête INSERT

```
int fbsql_insert_id (resource [link_identifier])
```

fbsql_insert_id() retourne l'identifiant généré par la colonne de type DEFAULT UNIQUE, lors de la dernière requête INSERT, avec la connexion *link_identifier*. Si *link_identifier* est omis, la dernière connexion ouverte est utilisée.

fbsql_insert_id() retournera 0 si la dernière requête n'a pas généré de valeur dans la colonne DEFAULT UNIQUE Si vous devez sauver cette valeur pour plus tard, n'oubliez pas d'appeler **fbsql_insert_id()** tout de suite après la requête qui a généré cette valeur.

> **Note :** La valeur de la fonction FrontBase SQL "`LAST_INSERT_ID()`" retourne toujours la dernière valeur générée par DEFAULT UNIQUE et n'est jamais annulée entre les requêtes.

fbsql_list_dbs (PHP 4 >= 4.0.6)

Liste les bases de données

```
resource fbsql_list_dbs (resource [link_identifier])
```

fbsql_list_dbs() retourne un résultat contenant la liste des bases de données disponibles sur le serveur [link_identifier]. Utilisez la fonction **fbsql_tablename()** pour passer en revue ce résultat.

Exemple 1. Exemple avec fbsql_list_dbs()

```
<?php
  $link = fbsql_connect('localhost', 'myname', 'secret');
  $db_list = fbsql_list_dbs($link);
  while ($row = fbsql_fetch_object($db_list)) {
    echo $row->Database . "\n";
  }
?>
```

L'exemple ci-dessus va afficher ceci :

```
database1
database2
database3
..
```

> **Note :** L'exemple ci-dessus peut aussi bien fonctionner avec la fonction **fbsql_fetch_row()** ou toute autre similaire.

fbsql_list_fields (PHP 4 >= 4.0.6)

Liste les champs d'un résultat FrontBase

```
resource fbsql_list_fields (string database_name, string table_name, resource
[link_identifier])
```

fbsql_list_fields() lit les informations à propos de la table *`table_name`*, dans la base de données *`table_name`*, sur la connexion *`link_identifier`*. Un résultat de requête est retourné, et pourra être utilisé avec les fonctions **fbsql_field_flags()**, **fbsql_field_len()**, **fbsql_field_name()** et **fbsql_field_type()**.

Un identifiant de résultat est une ressource PHP, représentée par un entier positif. **fbsql_list_fields()** retourne -1 en cas d'erreur. Une chaîne décrivant l'erreur sera alors placée dans la variable `$phperrmsg`. Un message d'erreur sera aussi affiché, à moins que la fonction n'ai été appelée avec l'opérateur de suppression des erreurs @.

Exemple 1. Exemple avec fbsql_list_fields()

```
<?php
  $link = fbsql_connect('localhost', 'myname', 'secret');
  $fields = fbsql_list_fields("database1", "table1", $link);
  $columns = fbsql_num_fields($fields);
  for ($i = 0; $i < $columns; $i++) {
    echo fbsql_field_name($fields, $i) . "\n";;
  }
?>
```

L'exemple ci-dessus va afficher :

```
field1
field2
field3
..
```

fbsql_list_tables (PHP 4 >= 4.0.6)

Liste les tables dans une base de données FrontBase

```
resource fbsql_list_tables (string database, resource [link_identifier])
```

fbsql_list_tables() liste les tables dans la base de données *database*, et retourne un résultat, tout comme **fbsql_db_query()**. **fbsql_tablename()** sert à extraire la liste des tables dans ce résultat.

fbsql_next_result (PHP 4 >= 4.0.6)

Déplace le pointeur interne vers le résultat suivant

```
bool fbsql_next_result (resource result_id)
```

Lorsque vous envoyez plus d'une commande SQL au serveur, ou que vous exécutez une procédure stockée avec de multiple résultats, cela va conduire le serveur à retourner plusieurs jeu de lignes. **fbsql_next_result()** va vérifier l'existence de plusieurs résultats disponibles sur le serveur. Si un autre jeu de résultat existe, **fbsql_next_result()** va détruire de résultat précédent, et préparer la lecture dans les nouvelles lignes.

fbsql_next_result() retourne `TRUE` si un autre résultat est disponbile, ou `FALSE` sinon.

Exemple 1. Exemple avec fbsql_next_result()

```
<?php
  $link = fbsql_connect("localhost", "_SYSTEM", "secret");
  fbsql_select_db("MyDB", $link);
  $SQL = "Select * from table1; select * from table2;";
  $rs = fbsql_query($SQL, $link);
  do {
      while ($row = fbsql_fetch_row($rs)) {}
  } while (fbsql_next_result($rs));
  fbsql_free_result($rs);
  fbsql_close($link);
?>
```

fbsql_num_fields (PHP 4 >= 4.0.6)

Lit le nombre de champs dans un résultat

```
int fbsql_num_fields (resource result)
```

fbsql_num_fields() retourne le nombre de champs dans le résultat *result*.

Voir aussi **fbsql_db_query()**, **fbsql_query()**, **fbsql_fetch_field()** et **fbsql_num_rows()**.

fbsql_num_rows (PHP 4 >= 4.0.6)

Lit le nombre de lignes dans un résultat

```
int fbsql_num_rows (resource result)
```

fbsql_num_rows() retourne le nombre de lignes dans le résultat *result*. Cette fonction n'est valable qu'avec les commandes SELECT. Pour connaître le nombre de lignes dans une requête INSERT, UPDATE ou DELETE, utilisez **fbsql_affected_rows()**.

Exemple 1. Exemple fbsql_num_rows()

```
<?php
  $link = fbsql_connect("localhost", "username", "password");
  fbsql_select_db("database", $link);
  $result = fbsql_query("SELECT * FROM table1;", $link);
  $num_rows = fbsql_num_rows($result);
  echo "$num_rows Rows\n";
?>
```

Voir aussi **fbsql_affected_rows()**, **fbsql_connect()**, **fbsql_select_db()** et **fbsql_query()**.

fbsql_pconnect (PHP 4 >= 4.0.6)

Ouvre une connexion persistante à un serveur FrontBase

```
resource fbsql_pconnect (string [hostname] [, string username [, string password]])
```

fbsql_pconnect() retourne une ressource représentant la connection au serveur FrontBase en cas de succès, ou bien FALSE en cas d'erreur.

fbsql_pconnect() établit une connexion persistante à un serveur FrontBase. En cas d'omission, les valeurs suivantes sont utilisées par défaut : *host* ='localhost', *username* = nom de l'utilisateur qui possède le processus, et *password* = pas de mot de passe.

Pour choisir le port d'accès au serveur FrontBase, voyez **fbsql_select_db()**.

fbsql_pconnect() se comporte comme **fbsql_connect()** avec deux différences majeures.

Premièrement, lors de la connexion, **fbsql_pconnect()** essaie de trouver une connexion permanante déjà ouverte sur cet hote, avec le même nom d'utilisateur et de mot de passe. Si une telle connexion est trouvée, son identifiant est retourné, sans ouvrir de nouvelle connexion.

Deuxièmement, la connexion au serveur MySQL ne sera pas terminée avec la fin du script. Au lieu de cela, le lien sera conservé pour un prochain accès (**fbsql_close()** ne terminera pas une connexion persistante établie par **fbsql_pconnect()**).

C'est pourquoi ce type de connexion est dite 'persistante'.

fbsql_query (PHP 4 >= 4.0.6)

Exécute une requête sur un serveur FrontBase

```
resource fbsql_query (string query [, resource link_identifier])
```

fbsql_query() envoie la requête *query* à la base de données courante, sur le serveur représenté par sa connexion *link_identifier*. Si *link_identifier* est omis, la dernière connexion ouverte est utilisée. Si aucune connexion n'a été ouverte, **fbsql_query()** essaie d'établir une connexion en appelant la fonction **fbsql_connect()** sans aucun argument.

> **Note :** La requête doit être terminée par un point-virgule!

fbsql_query() retourne une ressource en cas de succès, ou FALSE, en cas d'échec.

La requête suivante est invalide, et **fbsql_query()** échouera puis retournera FALSE:

Exemple 1. Exemple avec fbsql_query()(1)

```
<?php
  $result = fbsql_query("SELECT * WHERE 1=1;")
    or die("Requête invalide");
?>
```

La requête suivante est invalide si my_col n'est pas une colonne dans la table my_tbl : **fbsql_query()** échouera puis retournera FALSE :

Exemple 2. Exemple avec fbsql_query()(2)

```
<?php
  $result = fbsql_query("SELECT my_col FROM my_tbl")
    or die ("Invalid query");
?>
```

fbsql_query() échouera si vous n'avez pas les droits d'accès sur l'une des bases de données utilisée dans la requête.

Lorsque la requête réussit, vous pouvez utilisez **fbsql_num_rows()** pour savoir combien de lignes ont été retournée par une requête SELECT, ou bien **fbsql_affected_rows()** pour les autres requêtes (DELETE, INSERT, REPLACE et UPDATE).

Pour les requêtes SELECT, **fbsql_query()** retourne une ressource de résultat, que vous pouvez passer à **fbsql_result()**. Lors vous avez fini de lire le résultat, vous pouvez libérer les ressources utilisées en appelant **fbsql_free_result()**. Cependant, la mémoire sera automatiquement libérée à la fin du script.

Voir aussi **fbsql_affected_rows()**, **fbsql_db_query()**, **fbsql_free_result()**, **fbsql_result()**, **fbsql_select_db()** et **fbsql_connect()**.

fbsql_result (PHP 4 >= 4.0.6)

Lit des données dans un résultat

```
mixed fbsql_result (resource result, int row, mixed [field])
```

fbsql_result() lit le contenu du champs *field*, dans la ligne *row*, du résultat *result*. L'argument *field* peut être l'offset du champs, ou bien son nom, ou bien le nom de sa table plus point plus son nom. Si la colonne a été aliasée, utilisez de préférence l'alias.

Lorsque vous travaillez sur de grands résultats, il est vivement recommandé d'utiliser les fonctions qui lisent toute une ligne d'un coup, plutôt que **fbsql_result()** qui travaille ligne par ligne. Elles sont beaucoup plus rapides. Notez aussi que les offset numériques sont plus rapides que les offset nominaux.

L'utilisation de **fbsql_result()** ne doivent pas être mélangé avec d'autres fonctions qui utilisent aussi le résultat *`result`*.

Alternative vivement recommandées : **fbsql_fetch_row()**, **fbsql_fetch_array()** et **fbsql_fetch_object()**.

fbsql_select_db (PHP 4 >= 4.0.6)

Sélectionne une base de données FrontBase

```
resource fbsql_select_db (string database_name, resource [link_identifier])
```

fbsql_select_db() retourne `TRUE` en cas de succès et `FALSE` en cas d'erreur.

fbsql_select_db() remplace la base de données active courante par *`database_name`*, sur la connexion ouverte et représentée par *`link_identifier`*. Si *`link_identifier`* est omis, la dernière connexion ouverte sera utilisée. Si aucune connexion n'a été ouverte, **fbsql_select_db()** essaiera de se connecter en appelant **fbsql_connect()** sans argument.

Le client contacte FBExec pour connaître le numéro de port à utiliser pour la connexion à la base de données. Si le nom de la base est un numéro, le système l'utilisera comme numéro de port, et ne le demandera pas à FBExec. Le serveur Frontbase peut être démarré avec la commande : `FRontBase -FBExec=No -port=<port number> <database name>`.

Tous les prochains appel à **fbsql_query()** se feront dans la base *`database_name`*.

Voir aussi **fbsql_connect()**, **fbsql_pconnect()** et **fbsql_query()**.

fbsql_tablename (unknown)

Lit le nom de la table d'un champs

```
string fbsql_tablename (resource result, int i)
```

fbsql_tablename() retourne le nom de la table d'origine du champs *`i`*, dans le résultat *`result`*. La fonction **fbsql_num_rows()** peut être utilisée pour connaître le nombre de tables dans un résultat.

Exemple 1. Exemple avec fbsql_tablename()

```
<?php
  fbsql_connect("localhost:3306");
  $result = fbsql_list_tables("wisconsin");
  $i = 0;
  while ($i < fbsql_num_rows($result)) {
    $tb_names[$i] = fbsql_tablename($result, $i);
    echo $tb_names[$i] . "<br>";
    $i++;
  }
?>
```

fbsql_warnings (PHP 4 >= 4.0.6)

Active ou désactive les alertes FrontBase

```
boolean fbsql_warnings (boolean [OnOff])
```

fbsql_warnings() retourne `TRUE` si les alertes sont actives et `FALSE` en cas d'erreur.

fbsql_warnings() active ou désactive les alertes FrontBase suivant que *`OnOff`* est à `TRUE` ou `FALSE`.

XXV. FilePro

Ces fonctions permettent de lire des données enregistrées dans des bases non modifiables, sur des serveurs filePro.

filePro est une marque de fP Technologies, Inc. Vous pouvez avoir plus de détails sur filePro à http://www.fptech.com/.

filepro (PHP 3, PHP 4 >= 4.0b1)

Lit et vérifie un fichier.

```
bool filepro (string directory)
```

filepro() lit et vérifie un fichier, puis enregistre le nombre de champs et de lignes.

Aucun verrouillage n'est pratiqué : il vaut alors mieux ne pas modifier la base filePro lorsqu'elle est ouverte par PHP.

filepro_fieldname (PHP 3, PHP 4 >= 4.0b1)

Retourne le nom d'un champs.

```
string filepro_fieldname (int field_number)
```

filepro_fieldname() retourne le nom du champs d'index *`field_number`*.

filepro_fieldtype (PHP 3, PHP 4 >= 4.0b1)

Retourne le type d'un champs.

```
string filepro_fieldtype (int field_number)
```

Retourne le type du champs d'index *`field_number`*.

filepro_fieldwidth (PHP 3, PHP 4 >= 4.0b1)

Retourne la taille d'un champs.

```
int filepro_fieldwidth (int field_number)
```

Retourne la taille du champs d'index *`field_number`*.

filepro_retrieve (PHP 3, PHP 4 >= 4.0b1)

Retourne la valeur d'un champs.

```
string filepro_retrieve (int row_number, int field_number)
```

filepro_retrieve() retourne la valeur du champs d'index *`row_number`*, et à la ligne *`field_number`*.

filepro_fieldcount (PHP 3, PHP 4 >= 4.0b1)

Retourne le nombre de champs dans une base filePro.

```
int filepro_fieldcount (void)
```

filepro_fieldcount() retourne le nombre de champs (ou colonnes) d’une base filePro.

Voir aussi **filepro()**.

filepro_rowcount (PHP 3, PHP 4 >= 4.0b1)

Retourne le nombre de champs dans une base filePro.

```
int filepro_rowcount (void)
```

filepro_rowcount() retourne le nombre de lignes dans une base filePro.

Voir aussi **filepro()**.

XXVI. Système de fichiers

basename (PHP 3, PHP 4 >= 4.0b1)

Sépare le nom du fichier et le nom du dossier.

```
string basename (string path)
```

basename() prend en paramètre le chemin complet d'un fichier et en extrait le nom du fichier.

Sous Windows, les caractères (/) et antislash (\) sont utilisés comme séparateurs de dossier. Sous les autres OS, seul le caractère slash (/) est utilisé.

Exemple 1. Exemple avec basename()

```
<?php
  $path = "/home/httpd/html/index.php3";
  $file = basename($path);
  // $file est affecté avec "index.php3"
?>
```

Voir aussi **dirname()**.

chgrp (PHP 3, PHP 4 >= 4.0b1)

Change le groupe possesseur du fichier.

```
int chgrp (string filename, mixed group)
```

chgrp() essaie de changer le groupe propriétaire du fichier. Seul le super-utilisateur (root) peut changer le groupe propriétaire d'un fichier arbitrairement. Les utilisateurs classiques ne peuvent changer le groupe propriétaire d'un fichier que si l'utilisateur propriétaire du fichier est membre du groupe.

chgrp() renvoie TRUE en cas de succès, sinon renvoie FALSE.

Voir aussi **chown()** et **chmod()**.

> **Note : chgrp()** ne fonctionne pas sous Windows.

chmod (PHP 3, PHP 4 >= 4.0b1)

Change le mode du fichier.

```
int chmod (string filename, int mode)
```

chmod() remplace le mode du fichier *filename* par le mode *mode*.

Il est à noter que le mode *mode* est considéré comme un nombre en notation octale. Afin de vous en assurer, vous pouvez préfixer cette valeur par un zéro (*mode*):

```
<?php
  chmod( "/somedir/somefile", 755 );
// notation décimale; probablement FALSE
  chmod( "/somedir/somefile", 0755 );
// notation octale; valeur du mode correcte
?>
```

chmod() renvoie TRUE en cas de succès, FALSE sinon.

Voir aussi **chown()** et **chgrp()**.

> **Note : chmod()** ne fonctionne pas sous Windows.

chown (PHP 3, PHP 4 >= 4.0b1)

Change le groupe propriétaire du fichier.

```
int chown (string filename, mixed user)
```

chown() change le groupe propriétaire du fichier. Seul le super-utilisateur (root) peut changer le propriétaire arbitrairement d'un fichier.

chown() renvoie TRUE en cas de succès, "FALSE" sinon.

> **Note :** Sous Windows, **chown()** ne fait rien et retourne TRUE.

Voir aussi **chown()** et **chmod()**.

> **Note : chown()** est inopérante sous Windows.

clearstatcache (PHP 3, PHP 4 >= 4.0b1)

Efface le cache de **stat()**

```
void clearstatcache (void)
```

L'appel à la fonction stat ou lstat est relativement coÛteux en terme de temps d'exécution. Pour cela, le résultat du dernier appel à l'une des fonctions de statut, (voir la liste ci-dessous), est sauvegardé pour ré-utilisation ultérieure. Si vous voulez forcer la vérification du statut d'un fichier, dans le cas où le fichier aurait pu être modifié ou aurait disparu, vous devez utiliser la fonction **clearstatcache()** afin d'effacer de la mémoire les résultats du dernier appel à la fonction.

La valeur du cache n'est valable que pour la durée d'une requête.

Les fonctions affectées sont : **stat()**, **lstat()**, **file_exists()**, **is_writable()**, **is_readable()**, **is_executable()**, **is_file()**, **is_dir()**, **is_link()**, **filectime()**, **fileatime()**, **filemtime()**, **fileinode()**, **filegroup()**, **fileowner()**, **filesize()**, **filetype()**, et **fileperms()**.

copy (PHP 3, PHP 4 >= 4.0b1)

Copie un fichier.

```
int copy (string source, string dest)
```

copy() fait une copie du fichier. Elle renvoie TRUE en cas de succès, FALSE sinon.

Exemple 1. Exemple avec copy()

```
if ( !copy($file, $file.’.bak’) ) {
    print("La copie du fichier $file n’a pas réussi...<br>\n");
}
```

Voir aussi **rename()**.

delete (unknown)

Effacer

```
void delete (string file)
```

Ceci est une fausse entrée du manuel pour ceux qui recherchent en fait la fonction **unlink()** ou **unset()**.

Voir aussi **unlink()** pour effacer des fichiers et **unset()** pour effacer des variables.

dirname (PHP 3, PHP 4 >= 4.0b1)

Renvoie le nom du dossier.

```
string dirname (string path)
```

Si *path* contient le chemin d’un fichier ou dossier, **dirname()** retournera le nom du dossier qui le contient.

Sous Windows, les slash (/) et anti-slash (\) sont utilisés comme séparateurs de dossier. Dans les autres environnements, seul le slash (/) est utilisé.

Exemple 1. Exemple avec dirname()

```
<?php
  $path = "/etc/passwd";
  $file = dirname($path); // $file contient "/etc"
?>
```

Voir aussi **basename()**.

diskfreespace (PHP 3>= 3.0.7, PHP 4 >= 4.0b4)

Renvoie l’espace disque disponible dans le répertoire.

```
float diskfreespace (string directory)
```

diskfreespace() retournera le nombre d’octets disponibles sur le disque correspondant contenant le dossier *directory*.

Exemple 1. Exemple avec diskfreespace()

```
<?php
  $df = diskfreespace("/");
  // $df contient le nombre d'octets libres sur "/"
?>
```

disk_total_space (PHP 4 CVS only)

Retourne la taille d'un dossier

```
float disk_total_space (string directory)
```

disk_total_space() lit récursivement toutes les tailles du dossier *directory* et retourne la somme. *directory* peut être aussi une partition de disque.

Exemple 1. Exemple avec disk_total_space()

```
<?php
  $df = disk_total_space("/"); // $df contient le nombre d'octets libres
                               // dans le dossier "/"
?>
```

fclose (PHP 3, PHP 4 >= 4.0b1)

Ferme un fichier.

```
bool fclose (resource fp)
```

fclose() ferme le fichier *fp*.

fclose() retourne TRUE en cas de succès, et FALSE en cas d'échec.

Le pointeur de fichier doit être valide, et avoir été correctement ouvert par **fopen()** ou **fsockopen()**.

feof (PHP 3, PHP 4 >= 4.0b1)

Teste la fin du fichier.

```
int feof (resource fp)
```

feof() retourne TRUE si le pointeur *fp* est à la fin du fichier, ou si une erreur survient, sinon, retourne FALSE.

Le pointeur de fichier doit être valide, et avoir été correctement ouvert par **fopen()**, **popen()**, ou **fsockopen()**.

fflush (PHP 4 >= 4.0.1)

Envoi tout le contenu généré dans un fichier

```
int fflush (resource fp)
```

fflush() force l'écriture de toutes les données bufferisées dans le fichier désigné par *fp*. **fflush()** retourne TRUE en cas de succès, et FALSE sinon.

fp est un pointeur de fichier ouvert avec **fopen()**, **popen()**, ou **fsockopen()**.

fgetc (PHP 3, PHP 4 >= 4.0b1)

Renvoie le caractère que pointe le pointeur du fichier.

```
string fgetc (resource fp)
```

fgetc() retourne une chaîne contenant un seul caractère, lu depuis le fichier pointé par *fp*. **fgetc()** retourne FALSE à la fin du fichier (tout comme **feof()**).

Le pointeur de fichier doit être valide, et avoir été correctement ouvert par **fopen()**, **popen()**, ou **fsockopen()**.

Voir aussi **fread()**, **fopen()**, **popen()**, **fsockopen()** et **fgets()**.

fgetcsv (PHP 3>= 3.0.8, PHP 4 >= 4.0b1)

Renvoie la ligne courante et cherche les champs CSV

```
array fgetcsv (resource fp, int length, string [delimiter])
```

Identique à **fgets()** mais **fgetcsv()** analyse la ligne qu'il lit et recherche les champs CSV, qu'il va retourner dans un tableau les contenant. Le délimiteur de champs *delimiter* est la virgule, à moins que vous ne fournissiez un troisième argument.

fp doit être un pointeur valide, et avoir été correctement ouvert par **fopen()**, **popen()**, ou **fsockopen()**.

length doit être plus grand que la plus grande ligne trouvée dans un fichier CSV (en comptant les caractères de fin de ligne).

fgetcsv() retourne FALSE en cas d'erreur, ou en cas de fin du fichier.

Note : une ligne vide dans un fichier CSV sera retournée dans le tableau comme une chaîne vide, et ne sera pas traitée comme une erreur.

Exemple 1. Exemple avec fgetcsv()

```
<?php
  $row = 1;
  $fp = fopen ("test.csv","r");
  while ($data = fgetcsv ($fp, 1000, ",")) {
    $num = count ($data);
    print "<p> $num champs dans la ligne $row: <br>";
    $row++;
    for ($c=0; $c<$num; $c++) {
        print $data[$c] . "<br>";
    }
  }
  fclose ($fp);
?>
```

fgets (PHP 3, PHP 4 >= 4.0b1)

Renvoie la ligne courante sur laquelle se trouve le pointeur du fichier.

```
string fgets (int fp, int length)
```

fgets() retourne la chaîne lue jusqu'à la longueur *length* - 1 octet, ou bien la fin du fichier, ou encore un retour chariot (le premier des trois qui sera rencontré).

Si une erreur survient, **fgets()** retourne FALSE.

Erreur courante :

Les programmeurs habitués à la programmation 'C' noteront que **fgets()** ne se comporte pas comme son équivalent C lors de la rencontre de la fin du fichier.

fp doit être valide, et avoir été correctement ouvert par **fopen()**, **popen()**, ou **fsockopen()**.

Un exemple simple :

Exemple 1. Lecture d'un fichier ligne par ligne

```
<?php
  $fd = fopen ("/tmp/inputfile.txt", "r");
  while (!feof($fd)) {
    $buffer = fgets($fd, 4096);
    echo $buffer;
  }
  fclose ($fd);
?>
```

Voir aussi **fread()**, **fopen()**, **popen()**, **fgetc()**, **fsockopen()** et **socket_set_timeout()**.

fgetss (PHP 3, PHP 4 >= 4.0b1)

Renvoie la ligne courante sur laquelle se trouve le pointeur du fichier et élimine les balises HTML

```
string fgetss (int fp, int length, string [allowable_tags])
```

Identique à **fgets()**, mais **fgetss()** supprime toutes les balises HTML et PHP qu'il trouve dans le texte lu.

Vous pouvez aussi préciser les balises qui seront ignorées dans le troisième paramètre, optionnel.

> **Note :** *allowable_tags* a été ajouté dans PHP 3.0.13, PHP 4B3.

Voir aussi **fgets()**, **fopen()**, **fsockopen()**, **popen()** et **strip_tags()**.

file (PHP 3, PHP 4 >= 4.0b1)

Lit le fichier et renvoie le résultat dans un tableau.

```
array file (string filename [, int use_include_path])
```

Identique à **readfile()**, hormis le fait que **file()** retourne le fichier dans un tableau. Chaque élément du tableau correspond à une ligne du fichier, et les retour-chariots sont placés en fin de ligne.

Vous pouvez utiliser l'option *use_include_path* : en la mettant à "1", vous rechercherez aussi dans le dossier include_path.

```
<?php
// Lire une page web dans un tableau, et l'afficher
  $fcontents = file( 'http://www.php.net' );
  while ( list( $numero_ligne, $ligne ) = each( $fcontents ) ) {
   echo "<B>Ligne $numero_ligne:</B> ".htmlspecialchars( $ligne ) . "<br>\n";
  }
// lire une page web dans une chaîne
  $fcontents = join( '', file( 'http://www.php.net' ) );
?>
```

Voir aussi **readfile()**, **fopen()**, **fsockopen()** et **popen()**.

file_exists (PHP 3, PHP 4 >= 4.0b1)

Vérifie si un fichier existe.

```
int file_exists (string filename)
```

file_exists() retourne TRUE si le fichier *filename* existe, et FALSE sinon.

file_exists() ne fonctionne pas sur les fichiers distants. Les fichiers doivent être accessibles par le système de fichier du serveur.

Le résultat de **file_exists()** est mis en cache. Reportez-vous à **clearstatcache()** pour plus de détails.

fileatime (PHP 3, PHP 4 >= 4.0b1)

Renvoie la date à laquelle le fichier a été accédé pour la dernière fois.

```
int fileatime (string filename)
```

fileatime() renvoie la date à laquelle le fichier a été accédé pour la dernière fois, ou FALSE en cas d'erreur.

fileatime() ne fonctionne pas sur les fichiers distants. Les fichiers doivent être accessibles par le système de fichier du serveur.

Le résultat de **fileatime()** est mis en cache. Reportez-vous à **clearstatcache()** pour plus de détails.

filectime (PHP 3, PHP 4 >= 4.0b1)

Renvoie l'heure à laquelle l'inode a été accédé pour la dernière fois.

```
int filectime (string filename)
```

filectime() renvoie l'heure à laquelle l'inode *filename* a été accédé pour la dernière fois, ou FALSE en cas d'erreur.

filectime() ne fonctionne pas sur les fichiers distants. Les fichiers doivent être accessibles par le système de fichier du serveur.

Note: Sur la plupart des serveurs UNIX, un fichier est considéré comme modifié si les données de son inode sont modifiées. C'est-à-dire lorsque les permissions (utilisateur, groupe ou autre) ont été modifiées. Voyez aussi **filemtime()** (que vous pourrez utiliser lorsque vous créerez des indications telles que "Dernière modification : " sur les pages web) et **fileatime()**.

Notez aussi que sur certains systèmes UNIX, le ctime d'un fichier texte est considéré comme sa date de création. Cela est faux! Il n'y a pas de date de création de fichier sous la plupart des systèmes UNIX.

Le résultat de **filectime()** est mis en cache. Reportez-vous à **clearstatcache()** pour plus de détails.

filegroup (PHP 3, PHP 4 >= 4.0b1)

Lire le nom du groupe

```
int filegroup (string filename)
```

filegroup() renvoie le groupe qui possède le fichier *filename*, ou FALSE en cas d'erreur. L'identifiant de groupe est retourné au format numérique, utilisez **posix_getgrgid()** pour retrouver le nom du groupe.

filegroup() ne fonctionne pas sur les fichiers distants. Les fichiers doivent être accessibles par le système de fichier du serveur.

Le résultat de **filegroup()** est mis en cache. Reportez-vous à **clearstatcache()** pour plus de détails.

> **Note : filegroup()** est inopérante sous Windows.

fileinode (PHP 3, PHP 4 >= 4.0b1)

Renvoie le numéro d'inode du fichier.

```
int fileinode (string filename)
```

fileinode() renvoie le numéro d'inode du fichier *filename*, ou FALSE en cas d'erreur.

Le résultat de **fileinode()** est mis en cache. Reportez-vous à **clearstatcache()** pour plus de détails.

fileinode() ne fonctionne pas sur les fichiers distants. Les fichiers doivent être accessibles par le système de fichier du serveur.

> **Note : fileinode()** est inopérante sous Windows.

filemtime (PHP 3, PHP 4 >= 4.0b1)

Renvoie la date de dernière modification du fichier.

```
int filemtime (string filename)
```

filemtime() renvoie la date de dernière modification du fichier *filename*, ou FALSE en cas d'erreur.

Le résultat de **filemtime()** est mis en cache. Reportez-vous à **clearstatcache()** pour plus de détails.

filemtime() ne fonctionne pas sur les fichiers distants. Les fichiers doivent être accessibles par le système de fichier du serveur.

filemtime() retourne l'heure d'écriture des blocs données d'un fichier. Utilisez **date()** sur ce résultat pour obtenir une date de modification humainement lisible.

fileowner (PHP 3, PHP 4 >= 4.0b1)

Renvoie le nom du propriétaire du fichier.

```
int fileowner (string filename)
```

fileowner() renvoie le nom du possesseur du fichier *filename*, ou FALSE en cas d'erreur. L'identification du possesseur de fichier est numérique : il faut utiliser **posix_getpwuid()** pour retrouver le nom d'utilisateur.

fileowner() ne fonctionne pas sur les fichiers distants. Les fichiers doivent être accessibles par le système de fichier du serveur.

Le résultat de **fileowner()** est mis en cache. Reportez-vous à **clearstatcache()** pour plus de détails.

> **Note : fileowner()** est inopérante sous Windows.

fileperms (PHP 3, PHP 4 >= 4.0b1)

Renvoie les permissions affectées au fichier.

```
int fileperms (string filename)
```

fileperms() renvoie les permissions affectées au fichier *filename*, ou FALSE en cas d'erreur.

fileperms() ne fonctionne pas sur les fichiers distants. Les fichiers doivent être accessibles par le système de fichier du serveur.

Le résultat de **fileperms()** est mis en cache. Reportez-vous à **clearstatcache()** pour plus de détails.

filesize (PHP 3, PHP 4 >= 4.0b1)

Renvoie la taille du fichier.

```
int filesize (string filename)
```

filesize() renvoie la taille du fichier *filename*, ou FALSE en cas d'erreur.

filesize() ne fonctionne pas sur les fichiers distants. Les fichiers doivent être accessibles par le système de fichier du serveur.

Le résultat de **filesize()** est mis en cache. Reportez-vous à **clearstatcache()** pour plus de détails.

filetype (PHP 3, PHP 4 >= 4.0b1)

Retourne le type de fichier

```
string filetype (string filename)
```

filetype() renvoie le type du fichier *filename*. Les réponses possibles sont : fifo, char, dir, block, link, file, et unknown.

filetype() retourne FALSE en cas d'erreur.

filetype() ne fonctionne pas sur les fichiers distants. Les fichiers doivent être accessibles par le système de fichier du serveur.

Le résultat de **filetype()** est mis en cache. Reportez-vous à **clearstatcache()** pour plus de détails.

flock (PHP 3>= 3.0.7, PHP 4 >= 4.0b1)

Verrouille le fichier.

```
boolean flock (int fp, int operation)
```

PHP dispose d'un système complet de verrouillage de fichiers. Tous les programmes qui accèdent au fichier doivent utiliser la même méthode de verrouillage pour qu'il soit efficace.

flock() agit sur le fichier *fp* qui doit avoir été ouvert au préalable. *operation* est une des valeurs suivantes :

- Acquisition d'un verrou : *operation* = 1.
- Acquisition d'un verrou exclusif (écriture), *operation* = 2.
- Libération d'un verrou (partagé ou exclusif), *operation* = 3.
- Si vous voulez que **flock()** ne se bloque pas durant le verrouillage, ajoutez 4 à *operation*.

flock() permet de réaliser un système simple de verrous écriture / lecture, qui peut être utilisé sur n'importe quelle plate-forme (Unix et Windows compris).

flock() retourne TRUE en cas de succès, et FALSE sinon. (le verrou n'a pas pu être obtenu).

> **Avertissement**
>
> Sur la plupart des OS, **flock()** est implémenté au niveau processus. Lors de l'utilisation des API d'un serveur multi-thread, comme ISAPI, vous ne pouvez pas vous fier à **flock()** pour protéger vos fichiers contre des accès concurrents du même serveur.

fopen (PHP 3, PHP 4 >= 4.0b1)

Ouverture d'un fichier ou d'une URL.

```
int fopen (string filename, string mode, int [use_include_path])
```

Si *filename* commence par "http://" (insensible à la casse), une connexion HTTP 1.x est ouverte avec le serveur spécifié, et un pointeur sur la réponse fournie est retourné. Une en-tête 'Host:' est envoyé avec la requête, afin de gérer les virtual hosts basés sur les noms.

Notez que le pointeur de fichier retourné représente le corps de la réponse, et qu'il n'est pas possible d'accéder aux en-têtes HTTP avec cette fonction.

Les versions antérieures à PHP 4.0.6, ne gère pas les redirections automatiques, ce qui oblige à ajouter les slash finaux "/" pour indiquer un dossier.

Si *filename* commence par "ftp://" (insensible à la casse), une connexion FTP est ouverte avec le serveur spécifié, et un pointeur sur la réponse fournie est retourné. Si le serveur ne supporte par le mode FTP passif, **fopen()** échouera. Vous pouvez ouvrir des fichiers en lecture seulement, ou en écriture seulement (le full duplex n'est pas supporté).

Si *filename* commence par "php://stdin", "php://stdout", ou "php://stderr", le flot correspondant sera ouvert. (Cela a été introduit en PHP 3.0.13; dans les anciennes versions, les fichiers "/dev/stdin" ou "/dev/fd/0" devaient être utilisés pour accéder à ces flots).

Si *filename* commence par n'importe quoi d'autre, PHP tentera de lire ce fichier dans le système local, et un pointeur sur le fichier ouvert sera retourné.

Si l'ouverture échoue, **fopen()** retourne FALSE.

mode peut prendre les valeurs suivantes :

- 'r' - Ouvre en lecture seule, et place le pointeur de fichier au début du fichier.

- 'r+' - Ouvre en lecture et écriture, et place le pointeur de fichier au début du fichier.
- 'w' - Ouvre en écriture seule; place le pointeur de fichier au début du fichier et réduit la taille du fichier à 0. Si le fichier n'existe pas, on tente de le créer.
- 'w+' - Ouvre en lecture et écriture; place le pointeur de fichier au début du fichier et réduit la taille du fichier à 0. Si le fichier n'existe pas, on tente de le créer.
- 'a' - Ouvre en écriture seule; place le pointeur de fichier à la fin du fichier file. Si le fichier n'existe pas, on tente de le créer.
- 'a+' - Ouvre en lecture et écriture; place le pointeur de fichier à la fin du fichier. Si le fichier n'existe pas, on tente de le créer.

De plus, *mode* peut contenir la lettre 'b'. Cette option n'est utile que sur les systèmes qui font la différence entre les fichiers binaires et les fichiers textes (en bref, c'est une fonctionnalité Windows, totalement inutile sous Unix). S'il n'est pas nécessaire, il sera ignoré.

Vous pouvez utiliser le troisième paramètre optionnel pour explorer le dossier include_path, en le mettant à 1.

Exemple 1. Exemple avec fopen()

```
<?php
  $fp = fopen("/home/rasmus/file.txt", "r");
  $fp = fopen("http://www.php.net/", "r");
  $fp = fopen("ftp://user:password@example.com/", "w");
?>
```

Si vous rencontrez des problèmes en lecture ou écriture de fichier et que vous utilisez PHP en version module de serveur, n'oubliez pas que les fichiers auxquels vous accédez ne sont pas nécessairement accessibles au processus serveur.

Sous Windows, assurez-vous de bien échapper les anti-slash utilisés dans le chemin du fichier, ou bien utilisez des slash.

```
<?php
  $fp = fopen("c:\\data\\info.txt", "r");
?>
```

Voir aussi **fclose()**, **fsockopen()**, **socket_set_timeout()** et **popen()**.

fpassthru (PHP 3, PHP 4 >= 4.0b1)

Affiche la partie du fichier située après le pointeur du fichier.

```
int fpassthru (int fp)
```

fpassthru() lit tout le reste d'un fichier jusqu'à la fin, et dirige le résultat vers la sortie standard.

Si une erreur survient, **fpassthru()** retourne FALSE.

Le pointeur de fichier doit être valide, et doit avoir été correctement ouvert par **fopen()**, **popen()**, ou **fsockopen()**. Après lecture, **fpassthru()** va fermer le fichier (le pointeur *fp* sera alors invalide).

Si vous voulez simplement afficher le contenu d'un fichier, il suffit d'utiliser **readfile()**, ce qui épargnera l'appel à **fopen()**.

Voir aussi **readfile()**, **fopen()**, **popen()** et **fsockopen()**.

fputs (PHP 3, PHP 4 >= 4.0b1)

Ecrit dans un fichier.

```
int fputs (int fp, string str, int [length])
```

fputs() est un alias de **fwrite()**, et lui est identique en tous points. Notez que *length* est un paramètre optionnel, et s'il n'est pas spécifié, toute la chaîne est écrite.

fread (PHP 3, PHP 4 >= 4.0b1)

Lecture du fichier en mode binaire.

```
string fread (int fp, int length)
```

fread() lit jusqu'à *length* octets dans le fichier referencé par *fp*. La lecture s'arrête lorsque *length* octets ont été lus, ou que l'on a atteint la fin du fichier, ou qu'une erreur survient (le premier des trois).

```
<?php
// Lit un fichier, et le place dans une chaîne
  $filename = "/usr/local/quelquechose.txt";
  $fd = fopen($filename, "r");
  $contents = fread($fd, filesize ($filename));
  fclose($fd);
?>
```

Note : Sur les systèmes qui différencient les fichiers textes et binaires (i.e. Windows) le fichier doit être ouvert avec la lettre 'b' ajoutée au paramètre de mode de la fonction **fopen()**.

Note :

```
<?php
  $filename = "c:\\fichiers\\uneimage.gif";
  $fd = fopen($filename, "rb");
  $contents = fread($fd, filesize ($filename));
  fclose($fd);
?>
```

Voir aussi **fwrite()**, **fopen()**, **fsockopen()**, **popen()**, **fgets()**, **fgetss()**, **file()** et **fpassthru()**.

fscanf (PHP 4 >= 4.0.1)

Analyse un fichier en fonction d'un format

```
mixed fscanf (int handle, string format [, string var1])
```

fscanf() est similaire à **sscanf()**, mais elle prend comme entrée un fichier, associé à *handle* et l'interprète en fonction du format *format*. Si seulement deux paramètres sont passés à la fonction, les valeurs analysées seront retournées sous

forme de tableau. Si des arguments optionnels sont passés, la fonction retournera le nombre de valeurs assignées. Les options doivent être passées par référence.

Exemple 1. Exemple fscanf()

```
<?php
  $fp = fopen ("users.txt","r");
  while ($userinfo = fscanf ($fp, "%s\t%s\t%s\n")) {
    list ($name, $profession, $countrycode) = $userinfo;
    //... traitement des données
  }
  fclose($fp);
?>
```

Exemple 2. users.txt

```
janus  argonaute        gr
rodin  sculpteur        fr
sam    oncle            us
leonard   inventeur     it
```

Voir aussi **fread()**, **fgets()**, **fgetss()**, **sscanf()**, **printf()** et **sprintf()**.

fseek (PHP 3, PHP 4 >= 4.0b1)

Modifie le pointeur de fichier.

```
int fseek (int fp, int offset)
```

fseek() modifie le curseur de position dans le fichier *fp*. La nouvelle position mesurée en octets à partir du début du fichier, est obtenue en additionnant la distance *offset* à la position *whence*. Ce paramètre peut prendre les valeurs suivantes :
SEEK_SET - La position finale vaut *offset* octets.
SEEK_CUR - La position finale vaut la position courante ajoutée à *offset* octets.
SEEK_END - La position finale vaut la position courante par rapport à la fin du fichier, ajoutée de *offset*.

Si *whence* n'est pas spécifiée, il vaut par défaut SEEK_SET.

fseek() retourne TRUE en cas de succès, et sinon -1. Notez que positionner le pointeur au delà de la fin du fichier n'est pas une erreur.

fseek() ne peut pas être utilisé sur les pointeurs retournés par **fopen()** s'ils sont au format HTTP ou FTP.

Voir aussi **ftell()** et **rewind()**.

fstat (PHP 4 >= 4.0RC1)

Lit les informations sur un fichier à partir d'un pointeur de fichier

```
array fstat (int fp)
```

fstat() rassemble les informations sur le fichier dont on connaît le pointeur *fp*. **fstat()** est similaire à la fonction **stat()**, hormis le fait qu'elle utilise un pointeur de fichier, au lieu d'un nom de fichier.

fstat() retourne un tableau avec les éléments suivants :

- 1 : volume
- 2 : inode

- 3 : mode de protection du inode
- 4 : nombre de liens
- 5 : id de l'utilisateur propriétaire
- 6 : id du groupe propriétaire
- 7 : type du volume de l'inode *
- 8 : taille en octets
- 9 : date du dernier accès
- 10 : date de la dernière modification
- 11 : date du dernier changement
- 12 : taille de bloc du système pour les entrées/sorties(*)
- 13 : Nombre de blocs alloués

* - uniquement sur les systèmes qui supportent le type st_blksize. Les autres systèmes (i.e. Windows) retournent -1.

Les résultats de **fstat()** sont mis en cache. Reportez-vous à la fonction **clearstatcache()** pour plus de détails.

ftell (PHP 3, PHP 4 >= 4.0b1)

Renvoie la position du pointeur du fichier.

```
int ftell (int fp)
```

ftell() retourne la position courante du pointeur dans le fichier repéré par le pointeur *fp*, i.e., son offset.

Si une erreur survient, retourne FALSE.

Le pointeur de fichier doit être valide, et avoir été correctement ouvert par **fopen()** ou **popen()**.

Voir aussi **fopen()**, **popen()**, **fseek()** et **rewind()**.

ftruncate (PHP 4 >= 4.0RC1)

Tronque un fichier.

```
int ftruncate (int fp, int size)
```

ftruncate() prend le pointeur de fichier *fp* et le tronque à la taille de *size*.**ftruncate()** retourne TRUE en cas de succès, et FALSE sinon.

fwrite (PHP 3, PHP 4 >= 4.0b1)

Ecriture du fichier en mode binaire.

```
int fwrite (int fp, string string, int [length])
```

fwrite() écrit le contenu de la chaîne *string* dans le fichier pointé par *fp*. Si la longueur *length* est fournie, l'écriture s'arrêtera après *length* octets, ou à la fin de la chaîne (le premier des deux).

Notez que si *length* est fourni, alors l'option de configuration magic_quotes_runtime sera ignorée, et les slash seront conservés.

Note : De plus, *mode* peut contenir la lettre 'b'. Cette option n'est utile que sur les systèmes qui font la différence entre les fichiers binaires et les fichiers textes (en bref, c'est une fonctionnalité Windows, totalement inutile sous Unix). S'il n'est pas nécessaire, il sera ignoré.

Voir aussi **fread()**, **fopen()**, **fsockopen()**, **popen()** et **fputs()**.

set_file_buffer (PHP 3>= 3.0.8, PHP 4 >= 4.0.1)

Fixe la bufferisation de fichier

```
int set_file_buffer (int fp, int buffer)
```

L'écriture de fichier avec **fwrite()** utilise normalement un buffer de 8K. Cela signifie que si deux processus essaient d'écrire dans le même fichier, ils font une pause tous les 8ko pour laisser le temps à l'autre d'écrire à son tour. **set_file_buffer()** permet de modifier la taille du buffer de sortie pour le pointeur de fichier *fp* à *buffer* octets. Si *buffer* vaut 0, l'écriture se fera sans buffer. Cela force un processus à écrire toutes ses données dans un fichier avant que les autres puissent y accéder.

set_file_buffer() retourne 0 en cas de succès, ou EOF si la requête ne peut pas être honorée.

L'exemple suivant montre comment utiliser la fonction **set_file_buffer()** pour créer un fichier sans buffer.

Exemple 1. Exemple avec set_file_buffer()

```
<?php
  $fp=fopen($file, "w");
  if($fp){
    set_file_buffer($fp, 0);
    fputs($fp, $output);
    fclose($fp);
  }
?>
```

Voir aussi **fopen()** et **fwrite()**.

is_dir (PHP 3, PHP 4 >= 4.0b1)

Indique si le nom de fichier est un dossier.

```
boolean is_dir (string filename)
```

is_dir() retourne TRUE si *filename* existe et est un dossier.

Le résultat de **is_dir()** est mis en cache. Voir la fonction **clearstatcache()** pour plus de détails.

is_dir() ne fonctionne pas sur les fichiers distants. Les fichiers doivent être accessibles par le système de fichier du serveur.

Voir aussi **is_file()** et **is_link()**.

is_executable (PHP 3, PHP 4 >= 4.0b1)

Indique si le fichier est exécutable.

```
boolean is_executable (string filename)
```

is_executable() retourne TRUE si *filename* existe et est exécutable.

Le résultat de **is_executable()** est mis en cache. Voir la fonction **clearstatcache()** pour plus de détails.

is_executable() ne fonctionne pas sur les fichiers distants. Les fichiers doivent être accessibles par le système de fichier du serveur.

Voir aussi **is_file()** et **is_link()**.

is_file (PHP 3, PHP 4 >= 4.0b1)

Indique si le fichier est un véritable fichier.

```
boolean is_file (string filename)
```

is_file() retourne TRUE si *filename* existe et est un fichier (et pas un dossier).

Le résultat de **is_file()** est mis en cache. Voir la fonction **clearstatcache()** pour plus de détails.

is_file() ne fonctionne pas sur les fichiers distants. Les fichiers doivent être accessibles par le système de fichier du serveur.

Voir aussi **is_dir()** et **is_link()**.

is_link (PHP 3, PHP 4 >= 4.0b1)

Indique si le fichier est un lien symbolique.

```
boolean is_link (string filename)
```

is_link() retourne TRUE si *filename* existe et est un lien symbolique.

Le résultat de **is_link()** est mis en cache. Voir la fonction **clearstatcache()** pour plus de détails.

is_link() ne fonctionne pas sur les fichiers distants. Les fichiers doivent être accessibles par le système de fichier du serveur.

Voir aussi **is_dir()** et **is_file()**.

> **Note : is_link()** est inopérante sous Windows.

is_readable (PHP 3, PHP 4 >= 4.0b1)

Indique un fichier est autorisé en lecture

```
boolean is_readable (string filename)
```

is_readable() retourne TRUE si *filename* existe et est accessible en lecture.

N'oubliez pas que PHP accède aux fichiers avec les mêmes autorisations que l'utilisateur qui fait tourner le serveur web (souvent, c'est 'nobody', personne).

Le résultat de **is_readable()** est mis en cache. Voir la fonction **clearstatcache()** pour plus de détails.

is_readable() ne fonctionne pas sur les fichiers distants. Les fichiers doivent être accessibles par le système de fichier du serveur.

Voir aussi **is_writable()**.

is_writable (PHP 4)

Indique si un fichier est autorisé en écriture.

```
boolean is_writable (string filename)
```

is_writable() retourne TRUE si *filename* existe et est accessible en écriture.

N'oubliez pas que PHP accède aux fichiers avec les mêmes autorisations que l'utilisateur qui fait tourner le serveur web (souvent, c'est 'nobody', personne).

is_writable() ne fonctionne pas sur les fichiers distants. Les fichiers doivent être accessibles par le système de fichier du serveur.

Le résultat de **is_writable()** est mis en cache. Voir la fonction **clearstatcache()** pour plus de détails.

Voir aussi **is_readable()**.

is_writeable (PHP 3, PHP 4 >= 4.0b1)

Indique si un fichier est autorisé en écriture.

```
boolean is_writeable (string filename)
```

is_writeable() est un alias de **is_writable()**.

is_uploaded_file (PHP 3>= 3.0.17, PHP 4 >= 4.0.3)

Indique si le fichier a été téléchargé par HTTP POST

```
boolean is_uploaded_file (string filename)
```

is_uploaded_file() est disponible à partir des versions PHP 3.0.16 et 4.0.2.

is_uploaded_file() retourne TRUE si le fichier `filename` a été téléchargé par HTTP POST. Cela est très utile pour vous assurer qu'un utilisateur n'essaie pas d'accéder intentionnellement à un fichier auquel il n'a pas droit (comme `/etc/passwd`).

Ce type de vérification est spécialement important s'il est possible que les fichiers téléchargés révélent leur contenu à l'utilisateur, ou même aux utilisateurs du même système.

Voir aussi **move_uploaded_file()**, et la section Chargement de fichier pour un exemple simple.

link (PHP 3, PHP 4 >= 4.0b1)

Crée un lien.

```
int link (string target, string link)
```

link() crée un lien.

Voir aussi **symlink()** pour créer des liens symboliques et **readlink()** avec **linkinfo()**.

> **Note : link()** est inopérante sous Windows.

linkinfo (PHP 3, PHP 4 >= 4.0b1)

Renvoie les informations à propos d'un lien.

```
int linkinfo (string path)
```

linkinfo() renvoie les informations à propos d'un lien, c'est-à-dire le champs st_dev de la structure d'information UNIX (comme en langage C). **linkinfo()** sert à vérifier si un lien (repéré par *path*) existe (en utilisant la même méthode que la macro S_ISLNK de stat.h). **linkinfo()** retourne FALSE en cas d'erreur.

Voir aussi **symlink()**, **link()** et **readlink()**.

> **Note : linkinfo()** est inopérante sous Windows.

mkdir (PHP 3, PHP 4 >= 4.0b1)

Crée un dossier.

```
int mkdir (string pathname, int mode)
```

mkdir() tente de créer un dossier dans le chemin *pathname*.

Notez que vous aurez à préciser le mode en base octale, ce qui signifie que vous aurez probablement un 0 comme premier chiffre. Le mode sera aussi modifié par le umask courant, que vous pouvez modifier avec la fonction **umask()**.

```
<?php
  mkdir ("/chemin/de/mon/dossier", 0700);
?>
```

mkdir() retourne TRUE en cas de succès, et FALSE en cas d'échec.

Voir aussi **rmdir()**.

move_uploaded_file (PHP 4 >= 4.0.3)

Déplace un fichier téléchargé.

```
boolean move_uploaded_file (string filename, string destination)
```

move_uploaded_file() est disponible à partir des versions PHP 3.0.16 et 4.0.2.

move_uploaded_file() s'assure que le fichier *filename* est un fichier téléchargé par HTTP POST. Si le fichier est valide, il est déplacé jusqu'à *destination*.

Si *filename* n'est pas valide, rien ne se passe, et **move_uploaded_file()** retournera FALSE.

Si *filename* est un fichier téléchargé, mais que pour une raison quelconque, il ne peut être déplacé, rien ne se passe, et **move_uploaded_file()** retourne FALSE. De plus, une alerte sera affichée.

Ce type de vérification est spécialement important s'il est possible que les fichiers téléchargés révèlent leur contenu à l'utilisateur, ou même aux utilisateurs du même système.

Voir aussi **move_uploaded_file()** et la section Chargement de fichier pour un exemple simple.

pathinfo (PHP 4 >= 4.0.3)

Retourne des informations sur un chemin système

```
array pathinfo (string path)
```

pathinfo() retourne un tableau associatif, contenant les informations sur le chemin *path*. Les éléments suivants sont retournés : *dirname*, *basename* et *extension*.

Exemple 1. Exemple avec pathinfo()

```
<?php
  $path_parts = pathinfo("/www/htdocs/index.html");
  echo $path_parts["dirname"] . "\n";
  echo $path_parts["basename"] . "\n";
  echo $path_parts["extension"] . "\n";
?>
```

Va afficher :

```
/www/htdocs
index.html
html
```

Voir aussi **dirname()**, **basename()** et **realpath()**.

pclose (PHP 3, PHP 4 >= 4.0b1)

Ferme un processus de pointeur de fichier.

```
int pclose (int fp)
```

pclose() ferme un processus de pointeur de fichier ouvert par **popen()**.

Le pointeur de fichier doit être valide, et avoir été ouvert correctement par **popen()**.

pclose() retourne le statut final du processus exécuté.

Voir aussi **popen()**.

popen (PHP 3, PHP 4 >= 4.0b1)

Crée un processus de pointeur de fichier.

```
int popen (string commet, string mode)
```

popen() ouvre un processus fils en faisant un fork de la commande.

popen() retourne un pointeur de fichier identique à celui retourné par **fopen()**, hormis le fait qu'il sera unidirectionnel (lecture seule, ou écriture seule), et doit être terminé par **pclose()**. Ce pointeur peut être utilisé avec **fgets()**, **fgetss()** et **fputs()**.

Si une erreur survient, retourne FALSE.

```
<?php
  $fp = popen("/bin/ls", "r");
?>
```

Voir aussi **pclose()**.

readfile (PHP 3, PHP 4 >= 4.0b1)

Affiche un fichier.

```
int readfile (string filename, int [use_include_path])
```

readfile() lit le fichier *filename* et l'envoit à la sortie standard.

readfile() retourne le nombre d'octets lus depuis le fichier. Si une erreur survient, retourne FALSE.

Si *filename* commence par "http://" (insensible à la casse), une connexion HTTP 1.0 sera ouverte avec le serveur spécifié, et le texte de la réponse sera affiché sur la sortie standard.

Les versions antérieures à PHP 4.0.6, ne gère pas les redirections automatiques, ce qui oblige à ajouter les slash finaux "/" pour indiquer un dossier.

Si *filename* commence par "ftp://" (insensible à la casse), une connexion FTP est ouverte avec l'hôte spécifié et la réponse du serveur est affichée. Si le serveur ne supporte les connexions passives, la requête échouera.

Si *filename* ne commence par aucun des cas précédents, le fichier sera ouvert sur l'hôte local, et envoyé à la sortie standard.

Vous pouvez utiliser le deuxième paramètre optionnel pour explorer le dossier include_path, en passant la valeur de 1.

Voir aussi **fpassthru()**, **file()**, **fopen()**, **include()**, **require()** et **virtual()**.

readlink (PHP 3, PHP 4 >= 4.0b1)

Renvoie le nom du fichier vers lequel pointe un lien symbolique.

```
string readlink (string path)
```

readlink() fait la même chose que la fonction readlink en C : elle retourne le contenu du lien symbolique repéré par *path*, ou FALSE en cas d'erreur.

Voir aussi **symlink()**, **readlink()** et **linkinfo()**.

> **Note : readlink()** est inopérante sous Windows.

rename (PHP 3, PHP 4 >= 4.0b1)

Renomme un fichier.

```
int rename (string oldname, string newname)
```

rename() rente de renommer le fichier *oldname* en *newname*.

rename() retourne TRUE en cas de succès et FALSE sinon.

rewind (PHP 3, PHP 4 >= 4.0b1)

Replace le pointeur de fichier au début.

```
int rewind (int fp)
```

rewind() replace le pointeur du fichier *fp* au début.

Si une erreur survient, retourne FALSE.

Le pointeur de fichier doit être valide, et avoir été correctement ouvert par **fopen()**.

Voir aussi **fseek()** et **ftell()**.

rmdir (PHP 3, PHP 4 >= 4.0b1)

Efface un dossier.

```
int rmdir (string dirname)
```

rmdir() tente d'effacer le dossier dont le chemin est *dirname*. Le dossier doit être vide, et le script doit avoir les autorisations adéquates.

Si une erreur survient, **rmdir()** retourne FALSE.

Voir aussi **mkdir()**.

stat (PHP 3, PHP 4 >= 4.0b1)

Renvoie les informations à propos d'un fichier.

```
array stat (string filename)
```

stat() renvoie les informations à propos du fichier filename.

stat() retourne un tableau avec les éléments suivants :

- 0 : volume
- 1 : inode
- 2 : droits d'accès au fichier (mode de protection du inode). A convertir en octal. Voir aussi **fileperms()**.
- 3 : nombre de liens
- 4 : id de l'utilisateur propriétaire
- 5 : id du groupe propriétaire
- 6 : type du volume de l'inode *
- 7 : taille en octets
- 8 : date du dernier accès
- 9 : date de la dernière modification
- 10 : date du dernier changement
- 11 : taille de bloc du système pour les entrées/sorties *
- 12 : nombre de blocs alloués

* - uniquement sur les systèmes qui supportent le type st_blksize. Les autres systèmes (i.e. Windows) retournent -1.

stat() retourne FALSE en cas d'erreur.

stat() ne gère pas les URL comme peut le faire **fopen()**.

Les résultats de **stat()** sont mis en cache. Reportez-vous à la fonction **clearstatcache()** pour plus de détails.

lstat (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Renvoie les informations à propos d'un fichier ou d'un lien symbolique.

```
array lstat (string filename)
```

lstat() est identique à **stat()** mais elle accepte aussi un lien symbolique comme argument.

lstat() retourne un tableau avec les éléments suivants :

- 0 : volume
- 1 : inode
- 2 : droits d'accès au fichier (mode de protection du inode). A convertir en octal. Voir aussi **fileperms()**.
- 3 : nombre de liens
- 4 : id de l'utilisateur propriétaire
- 5 : id du groupe propriétaire
- 6 : type du volume de l'inode *
- 7 : taille en octets
- 8 : date du dernier accès
- 9 : date de la dernière modification
- 10 : date du dernier changement
- 11 : taille de bloc du système pour les entrées/sorties *
- 12 : nombre de blocs alloués

* - uniquement sur les systèmes qui supportent le type st_blksize. Les autres systèmes (i.e. Windows) retournent -1.

Les résultats de **lstat()** sont mis en cache. Reportez-vous à la fonction **clearstatcache()** pour plus de détails.

realpath (PHP 4 >= 4.0b4)

Retourne le chemin canonique absolu.

```
string realpath (string path)
```

realpath() résoud tous les liens symboliques, et remplace toutes les références '/./', '/../' et '/' de *path* puis retourne le chemin canonique absolu ainsi trouvé. Le résultat ne contient aucun lien symbolique, '/./' ou '/../'.

Exemple 1. Exemple realpath()

```
<?php
  $real_path = realpath("../../index.php");
?>
```

symlink (PHP 3, PHP 4 >= 4.0b1)

Crée un lien symbolique.

```
int symlink (string target, string link)
```

symlink() crée un lien symbolique pour l'objet *target* avec le nom de *link*.

Voir aussi **link()** pour créer des liens durs et **readlink()** ainsi que **linkinfo()**.

> **Note : symlink()** est inopérante sous Windows.

tempnam (PHP 3, PHP 4 >= 4.0b1)

Crée un fichier avec un nom unique.

```
string tempnam (string dir, string prefix)
```

tempnam() crée un fichier temporaire unique dans le dossier *dir*. Si le dossier n'existe pas, **tempnam()** va générer un nom de fichier dans le dossier temporaire du système.

Avant PHP 4.0.6, le comportement de **tempnam()** dépendait de l'OS sous-jacent. Sous Windows, la variable d'environnement TMP remplace le paramètre *dir*; sous Linux, la variable d'environnement TMPDIR a la priorité, tandis que pour les OS en système V R4, le paramètre *dir* sera toujours utilisé, si le dossier qu'il représente existe. Consultez votre documentation pour plus de détails.

tempnam() retourne le nom du fichier temporaire, ou la chaîne `NULL` en cas d'échec.

Exemple 1. Exemple avec tempnam()

```
<?php
  $tmpfname = tempnam("/tmp", "FOO");
?>
```

tmpfile (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Crée un fichier temporaire

```
int tmpfile (void)
```

tmpfile() crée un fichier temporaire avec un nom unique, ouvert en écriture, et retourne un pointeur de fichier, identique à ceux retournés par **fopen()**. Ce fichier sera automatiquement effacé lorsqu'il sera fermé (avec **fclose()**), ou lorsque le script sera terminé.

Pour plus de détails, consultez votre documentation système sur la fonction `tmpfile(3)`, et sur `stdio.h`.

Voir aussi **tempnam()**.

touch (PHP 3, PHP 4 >= 4.0b1)

Affecte une nouvelle date de modification à un fichier.

```
int touch (string filename, int time)
```

touch() tente de forcer la date de modification du fichier nommé *filename* à la date de *time*. Si *time* est omis, c'est l'heure courante qui est utilisée.

Si le fichier n'existe pas, il est créé.

touch() retourne TRUE en cas de succès, et FALSE sinon.

Exemple 1. Exemple avec touch()

```
if ( touch($NomDeFichier) ) {
  print "La date de modification de $NomDeFichier a été fixée à maintenant";
} else {
  print "Désolé, il est impossible de changer la date de modification de $NomDeFichier";
}
```

umask (PHP 3, PHP 4 >= 4.0b1)

Change le "umask" courant.

```
int umask (int mask)
```

umask() change le umask de PHP : mask & 0777 et retourne le vieux umask. Lorsque PHP est utilisé comme module de serveur, le umask reprend sa valeur à la fin de chaque script.

umask() appelé sans argument retourne simplement le umask courant.

unlink (PHP 3, PHP 4 >= 4.0b1)

Efface un fichier.

```
int unlink (string filename)
```

unlink() efface *filename*. Identique à la fonction Unix C unlink().

unlink() retourne FALSE en cas d'échec.

Voir aussi **rmdir()** pour supprimer des dossiers.

> **Note : unlink()** ne fonctionne pas sous Windows.

XXVII. Forms Data Format

Forms Data Format (FDF) est un format de fomulaire pour les documents PDF. Vous pouvez lire la documentation (en anglais) à http://partners.adobe.com/asn/developer/acrosdk/forms.html pour plus de détails sur les tenants et les aboutissants.

> **Note :** Si vous rencontrez des problèmes de configuration de PHP avec le support fdftk, vérifiez bien que le fichier d'en-têtes `FdfTk.h` et la librairie `libFdfTk.so` sont bien situés. Elle devrait être dans les dossiers `fdftk-dir/include` et `fdftk-dir/lib`. Cela ne sera pas le cas si vous avez simplement décompressé la distribution FdfTk.

L'esprit de FDF est similaire à celui des formulaires HTML. Les différences résident dans les moyens de transmission des données au serveur, lorsque le bouton "submit" (soumettre) est pressé (ce qui est du ressort de Form Data Format) et le format de formulaire lui-même (qui est plutôt du ressort de Portable Document Format, PDF). Gérer des données FDF est un des objectifs des fonctions FDF. Mais il y en a d'autres. Vous pouvez aussi prendre un formulaire PDF, et pré-remplir les champs, sans modifier le formulaire lui-même. Dans ce cas, on va créer un document FDF (**fdf_create()**), remplir les champs (**fdf_set_value()**) et l'associer à un fichier PDF (**fdf_set_file()**). Finalement, le tout sera envoyé au client, avec le type MIME "application/vnd.fdf". Le module "Acrobat reader" de votre navigateur va reconnaître ce type MIME, et lire le fichier PDF, puis le remplis avec FDF.

Si vous éditez un fichier FDF avec un éditeur de texte, vous trouverez un catalogue d'objet avec le nom de `FDF`. Cet objet peut contenir des entrées telles que `Fields`, `F`, `Status` etc.. Les entrées les plus couramment utilisées sont `Fields`, qui indique une liste de champs de contrôle, et `F` qui contient le nom du fichier PDF a qui appartiennent ces données. Ces entrées sont désignées dans la documentation PDF sous le nom de /F-Key ou /Status-Key. La modification de ces entrées est possible avec les fonctions **fdf_set_file()** et **fdf_set_status()**. Les champs sont modifiables avec les fonctions **fdf_set_value()**, **fdf_set_opt()** etc..

Les exemples suivants montre comme évaluer les données du formulaire.

Exemple 1. Evaluer un document FDF

```
<?php
// Sauver le fichier FDF dans un fichier temporaire.
$fdffp = fopen("test.fdf", "w");
fwrite($fdffp, $HTTP_FDF_DATA, strlen($HTTP_FDF_DATA));
fclose($fdffp);
// Ouvrir le fichier temporaire, et utiliser les données.
// Le formulaire pdf contenait différents fichiers texte, avec pour nom :
// volume, date, comment, publisher, preparer, ainsi que deux boîtes
// à cocher show_publisher et show_preparer.
$fdf = fdf_open("test.fdf");
$volume = fdf_get_value($fdf, "volume");
echo "La valeur du champs volume était : '<B>$volume</B>'<br>";
$date = fdf_get_value($fdf, "date");
echo "La valeur du champs date était '<B>$date</B>'<br>";
$comment = fdf_get_value($fdf, "comment");
echo "La valeur du champs comment était '<B>$comment</B>'<br>";
if(fdf_get_value($fdf, "show_publisher") == "On") {
  $publisher = fdf_get_value($fdf, "publisher");
  echo "La valeur du champs publisher était : '<B>$publisher</B>'<br>";
} else
  echo "La valeur du champs ne doit pas être affichée.<br>";
if(fdf_get_value($fdf, "show_preparer") == "On") {
  $preparer = fdf_get_value($fdf, "preparer");
  echo "La valeur du champs preparer était  '<B>$preparer</B>'<br>";
} else
  echo "La valeur du champs Preparer ne doit pas être affiché.<br>";
fdf_close($fdf);
?>
```

fdf_open (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Ouvre un document FDF.

```
resource fdf_open (string filename)
```

fdf_open() ouvre un fichier avec formulaire. Le fichier doit contenir les données retournées par le formulaire PDF. Actuellement, le fichier doit être créée 'manuellement', en utilisant la fonction **fopen()** et en y écrivant le contenu du tableau HTTP_FDF_DATA avec la fonction **fwrite()**. Un mécanisme comparable aux formulaires HTML qui créent une variable pour chaque champs entrant, n'existe pas.

Exemple 1. Accéder aux données du formulaire

```
<?php
// Sauver le fichier FDF dans un fichier temporaire.
$fdffp = fopen("test.fdf", "w");
fwrite($fdffp, $HTTP_FDF_DATA, strlen($HTTP_FDF_DATA));
fclose($fdffp);
// Ouvrir le fichier temporaire, et utiliser les données.
$fdf = fdf_open("test.fdf");
...
fdf_close($fdf);
?>
```

Voir aussi **fdf_close()**.

fdf_close (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Ferme un document FDF.

```
boolean fdf_close (resource fdf_document)
```

fdf_close() ferme le document FDF.

Voir aussi **fdf_open()**.

fdf_create (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Crée un nouveau document FDF.

```
int fdf_create (void )
```

fdf_create() crée un nouveau document FDF. Cette fonction est nécessaire pour ceux qui veulent pré remplir les champs d'un formulaire dans un fichier PDF.

Exemple 1. Pré remplir un formulaire PDF

```
<?php
$outfdf = fdf_create();
fdf_set_value($outfdf, "volume", $volume, 0);
fdf_set_file($outfdf, "http:/testfdf/resultlabel.pdf");
fdf_save($outfdf, "outtest.fdf");
fdf_close($outfdf);
Header("Content-type: application/vnd.fdf");
```

```
$fp = fopen("outtest.fdf", "r");
fpassthru($fp);
unlink("outtest.fdf");
?>
```

Voir aussi **fdf_close()**, **fdf_save()** et **fdf_open()**.

fdf_save (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Sauver un document FDF.

```
int fdf_save (string filename)
```

fdf_save() sauve un document FDF. Le FDF Toolkit fournit un moyen d'envoyer le contenu d'un document FDF à au fichier de sortie stdout si le paramètre *filename* vaut '.'. Ceci ne fonctionne pas si PHP est sous la forme d'un module Apache. Dans ce cas, il faudra écrire le résultat dans un fichier, et utiliser **fpassthru()** pour l'afficher au client.

Voir aussi **fdf_close()** et pour avoir un exemple **fdf_create()**.

fdf_get_value (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Mot la valeur d'un champs.

```
string fdf_get_value (int fdf_document, string fieldname)
```

fdf_get_value() retourne la valeur d'un champs.

Voir aussi **fdf_set_value()**.

fdf_set_value (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Fixe la valeur d'un champs.

```
boolean fdf_set_value (int fdf_document, string fieldname, string value, int isName)
```

fdf_set_value() fixe la valeur d'un champs. Le dernier paramètre détermine si la valeur doit être convertie en nom PDF (*isName* = 1) ou affecter une chaîne PDF à un contrôle (*isName* = 0).

Voir aussi **fdf_get_value()**.

fdf_next_field_name (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Lit le nom du champs suivant.

```
string fdf_next_field_name (int fdf_document, string fieldname)
```

fdf_next_field_name() retourne le nom du champs après le champs *fieldname* ou le nom du premier champs, si le second paramètre est NULL.

Voir aussi **fdf_set_value()** et **fdf_get_value()**.

fdf_set_ap (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Fixe l'apparence d'un champs.

```
boolean fdf_set_ap (int fdf_document, string field_name, int face, string filename, int
page_number)
```

fdf_set_ap() fixe l'apparence d'un champs (i.e. la valeur de la clé /AP). Les valeurs possibles de *face* sont sont 1=FDFNormalAP, 2=FDFRolloverAP, 3=FDFDownAP.

fdf_set_status (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Fixe la valeur de la clé /STATUS.

```
boolean fdf_set_status (int fdf_document, string status)
```

fdf_set_status() fixe la valeur de la clé /STATUS.

Voir aussi **fdf_get_status()**.

fdf_get_status (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Lit la valeur de la clé /STATUS.

```
string fdf_get_status (int fdf_document)
```

fdf_get_status() retourne la valeur de la clé /STATUS.

Voir aussi **fdf_set_status()**.

fdf_set_file (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Fixe la valeur de la clé /F.

```
boolean fdf_set_file (int fdf_document, string filename)
```

fdf_set_file() Fixe la valeur de la clé /F. la clé /F est simplement une référence sur un formulaire PDF qui doit être pré-remplis. Dans un environnement web, c'est une URL (e.g. http:/testfdf/resultlabel.pdf).

Voir aussi **fdf_get_file()** et pour un exemple, **fdf_create()**.

fdf_get_file (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Lit la valeur de la clé /F.

```
string fdf_get_file (int fdf_document)
```

fdf_set_file() lit la valeur de la clé /F.

Voir aussi **fdf_set_file()**.

fdf_set_flags (PHP 4 >= 4.0.2)

Modifie une option d'un champs

```
boolean fdf_set_flags (int fdf_document, string fieldname, int whichFlags, int newFlags)
```

fdf_set_flags() modifie certaines options du champs *fieldname*.

Voir aussi **fdf_set_opt()**.

fdf_set_opt (PHP 4 >= 4.0.2)

Modifie une option d'un champs

```
boolean fdf_set_opt (int fdf_document, string fieldname, int element, string str1, string str2)
```

fdf_set_opt() modifie les options du champs *fieldname*.

Voir aussi **fdf_set_flags()**.

fdf_set_submit_form_action (PHP 4 >= 4.0.2)

Modifie l'action javascript d'un champs

```
boolean fdf_set_submit_form_action (int fdf_document, string fieldname, int trigger, string script, int flags)
```

fdf_set_submit_form_action() affecte un javascript au champs *fieldname*, exécuté lors de la validation d'un formulaire.

Voir aussi **fdf_set_javascript_action()**.

fdf_set_javascript_action (PHP 4 >= 4.0.2)

Modifie l'action javascript d'un champs

```
boolean fdf_set_javascript_action (int fdf_document, string fieldname, int trigger, string script)
```

fdf_set_javascript_action() affecte un javascript au champs *fieldname*, exécuté lors de la validation d'un formulaire.

Voir aussi **fdf_set_submit_form_action()**.

fdf_set_encoding (unknown)

Modifie l'encodage des caractères

```
bool fdf_set_encoding (int fdf_document, string encoding)
```

fdf_set_encoding() modifie l'encodage des caractères du document FDF *fdf_document*. Le paramètre *encoding* doit être un nom d'encodage valide, tels que "`Shift-JIS`" ou "`Unicode`".

fdf_set_encoding() a été ajoutée en PHP 4.0.7.

XXVIII. FTP

FTP : File Transfer Protocol (Protocole de transfert de fichiers). Ces fonctions implémentent un client pour accéder aux serveurs FTP, comme défini dans http://www.faqs.org/rfcs/rfc959.html.

Les constantes suivantes sont définies dans le module FTP : `FTP_ASCII` et `FTP_BINARY`.

Pour activer le module FTP de votre configuration PHP, il faut utiliser l'option `--enable-ftp` en PHP 4, et l'option `--with-ftp` en PHP 3 avec le script de configuration.

Exemple 1. Exemple de connexion FTP

```
<?php
// création de la connexion
$conn_id = ftp_connect("$ftp_server");
// authentification avec nom de compte et mot de passe
$login_result = ftp_login($conn_id, "$ftp_user_name", "$ftp_user_pass");
// vérification de la connexion
if ((!$conn_id) || (!$login_result)) {
        echo "La connexion FTP a échoué!";
        echo "Tentative de connexion à $ftp_server avec $user";
        die;
    } else {
        echo "Connecté à $ftp_server, avec $user";
    }
// téléchargement d'un fichier
$upload = ftp_put($conn_id, "$destination_file", "$source_file", FTP_BINARY);
// Vérification de téléchargement
if (!$upload) {
        echo "Le téléchargement Ftp a échoué!";
    } else {
        echo "Téléchargement de $source_file sur $ftp_server en $destination_file";
    }
// fermeture de la connexion FTP.
ftp_quit($conn_id);
?>
```

ftp_connect (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Ouvre une connexion FTP

```
resource ftp_connect (string host, int [port])
```

ftp_connect() retourne un flot FTP en cas de succès, et FALSE sinon.

ftp_connect() ouvre une connexion FTP avec l'hôte *host*. Le paramètre *port* spécifie le port de connexion. S'il est omis, le port 21 sera utilisé.

ftp_login (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Authentification d'une connexion FTP

```
bool ftp_login (resource ftp_stream, string username, string password)
```

ftp_login() retourne TRUE en cas de succès, et FALSE sinon.

ftp_login() authentifie le flot FTP.

ftp_pwd (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Retourne le nom du dossier courant.

```
string ftp_pwd (resource ftp_stream)
```

ftp_pwd() retourne le nom du dossier courant, ou FALSE en cas d'erreur.

ftp_cdup (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Change de dossier, et passe au dossier parent.

```
bool ftp_cdup (resource ftp_stream)
```

ftp_cdup() retourne TRUE en cas de succès, et FALSE sinon.

ftp_cdup() change de dossier, et passe au dossier parent.

ftp_chdir (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Change le dossier courant.

```
bool ftp_chdir (resource ftp_stream, string directory)
```

ftp_chdir() retourne TRUE en cas de succès, et FALSE sinon.

ftp_chdir() change le dossier courant en *directory*.

ftp_mkdir (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Crée un dossier.

```
string ftp_mkdir (resource ftp_stream, string directory)
```

ftp_mkdir() retourne le nom du dossier ainsi créé en cas de succès, et FALSE sinon.

ftp_mkdir() crée le dossier nommé *directory*.

ftp_rmdir (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Efface un dossier.

```
bool ftp_rmdir (resource ftp_stream, string directory)
```

ftp_rmdir() retourne TRUE en cas de succès, et FALSE sinon.

ftp_rmdir() efface le dossier *directory*.

ftp_nlist (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Retourne la liste des fichiers dans un dossier.

```
array ftp_nlist (resource ftp_stream, string directory)
```

ftp_nlist() retourne un tableau de nom de fichiers en cas de succès, et FALSE sinon.

ftp_rawlist (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Fait une liste détaillée de fichiers dans un dossier.

```
array ftp_rawlist (resource ftp_stream, string directory)
```

ftp_rawlist() exécute la commande FTP LIST, et retourne le résultat dans un tableau. Chaque élément du tableau correspond à une ligne du résultat de la commande. Le résultat n'est pas analysé, et est retourné brut. L'identifiant de système retourné par **ftp_systype()** sera utile pour déterminer la façon d'interpréter le résultat.

ftp_systype (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Retourne un identifiant de type de serveur FTP.

```
string ftp_systype (resource ftp_stream)
```

ftp_systype() retourne le type de serveur, ou FALSE en cas d'erreur.

ftp_pasv (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Active ou désactive le mode passif.

```
bool ftp_pasv (resource ftp_stream, int pasv)
```

ftp_pasv() retourne TRUE en cas de succès, et FALSE sinon.

ftp_pasv() active le mode passif si *pasv* est à TRUE (et le désactive si *pasv* est à FALSE). En mode passif, les données de connexion sont initiées par le client, plutôt que par le serveur.

ftp_get (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Télécharge un fichier depuis un serveur FTP.

```
bool ftp_get (resource ftp_stream, string local_file, string remote_file, int mode)
```

ftp_get() retourne TRUE en cas de succès, et FALSE sinon.

ftp_get() télécharge le fichier *remote_file* depuis le serveur FTP, et le sauve dans le fichier local *local_file*. Le mode de transfert *mode* spécifié doit être soit FTP_ASCII ou FTP_BINARY.

ftp_fget (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Télécharge un fichier depuis un serveur FTP et le sauve dans un fichier déjà ouvert.

```
bool ftp_fget (resource ftp_stream, int fp, string remote_file, int mode)
```

ftp_fget() retourne TRUE en cas de succès, et FALSE sinon.

ftp_fget() télécharge le fichier *remote_file* depuis le serveur FTP, et l'écrit dans le fichier identifié par *fp*. Le mode de transfert *mode* spécifié doit être FTP_ASCII ou FTP_BINARY.

ftp_put (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Charge un fichier sur un serveur FTP.

```
bool ftp_put (resource ftp_stream, string remote_file, string local_file, int mode)
```

ftp_put() retourne TRUE en cas de succès, et FALSE sinon.

ftp_put() enregistre le fichier *local_file* sur le serveur FTP, sous le nom de *remote_file*. Le mode de transfert *mode* spécifié doit être FTP_ASCII ou FTP_BINARY.

ftp_fput (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Charge un fichier ouvert sur un serveur FTP.

```
bool ftp_fput (resource ftp_stream, string remote_file, int fp, int mode)
```

ftp_fput() retourne TRUE en cas de succès, et FALSE sinon.

ftp_fput() charge les données issues du fichier identifié par *fp* jusqu'à la fin du fichier. Le résultat est stocké dans le fichier *remote_file* sur le serveur FTP. Le mode de transfert *mode* spécifié doit être FTP_ASCII ou FTP_BINARY.

ftp_size (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Retourne la taille d'un fichier.

```
int ftp_size (resource ftp_stream, string remote_file)
```

ftp_size() retourne la taille du fichier en cas de succès, et FALSE sinon.

ftp_size() retourne la taille d'un fichier sur un serveur FTP. Si une erreur survient, ou que le fichier n'existe pas, la valeur -1 est retournée. Certains serveurs FTP ne supportent pas cette fonction.

ftp_mdtm (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Retourne la date de dernière modification d'un fichier sur un serveur FTP.

```
int ftp_mdtm (resource ftp_stream, string remote_file)
```

ftp_mdtm() retourne un UNIX timestamp en cas de succès, et FALSE sinon.

ftp_mdtm() lit la date de dernière modification d'un fichier et retourne le UNIX timestamp. Si une erreur survient, ou si le fichier n'existe pas,la valeur -1 est retournée. Certains serveurs FTP ne supportent pas cette fonction.

ftp_rename (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Renomme un fichier sur un serveur FTP.

```
bool ftp_rename (resource ftp_stream, string from, string to)
```

ftp_rename() retourne TRUE en cas de succès, et FALSE sinon.

ftp_rename() renomme le fichier ou dossier *from* en *to*, sur le serveur *ftp_stream*.

ftp_delete (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Efface un fichier sur un serveur FTP.

```
bool ftp_delete (resource ftp_stream, string path)
```

ftp_delete() retourne TRUE en cas de succès, et FALSE sinon.

ftp_delete() efface le fichier *path* sur un serveur FTP.

ftp_site (PHP 3>= 3.0.15, PHP 4 >= 4.0RC1)

Envoie la commande SITE au serveur.

```
bool ftp_site (resource ftp_stream, string cmd)
```

ftp_site() retourne TRUE en cas de succès, et FALSE sinon.

ftp_site() envoie la commande *cmd* au serveur FTP. Les commandes SITE ne sont pas normalisées, et peuvent varier d'un serveur à l'autre. Elles permettent de gérer notamment les permissions de fichier, et les groupes.

ftp_quit (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Ferme une connexion FTP.

```
bool ftp_quit (resource ftp_stream)
```

ftp_connect() ferme la connexion *ftp_stream*.

XXIX. Fonctions

Ces fonctions effectuent les manipulations liées à la gestion des fonctions.

call_user_func_array (PHP 4 >= 4.0.4)

Appelle une fonction utilisateur avec les paramètres rassemblés en tableau

```
mixed call_user_func_array  (string function_name [, array paramarr])
```

call_user_func_array() appelle la fonction utilisateur *function_name* avec les paramètres *paramarr*, rassemblés dans un tableau. Par exemple:

```
<?php
function debug($var, $val)
    echo "***DEBUGGING\nVARIABLE: $var\nVALUE:";
    if (is_array($val) || is_object($val) || is_resource($val))
        print_r($val);
    else
        echo "\n$val\n";
    echo "***\n";
}
$c = mysql_connect();
$host = $HTTP_SERVER_VARS["SERVER_NAME"];
call_user_func_array (’debug’, array("host", $host));
call_user_func_array (’debug’, array("c", $c));
call_user_func_array (’debug’, array("HTTP_POST_VARS", $HTTP_POST_VARS));
?>
```

Voir aussi **call_user_func()**, **call_user_method()** et **call_user_method_array()**.

Note : call_user_func_array() a été ajouté en version PHP 4.05.

call_user_func (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Appelle une fonction utilisateur

```
mixed call_user_func  (string function_name [, mixed parameter [, mixed ...]])
```

call_user_func() appelle la fonction utilisateur *function_name*, et lui passe les paramètres *parameter*. Par exemple :

```
function barbier ($type) {
    print "Vous vouliez une coupe $type, pas de problème";
}
call_user_func (’barbier’, "iroquois");
call_user_func (’barbier’, "militaire");
call_user_func (’barbier’, "au bol");
```

Voir aussi **call_user_func_array()**, **call_user_method()** et **call_user_method_array()**.

create_function (PHP 4 >= 4.0.1)

Crée une fonction anonyme (style lambda)

```
string create_function (string args, string code)
```

create_function() crée une fonction anonyme, à partir des paramètres passés, et retourne un nom de fonction unique. Généralement, les arguments *args* sont présentés sous la forme d'une chaîne à guillemets simples, et la même recommandation vaut pour *code*. La raison de l'utilisation des guillemets simples est de proteger les noms de variables du remplacement par leur valeur. Si vous utilisez les guillemets doubles, n'oubliez pas d'échapper les noms de variables (i.e. `\$avar`).

Vous pouvez utiliser cette fonction pour (par exemple) créer une fonction à partir d'informations récoltés durant l'éxécution.

Exemple 1. Création d'une fonction anonyme avec create_function()

```
<?php
$newfunc = create_function('$a,$b','return "ln($a) + ln($b) = ".log($a * $b);');
echo "Nouvelle fonction anonyme : $newfunc\n";
echo $newfunc(2,M_E)."\n";
// affichera :
// Nouvelle fonction anonyme : lambda_1
// ln(2) + ln(2.718281828459) = 1.6931471805599
?>
```

Ou, pour pouvoir appliquer une fonction générique à une liste d'arguments.

Exemple 2. Traitement générique par fonction avec create_function()

```
<?php
function process($var1, $var2, $farr) {
    for ($f=0; $f < count($farr); $f++)
    echo $farr[$f]($var1,$var2)."\n";
}
// création d'une série de fonction mathématiques
$f1 = 'if ($a>=0) {return "b*a^2 = ".$b*sqrt($a);} else {return FALSE;}';
$f2 = "return \"min(b^2+a, a^2,b) = \".min(\$a*\$a+\$b,\$b*\$b+\$a);";
$f3 = 'if ($a> 0 && $b != 0) {return "ln(a)/b = ".log($a)/$b;} else {return FALSE;}';
$farr = array(
    create_function('$x,$y', 'return "un peu de trigo : ".(sin($x) + $x*cos($y));'),
    create_function('$x,$y', 'return "une hypoténuse: ".sqrt($x*$x + $y*$y);'),
    create_function('$a,$b', $f1),
    create_function('$a,$b', $f2),
    create_function('$a,$b', $f3)
    );
echo "\nUtilisation de la première liste de fonctions anonymes\n";
echo "paramétres: 2.3445, M_PI\n";
process(2.3445, M_PI, $farr);
// Maintenant une liste de fonction sur chaîne de caractères
$garr = array(
    create_function('$b,$a','if (strncmp($a,$b,3) == 0) return "** \"$a\" '.
    'et \"$b\"\n** Ces chaînes de ressemblent!! (regarde les trois pre-
miers caractères)";'),
    create_function('$a,$b,'; return "CRCs: ".crc32($a)." , ".crc32(b);'),
    create_function('$a,$b,'; return "simila-
rité(a,b) = ".similar_text($a,$b,&$p)."($p%)";')
    );
echo "\nUtilisation de la secondes liste de fonctions anonymes\n";
process("Twas brilling and the slithy toves", "Twas the night", $garr);
?>
```

Et lorsque vous utilisez le code ci-dessus, l'affichage va être

```
Utilisation de la première liste de fonctions anonymes
paramétres: 2.3445, M_PI
Un peu de trigo: -1.6291725057799
Une hypoténuse: 3.9199852871011
b*a^2 = 4.8103313314525
min(b^2+a, a^2,b) = 8.6382729035898
ln(a/b) = 0.27122299212594
Utilisation de la seconde liste de fonctions anonymes
** "Twas the night" et "Twas brilling and the slithy toves"
** Ces chaînes de ressemblent!! (regarde les trois premiers caractères)
CRCs: -725381282 , 1908338681
similarité(a,b) = 11(45.833333333333%)
```

Mais l'utilisation la plus courante des fonctions lambda est la fonction de callback, par exemple lors de l'utilisation de **array_walk()** ou **usort()**

Exemple 3. Utilisation de fonctions anonymes comme fonction de callback

```
<?php
$av = array("la ","une ","cette ","une certaine ");
array_walk($av, create_function('&$v,$k','$v = $v."maison";'));
print_r($av);  // En PHP 3 utilisez var_dump()
// affiche:
// Array
// (
//   [0] => la maison
//   [1] => une maison
//   [2] => cette maison
//   [3] => une certaine maison
// )
// un tableau de chaîne classé par taille
$sv = array("petite","moyenne","tres longue","vraiment tres longue");
print_r($sv);
// affiche:
// Array
// (
//   [0] => petite
//   [1] => moyenne
//   [2] => tres longue
//   [3] => vraiment tres longue
// )
// Tri par ordre de taille décroissant
usort($sv, create_function('$a,$b','return strlen($b) - strlen($a);'));
print_r($sv);
// outputs:
// Array
// (
//   [0] => vraiment tres longue
//   [1] => tres longue
//   [2] => moyenne
//   [3] => petite
// )
?>
```

func_get_arg (PHP 4 >= 4.0b4)

Retourne un élément de la liste des arguments

```
mixed func_get_arg (int arg_num)
```

func_get_arg() retourne l'argument à la position *arg_num* dans la liste d'argument d'une fonction utilisateur. Les arguments sont comptés en commençant à zéro. **func_get_arg()** générera une alerte si elle est appelée hors d'une fonction.

Si *arg_num* est plus grand que le nombre d'arguments passés, une alerte est générée et la fonction retourne FALSE.

```
<?php
  function foo() {
     $numargs = func_num_args();
     echo "Nombre d'arguments: $numargs<br>\n";
     if ($numargs >= 2) {
     echo "Le second argument est: " . func_get_arg (1) . "<br>\n";
     }
  }
  foo(1, 2, 3);
?>
```

func_get_arg() peut être utilisé conjointement à **func_num_args()** et **func_get_args()** pour permettre aux fonctions utilisateurs d'accepter un nombre variable d'arguments.

> **Note : func_get_arg()** a été ajoutée en PHP 4.

func_get_args (PHP 4 >= 4.0b4)

Retourne les arguments d'une fonction sous forme de tableau

```
array func_get_args (void )
```

func_get_args() retourne un tableau dont les éléments correspondent aux éléments de la liste d'arguments de la fonction. **func_get_args()** générera une alerte si elle est appelée hors d'une fonction.

```
<?php
function foo() {
    $numargs = func_num_args();
    echo "Nombre d'arguments: $numargs<br>\n";
    if ($numargs >= 2) {
    echo "Le second argument est: " . func_get_arg (1) . "<br>\n";
    }
    $arg_list = func_get_args();
    for ($i = 0; $i < $numargs; $i++) {
    echo "L'argument $i est: " . $arg_list[$i] . "<br>\n";
    }
}
foo (1, 2, 3);
?>
```

func_get_arg() peut être utilisé conjointement à **func_num_args()** et **func_get_args()** pour permettre aux fonctions utilisateurs d'accepter un nombre variable d'arguments.

> **Note : func_get_arg()** a été ajoutée en PHP 4.

func_num_args (PHP 4 >= 4.0b4)

Retourne le nombre d'arguments passé à la fonction

```
int func_num_args (void )
```

func_num_args() retourne le nombre d'arguments passé à la fonction utilisateur courante. **func_num_args()** générera une alerte si elle est appelée hors d'une fonction.

```
<?php
function foo() {
    $numargs = func_num_args();
    echo "Nombre d'arguments: $numargs\n";
}
foo (1, 2, 3);    // affiche 'Nombre d'arguments: 3'
?>
```

func_get_arg() peut être utilisé conjointement à **func_num_args()** et **func_get_args()** pour permettre aux fonctions utilisateurs d'accepter un nombre variable d'arguments.

> **Note : func_get_arg()** a été ajoutée en PHP 4.

function_exists (PHP 3>= 3.0.7, PHP 4 >= 4.0b1)

Indique si une fonction est définie.

```
boolean function_exists (string function_name)
```

function_exists() vérifie la liste des fonctions définies par l'utilisateur, et retourne TRUE si *function_name* y est trouvé, FALSE sinon.

```
<?php
  if (function_exists('imap_open')) {
    echo "Les fonctions IMAP sont disponibles.<br>\n";
  } else {
    echo "Les fonctions IMAP ne sont pas disponibles.<br>\n";
  }
?>
```

Notez qu'une fonction peut exister, même si elle est indisponible, à cause de la configuration ou des options de compilation.

Voir aussi **method_exists()**.

get_defined_functions (PHP 4 >= 4.0.4)

Liste toutes les fonctions définies

```
array get_defined_functions (void )
```

get_defined_functions() retourne un tableau multi- dimensionnel, contenant la liste de toutes les fonctions définies, aussi bien les fonctions internes à PHP que celle déjà définie par l'utilisateur. Les noms des fonctions internes sont accessibles via `$arr["internal"]`, et les fonctions utilisateur sont accessibles via `$arr["user"]`.

```
<?php
  function maligne($id, $data) {
    return "<tr><th>$id</th><td>$data</td></tr>\n";
  }
  $arr = get_defined_functions();
  print_r($arr);
?>
```

Ce script va afficher :

```
Array
(
    [internal] => Array
        (
            [0] => zend_version
            [1] => func_num_args
            [2] => func_get_arg
            [3] => func_get_args
            [4] => strlen
            [5] => strcmp
            [6] => strncmp
            ...
            [750] => bcscale
            [751] => bccomp
        )
    [user] => Array
        (
            [0] => maligne
        )
)
```

Voir aussi **get_defined_vars()**.

register_shutdown_function (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Enregistre une fonction pour exécution à l'extinction

```
int register_shutdown_function  (string func)
```

register_shutdown_function() enregistre la fonction *func* pour exécution à l'extinction du script.

Erreur courante :

Etant donné qu'aucun affichage n'est possible depuis la fonction *func*, vous ne pouvez pas y mettre d'informations de débuggage par **print()** ou **echo()**!

register_tick_function (PHP 4 >= 4.0.3)

Enregistre une fonction exécutée à chaque tick

```
void register_tick_function  (string func [, mixed arg])
```

register_tick_function() enregistre la fonction *func* pour être exécutée à chaque fois qu'un tick est appelé.

unregister_tick_function (PHP 4 >= 4.0.3)

Annule la fonction exécutée à chaque tick

```
void unregister_tick_function  (string func [, mixed arg])
```

unregister_tick_function() annule l'exécution automatique de *func* à chaque tick.

XXX. GNU Gettext

Les fonctions gettext implémente l'API NLS (Native Language Support) qui peut servir à internationaliser vos scripts PHP. Lisez la documentation GNU pour plus d'explications sur ces fonctions.

bindtextdomain (PHP 3>= 3.0.7, PHP 4 >= 4.0b1)

Fixe le chemin d'un domaine.

```
string bindtextdomain (string domain, string directory)
```

bindtextdomain() fixe le chemin du domaine *domain* à *directory*.

dcgettext (PHP 3>= 3.0.7, PHP 4 >= 4.0b1)

Remplace le domaine lors d'une recherche.

```
string dcgettext (string domain, string message, int category)
```

dcgettext() permet de remplacer le domaine courant lors de la rechrche d'un message. Elle permet aussi de spécifier une catégorie.

dgettext (PHP 3>= 3.0.7, PHP 4 >= 4.0b1)

Remplace le domaine courant.

```
string dgettext (string domain, string message)
```

dgettext() remplace le domaine courant.

gettext (PHP 3>= 3.0.7, PHP 4 >= 4.0b1)

Recherche un message dans le domaine courant.

```
string gettext (string message)
```

gettext() retourne une chaîne traduite, si elle en a trouvé une dans la table de traduction, ou bien le message *message*, s'il n'a pas été trouvé. Vous pouvez utiliser le caractère souligné (_) comme alias de cette fonction.

Exemple 1. Vérfication gettext()

```
<?php
// Choix l'allemand
putenv("LANG=de");
// Spécifie la localisation des tables de traduction
bindtextdomain("myPHPApp", "./locale");
// Choisi le domaine
textdomain("myPHPApp");
// Affiche un message de test
print (gettext ("Bienvenue sur mon application PHP"));
?>
```

textdomain (PHP 3>= 3.0.7, PHP 4 >= 4.0b1)

Fixe le domaine par défaut.

```
string textdomain (string text_domain)
```

textdomain() fixe le domaine à utiliser lors de recherche avec **gettext()**. Ce domaine dépend généralement de l'application. Le domaine par défaut précédent est retourné. Appelez cette fonction sans paramètre pour avoir la valeur courante, sans la modifier.

XXXI. GMP

Ces fonctions vous permettent de travailler avec des nombres de taille arbitraire, en utilisant la librairie GNU MP. Pour pouvoir y accéder, vous devez compiler PHP avec le support GMP en utilisant l'option `--with-gmp`.

Vous pouvez télécharger GMP sur le site de http://www.swox.com/gmp/. Ce site propose aussi un manuel GMP.

Vous devez utiliser GMP version 2 ou plus récent pour utiliser ces fonctions. Certaines d'entre elles peuvent requérir une version encore plus récente de GMP.

Ces fonctions ont été ajoutée dans PHP 4.0.4.

> **Note :** La plupart des fonctions GMP acceptent des nombres GMP comme arguments, définis ci-dessous comme `resource`. Cependant, la plus part de ces fonctions acceptent aussi des nombres et des chaînes à partir du moment où on peut les convertir en nombre. Si une fonction utilisant les entiers est plus rapide, elle sera automatiquement appelée si les arguments fournis sont des entiers. Cela se fait de manière transparante : vous pouvez donc utiliser des entiers avec les fonctions GMP sans perte de vitesse. Voir aussi **gmp_init()**.

Exemple 1. Factorielle avec GMP

```
<?php
function fact($x) {
  if($x <= 1)
        return 1;
  else
        return gmp_mul($x,fact($x-1));
}
print gmp_strval(fact(1000))."\n";
?>
```

Cet exemple va calculer factiorielle de 1000 (un plutôt grand nombre) très vite.

gmp_init (PHP 4 >= 4.0.4)

Crée un nombre GMP

```
resource gmp_init (mixed number)
```

gmp_init() crée un nombre GMP, à partir d'un entier ou d'une chaîne. Les chaînes peuvent être en décimal ou en hexadécimal. Dans ce dernier cas, la chaîne doit commencer par`0x`.

Exemple 1. Création d'un nombre GMP

```
<?php
  $a = gmp_init(123456);
  $b = gmp_init("0xFFFFDEBACDFEDF7200");
?>
```

> **Note :** Il n'est pas nécessaire d'appeler cette fonction si vous voulez utiliser des entiers ou des chaînes à la place de nombre GMP dans les fonctions GMP, comme par exemple **gmp_add()**. Les arguments de cette fonction sont automatiquement convertis en nombres GMP, si cette conversion est possible et nécessaire, en utilisant les mêmes règles que **gmp_init()**.

gmp_intval (PHP 4 >= 4.0.4)

Convertit un nombre GMP en entier.

```
int gmp_intval (resource gmpnumber)
```

gmp_intval() convertit un nombre GMP en entier.

> **Avertissement**
>
> **gmp_intval()** retourne un résultat cohérent uniquement si le nombre GMP peut être représenté par un entier PHP (i.e. type long signé). Si vous vous voulez simplement afficher un nombre GMP, utilisez plutôt **gmp_strval()**.

gmp_strval (PHP 4 >= 4.0.4)

Convertit un nombre GMP en chaîne

```
string gmp_strval (resource gmpnumber [, int base])
```

gmp_strval() convertit un nombre GMP en chaîne de caractères, dans la base *`base`*. La base par défaut est 10. Les valeurs possibles vont de 2 à 36.

Exemple 1. Conversion d'un nombre GMP en chaîne

```
<?php
  $a = gmp_init("0x41682179fbf5");
  printf("Décimal: %s, base 36: %s", gmp_strval($a), gmp_strval($a,36));
?>
```

gmp_add (PHP 4 >= 4.0.4)

Addition de 2 nombres GMP

```
resource gmp_add (resource a, resource b)
```

gmp_add() additionne les nombres GMP *a* et *b*. Le résultat est un nombre GMP.

gmp_sub (PHP 4 >= 4.0.4)

Soustraction de 2 nombres GMP

```
resource gmp_sub (resource a, resource b)
```

gmp_sub() soustrait le nombre GMP *b* de *a*. Le résultat est un nombre GMP.

gmp_mul (PHP 4 >= 4.0.4)

Multiplication de 2 nombres GMP

```
resource gmp_mul (resource a, resource b)
```

gmp_mul() multiplie les nombres GMP *a* et *b*. Le résultat est un nombre GMP.

gmp_div_q (PHP 4 >= 4.0.4)

Divisions de 2 nombres GMP

```
resource gmp_div_q (resource a, resource b [, int round])
```

gmp_div_q() divise le nombre GMP *b* par *a*. Le résultat est un entier. L'arrondi du résultat est défini par *round*, qui peut prendre l'une des valeurs suivantes :

- *GMP_ROUND_ZERO*: Le résultat est tronqué vers 0.
- *GMP_ROUND_PLUSINF*: Le résultat est tronqué vers `+infinity`
- *GMP_ROUND_MINUSINF*: Le résultat est tronqué vers `-infinity`

gmp_div_q() peut aussi être appelée **gmp_div()**.

Voir aussi **gmp_div_r()**, **gmp_div_qr()**.

gmp_div_r (PHP 4 >= 4.0.4)

Reste de la division de deux nombres GMP

```
resource gmp_div_r (resource n, resource d [, int round])
```

gmp_div_r() le reste de la division entière de *n* par *d*. Le reste a le même signe que *n*, s'il est non nul.

Voir **gmp_div_q()** pour une description du parmètre *round*.

Voir aussi **gmp_div_q()**, **gmp_div_qr()**.

gmp_div_qr (PHP 4 >= 4.0.4)

Divise deux nombres GMP

```
array gmp_div_qr (resource n, resource d [, int round])
```

gmp_div_qr() divise *n* par *d* et retourne un tableau, dont le premier élément est `[n/d]` (le quotient entier de la division) et le second est `(n - [n/d] * d)` (le reste).

Voir **gmp_div_q()** pour une description du parmètre *round*.

Exemple 1. Division de nombres GMP

```
<?php
  $a = gmp_init("0x41682179fbf5");
  $res = gmp_div_qr($a, "0xDEFE75");
  printf("Le résultat est: q - %s, r - %s", gmp_strval($res[0]), gmp_strval($res[1]));
?>
```

Voir aussi **gmp_div_q()**, **gmp_div_r()**.

gmp_div (PHP 4 >= 4.0.4)

Divise deux nombres GMP

```
resource gmp_div (resource a, resource b)
```

gmp_div() est un alias de **gmp_div_q()**.

gmp_mod (PHP 4 >= 4.0.4)

Modulo GMP

```
resource gmp_mod (resource n, resource d)
```

gmp_mod() calcule *n* modulo *d*. Le résultat est toujours positif ou nul, car le signe de *d* est ignoré.

gmp_divexact (PHP 4 >= 4.0.4)

Division exacte de nombres GMP

```
resource gmp_divexact (resource n, resource d)
```

gmp_divexact() divis *n* par *d*, en utilisant les algorithmes de "division exacte". Cette fonction ne fournit de résultats cohérents que s'il est su par avance que *d* divise *n*.

gmp_cmp (PHP 4 >= 4.0.4)

Compare des nombres GMP

```
int gmp_cmp (resource a, resource b)
```

gmp_cmp() retourne une valeur positive si `a > b`, zéro si `a = b` et négative si `a < b`.

gmp_neg (PHP 4 >= 4.0.4)

Opposé de nombre GMP

```
resource gmp_neg (resource a)
```

gmp_neg() retourne l'opposé de **gmp_neg()***a* (-1 * *a*).

gmp_abs (PHP 4 >= 4.0.4)

Valeur absolue GMP

```
resource gmp_abs (resource a)
```

gmp_abs() retourne la valeur absolue de *a*.

gmp_sign (PHP 4 >= 4.0.4)

Signe du nombre GMP

```
int gmp_sign (resource a)
```

gmp_sign() retourne le signe de *a* : 1 si *a* est positif et -1 s'il est négatif.

gmp_fact (PHP 4 >= 4.0.4)

Factorielle GMP

```
resource gmp_fact (int a)
```

gmp_fact() calcule la factorielle de *a* : `a!`.

gmp_sqrt (PHP 4 >= 4.0.4)

Racine carrée GMP

```
resource gmp_sqrt (resource a)
```

gmp_sqrt() calcule la racine carrée de *a*.

gmp_sqrtrm (unknown)

Racine carrée avec reste GMP

```
array gmp_sqrtrm (resource a)
```

gmp_sqrtrm() retourne un tableau dont le premier élément est la racine carrée entière de *a* (voir aussi **gmp_sqrt()**), et le second est le reste de (i.e., la différence entre *a* et le carré du premier élément).

gmp_perfect_square (PHP 4 >= 4.0.4)

Carré parfait GMP

```
bool gmp_perfect_square (resource a)
```

gmp_perfect_square() retourn TRUE si *a* est un carré parfait, et FALSE sinon.

Voir aussi : **gmp_sqrt()**, **gmp_sqrtrm()**.

gmp_pow (PHP 4 >= 4.0.4)

Puissance

```
resource gmp_pow (resource base, int exp)
```

gmp_pow() élève *base* à la puissance *exp*. Dans le cas de 0^0, **gmp_pow()** retourne 1.*exp* ne doit pas être négatif.

gmp_powm (PHP 4 >= 4.0.4)

Puissance et modulo

```
resource gmp_powm (resource base, resource exp, resource mod)
```

gmp_powm() calcule (*base* puissnace *exp*) modulo *mod*. Si *exp* est négatif, le résultat est indéfini.

gmp_prob_prime (PHP 4 >= 4.0.4)

Nombre GMP probablement premier

```
int gmp_prob_prime (resource a [, int reps])
```

Si **gmp_prob_prime()** retourne 0, *a* est défini comme non premier. Si **gmp_prob_prime()** retourne 1, alors *a* est "probablement" premier. Si **gmp_prob_prime()** retourne 2, alors *a* est sÛrement premier. *reps* peut raisonnablement varier de 5 à 10 (par défaut, c'est 10); une valeur supérieure réduit la probabilité qu'un nombre non premier soit identifié comme "probablement" premier.

gmp_prob_prime() utilise le test de probabilité Miller-Rabin.

gmp_gcd (PHP 4 >= 4.0.4)

PGCD

```
resource gmp_gcd (resource a, resource b)
```

gmp_gcd() calcule de PGCD (plus grand commun diviseur) de *a* et *b*. Le résultat est toujours positif, même si l'un des deux (ou les deux) nombres est négatif.

gmp_gcdext (PHP 4 >= 4.0.4)

PGCD étendu

```
array gmp_gcdext (resource a, resource b)
```

gmp_gcdext() calcule les entiers g, s, et t, tels que `a*s + b*t = g = gcd(a,b)`, où gcd est le pgcd de *a* et *b*. La fonction retourne un tableau avec les éléments g, s et t.

gmp_invert (PHP 4 >= 4.0.4)

Inverse modulo

```
resource gmp_invert (resource a, resource b)
```

gmp_invert() calcule l'invers de *a* modulo *b*. Retourne `FALSE` si un tel inverse n'existe pas.

gmp_legendre (PHP 4 >= 4.0.4)

Symbole de Legendre

```
int gmp_legendre (resource a, resource p)
```

gmp_legendre() calcule le symbole de Legendre (http://www.utm.edu/research/primes/glossary/LegendreSymbol.html) de *a* et *p*. *p* doit être positif et impair.

gmp_jacobi (PHP 4 >= 4.0.4)

Symbole de Jacobi

```
int gmp_jacobi (resource a, resource p)
```

gmp_jacobi() calcule le symbole de Jacobi (http://www.utm.edu/research/primes/glossary/JacobiSymbol.html) de *a* et *p*. *p* doit être positif et impair.

gmp_random (PHP 4 >= 4.0.4)

Nombre GMP aléatoire

```
resource gmp_random (int limiter)
```

gmp_random() génère un nombre alétoire. Ce nombre est limité par *limiter*. Si *limiter* est négatif, un nombre négatif est généré.

gmp_and (PHP 4 >= 4.0.4)

ET logique

```
resource gmp_and (resource a, resource b)
```

gmp_and() calcule le ET logique de *a* et *b*.

gmp_or (PHP 4 >= 4.0.4)

OU logique

```
resource gmp_or (resource a, resource b)
```

gmp_or() calcule le OU logique de *a* et *b*.

gmp_xor (PHP 4 >= 4.0.4)

OU exclusif logique

```
resource gmp_xor (resource a, resource b)
```

gmp_or() calcule le OU exclusif logique de *a* et *b*.

gmp_setbit (PHP 4 >= 4.0.4)

Modifie un bit

```
resource gmp_setbit (resource &a, int index [, bool set_clear])
```

gmp_setbit() modifie le bit *index* dans *a*. *set_clear* indique si le bit est mis à 0 ou 1. Par défaut, il est mis à 1.

gmp_clrbit (PHP 4 >= 4.0.4)

Annule un bit

```
resource gmp_clrbit (resource &a, int index)
```

gmp_clrbit() met le bit *index* à 0 dans le nombre GMP *a*.

gmp_scan0 (PHP 4 >= 4.0.4)

Recherche 0

```
int gmp_scan0 (resource a, int start)
```

gmp_scan0() recherche dans *a*, en commencant à la position *start*, vers les bits de poids lourd, jusqu'à ce qu'elle rencontre un bit à 0. Sa position est alors retournée.

gmp_scan1 (PHP 4 >= 4.0.4)

Recherche 1

```
int gmp_scan1 (resource a, int start)
```

gmp_scan1() recherche dans *a*, en commencant à la position *start*, vers les bits de poids lourd, jusqu'à ce qu'elle rencontre un bit à 1. Sa position est alors retournée.

gmp_popcount (PHP 4 >= 4.0.4)

Compte de population

```
int gmp_popcount (resource a)
```

gmp_popcount() dénombre la population de *a*.

gmp_hamdist (PHP 4 >= 4.0.4)

Distance de Hamming

```
int gmp_hamdist (resource a, resource b)
```

gmp_hamdist() retourne la distance de Hamming entre *a* et *b*. Les deux paramètres doivent être strictement positif.

XXXII. HTTP

Ces fonctions permettent de travailler sur les informations transmises au navigateur, via le protocole HTTP.

header (PHP 3, PHP 4 >= 4.0b1)

Envoie une en-tête HTTP.

```
int header (string string)
```

header() permet de spécifier une en-tête HTTP lors de l'envoi des fichiers HTML. Reportez-vous à HTTP 1.1 Specification (http://www.w3.org/Protocols/rfc2616/rfc2616) pour plus d'informations sur les en-têtes HTTP.

> **Note :** La fonction **header()** doit être appelée avant la première balise HTML, et avant n'importe quel envoi de commande PHP. C'est une erreur très courante que de lire du code avec la fonction **include()** ou avec auto_prepend et d'avoir des espaces ou des lignes vides dans ce code qui produisent un début de sortie avant que **header()** n'ait été appelé.

Il y a cependant deux en-têtes spéciales. Le premier est "Location". Non seulement il renvoie une en-tête au client, mais en plus, il envoie un statut de redirection à Apache. Du point de vue de l'auteur de script, cela importe peu, mais pour ceux qui connaissent les rouages internes d'Apache, c'est primordial.

```
<?php
  header("Location: http://www.php.net/");
/* Redirige le client vers le site PHP */
  exit();
/* Assure que le code ci-dessous n'est jamais exécuté. */
?>
```

Le deuxième type d'appel spécial regroupe toutes les en-têtes qui commencent par "HTTP/" (la casse n'est pas importante). Par exemple, si vous avez votre page d'erreur 404 Apache qui pointe sur un script PHP, c'est une bonne idée que de vous assurer que le script PHP génère une erreur 404. La première chose à faire dans votre script est :

```
<?php
  header("http/1.0 404 Not Found");
?>
```

Les scripts PHP génèrent souvent du HTML dynamiquement, qui ne doit pas être mis en cache, ni par le client, ni par les proxy intermédiaires. On peut forcer la désactivation du cache de nombreux clients et proxy avec

```
<?php
  header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");             // Date du passé
  header("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT"); // toujours modifié
  header("Cache-Control: no-cache, must-revalidate");           // HTTP/1.1
  header("Pragma: no-cache");                                   // HTTP/1.0
?>
```

N'oubliez jamais que **header()** doit être appelée avant que le moindre contenu ne soit envoyé, soit par des lignes HTML habituelles dans le fichier, soit par des affichges PHP. Une erreur très classique est de lire un fichier avec **include()** ou **require()**, et de laisser des espaces ou des lignes vides, qui généreront un affichage avant que la fonction **header()** ne soit appelée. Le même problème existe avec les fichiers PHP/HTML standards.

```
<?php
  require("user_logging.inc")
?>
<?php
  header("Content-Type: audio/x-pn-realaudio");
?>
// Erreur :  Notez la ligne blanche ci-dessus
```

Voir aussi **headers_sent()**.

headers_sent (PHP 3>= 3.0.8, PHP 4)

Indique si les entêtes HTTP ont déjà été envoyés

```
boolean headers_sent (void)
```

headers_sent() retourne TRUE si les entêtes HTTP ont déjà été envoyés, et FALSE sinon.

Voir aussi **header()**.

setcookie (PHP 3, PHP 4 >= 4.0b1)

Envoie un cookie

```
int setcookie (string name [, string value [, int expire [, string path [, string domain [, int secure]]]]])
```

setcookie() définit un cookie qui sera envoyé avec le reste des en-têtes. Les cookies doivent passer avant toute autre en-tête (c'est une restriction des cookies, pas de PHP). Cela vous impose d'appeler cette fonction avant toute balise <HTML> ou <HEAD>.

Tous les arguments sauf *name* (nom) sont optionnels. Si seul le nom est présent, le cookie portant ce nom sera supprimé du navigateur de l'internaute. Vous pouvez aussi utiliser une chaîne vide comme valeur, pour ignorer un argument. Le paramètre *expire* est un timestamp UNIX, du même genre que celui retourné par **time()** ou **mktime()**. Le paramètre *secure* indique que le cookie doit être uniquement transmis à travers une connexion HTTPS sécurisée.

Erreurs communes :

- Les cookies ne seront accessibles qu'au chargement de la prochaine page, ou au rechargement de la page courante.
- Les cookies doivent être effacés avec les mêmes paramètres que ceux utilisés lors de leur création.

En PHP 3, les appels multiples à **setcookie()** dans le même script seront effectués dans l'ordre inverse. Si vous essayez d'effacer un cookie avant d'insérer une nouvelle valeur, vous devez placer l'insertion avant l'effacement. En PHP 4, les appels multiples à **setcookie()** sont effectués dan un ordre naturel.

Les appels multiples à **setcookie()** dans la même page seront réalisés dans l'ordre inverse. Si vous essayez d'effacer un cookie avant d'insérer une autre valeur, il faut placer l'insertion avant l'effacement.

Quelques exemples :

Exemple 1. Exemples avec setcookie()

```
<?php
  setcookie("TestCookie","Valeur de test");
  setcookie("TestCookie",$value,time()+3600);  /* expire dans une heure */
  setcookie("TestCookie",$value,time()+3600,"/~rasmus/",".utoronto.ca",1);
?>
```

Notez que la partie "valeur" du cookie sera automatiquement encodée URL lorsque vous envoyez le cookie, et lorsque vous le recevez, il sera automatiquement décodé, et affecté à la variable du même nom que le cookie. Pour voir le résultat, essayez les scripts suivants :

```
<?php
  echo $TestCookie;
  echo $HTTP_COOKIE_VARS["TestCookie"];
?>
```

Vous pouvez aussi utiliser les cookies avec des tableaux, en utilisant la notation des tableaux. Cela a pour effet de créer autant de cookies que votre tableau a d'éléments, mais lorsque les cookies seront reçus par PHP, les valeurs seront placées dans un tableau :

```
<?php
  setcookie( "cookie[three]", "cookiethree" );
  setcookie( "cookie[two]", "cookietwo" );
  setcookie( "cookie[one]", "cookieone" );
  if ( isset( $cookie ) ) {
    while( list( $name, $value ) = each( $cookie ) ) {
      echo "$name == $value<br>\n";
    }
  }
?>
```

Pour d'autres informations sur les cookies, jetez un oeil sur http://www.netscape.com/newsref/std/cookie_spec.html.

Microsoft Internet Explorer 4 utilisé avec le Service Pack 1 ne gère pas bien les cookies qui possèdent un paramètre *`path`*.

Netscape Communicator 4.05 et Microsoft Internet Explorer 3.x semblent ne pas gérer correctement les cookies lorsque *`path`* et *`time`*> ne sont pas fournis.

XXXIII. Hyperwave

Introduction

Hyperwave a été developpé par IICM (http://www.iicm.edu) à Graz. Son nom original était Hyper-G et il a pris le nom de Hyperwave lors de sa commercialisation (en 1996, si mes souvenirs sont bons).

Hyperwave n'est pas gratuit. La version actuelle est la 4.1, disponible à www.hyperwave.com (http://www.hyperwave.com/). Une version limitée à 30 jours peut être demandée.

HIS est un système d'information similaire à une base de données, (HIS, Hyperwave Information Server). HIS se concentre sur l'enregistrement et la gestion des documents. Un document peut être n'importe quelle donnée, qui peut être stockée dans un fichier. Chaque document est accompagné par un enregistrement. Cet enregistrement contient des méta données à propos du document. Ces méta données sont des listes d'attributs qui peuvent être étendues par l'utilisateur. Un attribut est une paire clé/valeur de la forme : clé =valeur. L'enregistrement complet contient autant de paire que le désire l'utilisateur. Le nom d'un attribut n'a pas besoin d'être unique, c'est à dire qu'une même clé peut apparaître plusieurs fois dans un enregistrement. Cela peut être utile si vous devez donner un titre à votre document en plusieurs langues, par exemple. Dans un cas pareil, la convention est que chaque valeur de titre est précédée par deux lettres et deux points, tel que : 'fr:Titre en français' ou 'ge:Titel in deutsch'. D'autres attributs comme une description ou des mots clés sont aussi suceptibles de recourir à ce genre de procédé. Vous pouvez aussi remplacer l'abréviation de langage par une autre chaîne, tant qu'elle est séparée de la valeur par les deux points.

Chaque enregistrement a une représentation native qui contient toutes les paires clé/valeur, séparées par un retour à la ligne. L'extension Hyperwave reconnaît une autre représentation qui est un tableau associatif, oú les attributs servent de clés. Les attributs multilingues étant géré sous la forme d'un autre tableau associatif, dont les clés sont les chaînes de langue. En fait, tous les attributs multiformes sont gérés sous la forme de tableau associatif. (Cela n'est pas encore complètement codé. Uniquement les attributs de titre, description et mot clés sont traités correctement).

En dehors des documents, tous les hyper liens contenus dans un documents sont enregistrés dans un autre enregistrement. Les hyperliens qui sont à l'intérieur d'un document en seront supprimés, et enregistrés dans des objets particuliers, au moment de l'insertion dans la base de données. L'enregistrement des hyper-liens contient les informations d'origine et d'objectif. Afin d'accéder au document original, vous devre lire le document sans les liens, puis lire les liens, et les réinsérer (les **hw_pipedocument()** et **hw_gettext()** le font pour vous. L'avantage de séparer les liens du document est évident : une fois qu'un document, cible d'un hyperlien, a été renommé, le liens peut facilement être modifié. Le document contenant le lien n'est pas modifié pour autant. Vous pouvez même ajouter un lien à un document sans le modifier.

Dire que **hw_pipedocument()** et **hw_gettext()** font l'insertion automatiquement n'est pas aussi simple qu'il n'y paraît. L'insertion implique une certaine hiérarchie de document. Sur un serveur web, la hiérarchie est fournie par le système de fichiers, mais Hyperwave dispose de sa propre hiérarchie et les noms de fichiers ne reflètent pas la position d'un objet dans cette hiérarchie. Ainsi, la création de liens requière en premier lieu la construction de la hiérarchie et de l'espace des noms dans une hiérarchie web et un espace de nom web. La différence fondamentale entre Hyperwave et le web est qu'il y a une distinction claire en les noms et la hiérarchie dans Hyperwave. Le nom ne contient aucune information à propos de la position de l'objet dans la hiérarchie. Sur le web, le nom contient les informations de localisation dans la hiérarchie. Cela conduit à deux méthodes de d'accès : soit la hiérarchie Hyperwave et le nom de l'objet sont inscrit dans l'URL. Pour simplifier les choses, une deuxième approche est pratiquée. L'objet Hyperwave de nom 'mon_objet' correspond à l'URL 'http://hote/mon_objet', peu importe alors oú il est rangé dans la hiérarchie. Un objet dont le nom est 'parent/mon_objet' peut être le fils de l'objet 'mon_objet' dans la hiérarchie Hyperwave, bien que ce soit le contraire en convention web, et cela risque de perturber l'utilisateur.

Ayant pris cette décision, un deuxième problème surgit : comment faire l'interface avec PHP ? L' URL http://hote/mon_objet n'appelera aucun script PHP à moins que vous ne demandiez à votre serveur web de le remplacer par autre chose, comme par exemple : 'http://host/php3_script/mon_objet' et le script 'php3_script' utilise la variable $PATH_INFO pour rechercher l'objet 'mon_objet' sur le serveur Hyperwave. Il y a juste un petit inconvénient , qui peut facilement être corrigé. Réécrire une URL ne vous permettra aucun accès aux autres documents du serveur web. Un script de recherche dans le serveur Hyperwave serait impossible. Il vous faudra donc au moins une autre règle pour exclure certaines URL, comme par exemple celles qui commencent par http://host/Hyperwave. Voici de manière simple, la manière de partager un espace de nom entre un serveur web et un serveur Hyperwave serveur.

Basé sur le mécanisme précédent, on trouve l'insertion dans les documents.

Il est plus compliqué d'avoir PHP ne fonctionne pas comme un module de serveur, ou un scrip CGI, mais comme une application indépendante. Dans ce cas, il est utile d'inscrire la hiérarchie et le nom de fichier Hyperwave dans le système

de fichier. Mais comme cela risque d'entrer en conflit avec le séparateur de dossier ('/'), il faut le remplacer par un autre caractère,. '_'.

Le protocole réseau pour communiquer avec un serveur Hyperwave est appelé HG-CSP (http://www.hyperwave.de/7.17-hg-prot) (Hyper-G Client/Server Protocol). Il est basé sur des messages qui initie des actions, comme par exemple, lire l'entête de fichier. Dans les premières versions de Hyperwave Server deux clients natifs (Harmony, Amadeus) étaient fournis pour permettre la communication avec le serveur. Ils ont disparu lors de la commercialisation de Hyperwave. En tant qu'ersatz, un client appelé wavemaster est désormais fourni. wavemaster est un espèce ce convertisseur de protocole de HTTP en HG-CSP. L'idée est de faire toute l'administration de la base et la visualisation des documents par une interface web. Le wavemaster implémente un jeu d'emplacement pour certaines actions de personnalisation de l'interface. Ce jeu est appelé PLACE language. PLACE pêche encore par le manque de nombreuse fonctions de programmations, et le manque d'évolutivité. Cela a conduit à l'utilisation de JavaScript ce qui ne rend pas la vie facile.

Que PHP supporte Hyperwave permet de combler ces manques. PHP implémente tous les messages définis par HG-CSP mais fourni d'autres commandes puissantes, comme par exemple, celle de lire des documents complets.

Hyperwave dispose de sa propre terminologie pour localiser certaines informations. Cette terminologie a été largement étendue. Presque toutes les fonctions utilisent l'un des types de données suivants :

- object ID: un entier, unique pour chaque objet sur le serveur Hyperwave. C'est aussi un des attributs de l'enregistrement de l'objet (ObjectID).Les object ids sont souvent utilisées comme paramètre d'entrée pour spécifier un objet.
- object record: Une chaîne contenant des paires clé=valeur. Les paires sont séparées par un retour à la ligne. Un enregistrement d'objet peut facilement être converti en tableau, avec la fonction **hw_objrec2array()**. De nombreuses fonctions retournent un object records. Ces fonctions ont leur nom qui finit par obj.
- object array: Un tableau associatif qui contient tous les attributs d'un objet. La clé est l'attribut. Si un attribut apparaît plusieurs fois, il sera représenté sous la forme d'un tableau associatif ou indexé. Les attributs qui dépendent des langues (comme title, keyword ou description) seront représentés sous la forme d'un tableau associatif, dont les clés seront les abréviations de langues. Tous les autres attributs à valeur multiple prendront la forme d'un tableau indexé.
- hw_document: Ce type est un nouveau type de données, qui contient le document lui même, comme par exemple HTML, PDF etc. Il est optimisé pour les documents HTML mais peut s'utiliser avec n'importe quel format.

De nombreuses fonctions qui retournent un tableau d'enregistrements, retournent aussi un tableau associé, avec des informatiosn statistiques. Ce tableau est le dernier élément du tableau d'enregistrements. Les statistiques contiennent les entrées suivantes :

Hidden

Nombre d'objets dont l'attribut PresentationHints est Hidden.

CollectionHead

Nombre d'objet dont l'attribut PresentationHints est CollectionHead.

FullCollectionHead

Nombre d'objet dont l'attribut PresentationHints est FullCollectionHead.

CollectionHeadNr

Index du premier objet du tableau d'enregistrement avec l'attribut PresentationHints à CollectionHead.

FullCollectionHeadNr

Index du premier objet du tableau d'enregistrement avec l'attribut PresentationHints est FullCollectionHead.

Total

Total: Nombre d'enregistrements.

Intégration avec Apache

L’extension Hyperwave est utilisée de manière optimale lorsque PHP est compilé comme module Apache. Dans ce cas, le serveur Hyperwave sous jacent peut être caché quasiment totalement aux utilisateurs, si Apache utilise son moteur d’écriture. Les explications suivantes vous éclaireront :

Etant donné que PHP avec l’extension Hyperwave et Apache tend à remplacer la solution native basé sur Wavemaster, je vais supposer que le serveur Apache servira seulement d’interface Hyperwave. Ce n’est pas nécessaire, mais cela simplifie grandment la configuration. Le concept est très simple. Premièrement, vous avez besoin d’un script PHP qui évalue la variable PATH_INFO et considère que cette valeur est un objet Hyperwave. Appelons ce script ’Hyperwave’. L’URL http://votre.hote/Hyperwave/nom_objet retourne alors l’objet Hyperwave dont le nom est ’nom_objet’. Le script doit alors réagir suivant le type de l’objet. Si c’est un groupe, il devra probablement retourner une liste de fils. Si c’est un document, il pourra retourner son type MIME et son contenu. Une amélioration peut être obtenue en utilisant le moteur de réécriture d’Apache. D’un point de vue utilisateur, il est plus direct si l’URL http://votre.hote/nom_objet retourne l’objet. La règle de réécriture est simple :

```
RewriteRule ^/(.*) /usr/local/apache/htdocs/HyperWave/$1 [L]
```

Maintenant toutes les URL pointent sur un objet Hyperwave. Cela conduit à un problème simple. Il n’y a pas d’autre façon d’exécuter, c’est à dire rechercher, un autre script que ce script ’Hyperwave’. Cela pourra être corrigé avec une autre règle telle que:

```
RewriteRule ^/hw/(.*) /usr/local/apache/htdocs/hw/$1 [L]
```

Le dossier `/usr/local/apache/htdocs/hw` sera ainsi reservé pour d’autres scripts et fichiers. Assurez vous que cette règle est évaluée avant la première règle que nous avons définie. Il y a juste un léger inconvénient : tous les objets Hyperwave qui commencent par ’hw/’ seront cachés. Alors, assurez vous que vous n’utilisez pas de tels noms. Si vous avez besoin d’autres dossiers, par exemple, un dossier d’images, ajoutez simplement d’autres règles. N’oubliez pas de lancer le moteur de réécriture avec

```
RewriteEngine on
```

Mon expérience m’a montré que vous aurez besoin des scripts suivants :

- Retourne l’objet lui même
- Pour autoriser la recherche
- S’identifier
- Choisir une configuration
- Un script pour chaque fonction supplémentaire, comme afficher un objet, afficher des informations sur les utilisateurs, afficher le statut du serveur, etc...

A faire

Il reste encore beaucoup à faire :

- La fonction hw_InsertDocument doit être séparée en deux : **hw_insertobject()** et **hw_putdocument()**.
- Les noms de nombreuses fonctions ne sont pas encore fixés.
- La plupart des fonctions requièrent la connexion courante comme premier paramètre. Cela conduit à beaucoup de frappe clavier, même si il n’y a souvent qu’une seule connexion en jeu. Une connexion par défaut améliorerait ceci.

hw_Array2Objrec (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Convertit un tableau en un objet.

```
string hw_array2objrec (array object_array)
```

hw_array2objrec() convertit un *object_array* en un objet. Les attributs multiples tels que 'Title' en différentes langues seront traités correctement.

Voir aussi **hw_objrec2array()**.

hw_Children (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Liste des object ids des objets fils.

```
array hw_children (resource connection, resource objectID)
```

hw_children() retourne un tableau avec des object ids. Chaque object id est celui d'un des fils du groupe dont l'id est *objectID*. Ce tableau contient tous les fils, documents et groupes.

hw_ChildrenObj (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Liste des object records des objets fils.

```
array hw_childrenobj (resource connection, resource objectID)
```

hw_childrenobj() retourne un tableau avec des object records. Chaque object records est celui d'un des fils du groupe dont l'id est *objectID*. Ce tableau contient tous les fils, documents et groupes

hw_Close (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Ferme la connexion Hyperwave.

```
int hw_close (resource connection)
```

hw_close() retourne FALSE si la connexion n'est pas valide, et sinon, TRUE. Ferme la connexion connection à un serveur Hyperwave.

hw_Connect (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Ouvre une connexion Hyperwave.

```
resource hw_connect (string host, int port, string username, string password)
```

hw_connect() ouvre une connexion Hyperwave et retourne un identifiant de connexion, en cas de succès, ou FALSE, si la connexion n'a pas pu être créée. Chaque argument doit être entouré de guillemets, sauf le numéro de port. Les arguments *username* et *password* sont optionnels, et peuvent être ignorés. Dans ce cas, aucune identification ne sera faite au niveau du serveur. Cela revient à s'identifier en tant qu'utilisateur anonyme. Cette fonction retourne un identifiant de connexion qui sera nécessaire aux autres fonctions Hyperwave. Vous pouvez avoir plusieurs connexions simultanées. N'oubliez pas que les mots de passe ne sont pas cryptés.

Voir aussi **hw_pconnect()**.

hw_Cp (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Copie des objets.

```
resource hw_cp (resource connection, array object_id_array, int destination_id)
```

hw_cp() copie les objets ayant les identifiants *object_id_array*, et crée un groupe ayant l'object id *destination_id*.

La valeur retournée est le nombre d'objets copiés.

Voir aussi **hw_mv()**.

hw_Deleteobject (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Efface des objets.

```
int hw_deleteobject (resource connection, int object_to_delete)
```

hw_deleteobject() efface l'objet dont l'identifiant est *object_to_delete*. Toutes les instances de l'objets seront effacées.

hw_deleteobject() retourne TRUE si aucune erreur ne survient, et sinon, FALSE.

Voir aussi **hw_mv()**.

hw_DocByAnchor (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Identifiant d'objet de l'objet dans l'ancrage.

```
resource hw_docbyanchor (resource connection, int anchorID)
```

hw_docbyanchor() retourne l'dentifiant d'objet de l'objet dans l'ancrage *anchorID*.

hw_DocByAnchorObj (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Attributs de l'objet dans l'ancrage.

```
string hw_docbyanchorobj (resource connection, int anchorID)
```

hw_docbyanchorobj() retourne les attributs du document qui correspond à *anchorID*.

hw_DocumentAttributes (PHP 3>= 3.0.3, PHP 4 4.0b1 only)

Object record de hw_document.

```
string hw_documentattributes (int hw_document)
```

hw_documentattributes() retourne les attributs du document.

Voir aussi **hw_documentbodytag()**, **hw_documentsize()**.

hw_DocumentBodyTag (PHP 3>= 3.0.3, PHP 4 4.0b1 only)

Balise de corps d'un document.

```
string hw_documentbodytag (int hw_document)
```

hw_documentbodytag() retourne la balise BODY du document.Si le document est un document HTML , la balise BODY doit être affichée avant le document.

Voir aussi **hw_documentattributes()**, **hw_documentsize()**.

hw_DocumentContent (PHP 3>= 3.0.8)

Contenu d'un document.

```
string hw_documentcontent (int hw_document)
```

hw_documentcontent() retourne la balise BODY du document.Si le document est un document HTML , la balise BODY doit être affichée avant le document.

Voir aussi **hw_documentattributes()**, **hw_documentsize()**, **hw_documentsetcontent()**.

hw_DocumentSetContent (PHP 3>= 3.0.8)

Modifie/remplace le contenu d'un document.

```
string hw_documentsetcontent (int hw_document, string content)
```

hw_documentsetcontent() modifie/remplace le contenu d'un document. Si le document est un document HTML, le contenu représente tout qui est placé au dela de la balise BODY. Les informations de HEAD et de la balise BODY sont enregistrés dans les attributs. Si vous fournissez aussi ces informations dans le corps du document, le serveur Hyperwave modifiera les attributs. Cela n'est cependant pas une bonne idée. Si la fonction échoue, l'ancien contenu est restauré.

Voir aussi **hw_documentattributes()**, **hw_documentsize()**, **hw_documentcontent()**.

hw_DocumentSize (PHP 3>= 3.0.3, PHP 4 4.0b1 only)

Taille d'un document.

```
int hw_documentsize (int hw_document)
```

hw_documentsize() retourne la taille du document en octets.

Voir aussi **hw_documentbodytag()** et **hw_documentattributes()**.

hw_ErrorMsg (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Retourne un message d'erreur.

```
string hw_errormsg (resource connection)
```

hw_errormsg() retourne une chaîne contenant le dernier message d'erreur, ou 'No Error' (pas d'erreur). Si FALSE est retourné, cette fonction a échoué. Ce message est relatif à la dernière commande exécutée.

hw_EditText (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Retourne un document texte.

```
int hw_edittext (resource connection, int hw_document)
```

hw_edittext() charge le texte du document sur le serveur. Les attributs du document ne doivent pas être modifiés tant que le document est en train d'être édité. Cette fonction n'est disponible que sur les documents texte. Elle n'ouvrira pas de canal de transfert, et donc, bloquera le script durant le transfert.

Voir aussi **hw_pipedocument()**, **hw_free_document()**, **hw_documentbodytag()**, **hw_documentsize()**, **hw_outputdocument()** et **hw_gettext()**.

hw_Error (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Retourne le code d'erreur.

```
int hw_error (resource connection)
```

hw_error() retourne le code d'erreur de la dernière erreur. Si la valeur 0 est retournée, c'est qu'il n'y avait pas d'erreur. L'erreur se rapporte à la dernière commande.

hw_Free_Document (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Détruit un document.

```
int hw_free_document (resource hw_document)
```

hw_free_document() détruit un document Hyperwave.

hw_GetParents (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Identifiant d'objet des parents.

```
array hw_getparents (resource connection, resource objectID)
```

hw_getparents() retourne un tableau indexé avec les identifiants des objets parents de *objectID*.

hw_GetParentsObj (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Attributs des parents.

```
array hw_getparentsobj (resource connection, resource objectID)
```

hw_getparentsobj() retourne un tableau indexé, avec les attributs et un tableau associé, d'informations statistiques à propos des attributs. Ce tableau associé est le dernier élément du tableau retourné. Chaque attribut appartient au père de l'objet *objectID*.

hw_GetChildColl (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Identifiant d'objet des groupes fils.

```
array hw_getchildcoll (resource connection, resource objectID)
```

hw_getchildcoll() retourne un tableau contenant les identifiants d'objets des groupes fils du groupe *objectID*. Cette fonction ne retournera pas d'identifiants d'objets des documents fils.

Voir aussi **hw_getchilddoccoll()**.

hw_GetChildCollObj (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

object records d'un groupe d'enfants.

```
array hw_getchildcollobj (resource connection, resource objectID)
```

hw_getchildcollobj() retourne un tableau d'object record. Chaque object records appartient à un groupe d'enfants de la collection *objectID*. La fonction ne retournera pas de documents enfants.

Voir aussi **hw_childrenobj()**, **hw_getchilddoccollobj()**.

hw_GetRemote (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Retourne un document distant.

```
resource hw_getremote (resource connection, resource objectID)
```

hw_getremote() retourne un document distant. Les documents distants sont , en Hyperwave, des documents lus depuis une source externe. La plus part des documents éloignés sont des pages web externes, ou des requêtes sur une base de données. Afin de pouvoir accéder à des sources externes, grâce aux documents distants, Hyperwave introduit l'interface HGI (Hyperwave Gateway Interface) qui est similaire à CGI. Actuellement, seuls les protocoles de FTP, HTTP et certaines bases de données sont accessibles avec HGI. **hw_getremote()** retourne le document de la source distante. Si vous voulez utiliser cette fonction, il vous faut vous familiariser avec HGIs. Il est aussi préférable d'utiliser PHP plutôt que Hyperwave pour accéder aux sources externes. Le support des bases de données sera plus difficile avec Hyperwave que PHP.

Voir aussi **hw_getremotechildren()**.

hw_GetRemoteChildren (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Retourne les fils d'un document distant.

```
resource hw_getremotechildren (resource connection, string object record)
```

hw_getremotechildren() retourne les fils d'un document distant. Les fils d'un document distant sont des documents distants eux mêmes. Cela est cohérent si une requête sur une base de données doit être rendu plus sélective, comme expliqué dans Hyperwave Programmers' Guide. Si le nombre de fils est 1 la fonction va retourner le document lui même, la fonction retournera le document lui même, formaté Hyperwave Gateway Interface (HGI). Si le nombre de fils est supérieur 1 la fonction retournera un tableau d'attributs, qui pourra servir à une nouvelle requête avec **hw_getremotechildren()**. Ces attributs sont virtuels et n'existent pas sur le serveur Hyperwave, et ainsi, n'ont pas d'identifiant d'objet valide. L'ordre exact de ces objets est du ressort de HGI. Si vous voulez utiliser cette fonction, vous devez être très familier HGIs. Il vaut mieux PHP plutôt que Hyperwave pour accéder aux fichiers distants. Le support de base de données y est bien meilleur.

Voir aussi **hw_getremote()**.

hw_GetSrcByDestObj (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Retourne les ancrages qui pointent sur un objet.

```
array hw_getsrcbydestobj (resource connection, resource objectID)
```

hw_getsrcbydestobj() retourne les attributs de tous les ancrages qui pointent sur *objectID*. L'objet peut être un document ou un autre ancrage, de type destination.

Voir aussi **hw_getanchors()**.

hw_GetObject (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Attributs d'un objet.

```
array hw_getobject (resource connection, [int|array] objectID, string query)
```

hw_getobject() retourne les attributs de l'objet dont l'identifiant est *objectID*, si le second paramètre est un entier. Si le second paramètre est un tableau, la fonction retournera un tableau d'attributs. Dans ce cas, le dernier paramètre est aussi évalué.

query a la syntaxe suivante :

<expression> ::= "(" <expression> ")" |

"!" <expression> | /* NOT */

<expression> "||" <expression> | /* OR */

<expression> "&&" <expression> | /* AND */

<attribute> <operator> <value>

<attribute> ::= /* * n'importe quel attribut (Title, Author, DocumentType ...) */

<operator> ::= "=" | /* égal */

"<" | /* moins que (comparaison de type chaîne) */

">" | /* plus que (comparaison de type chaîne) */

"~" /* recherche par expression régulière */

query permet de sélectionner une nouvelle fois certains objets dans la liste des objets donnés. Contrairement aux autres requêtes, celle ci peut utiliser des attributs non indexés. Le nombre d'attributs retourné dépend de la requête de la requête, et des autorisations d'accès aux objets.

Voir aussi **hw_getandlock()**, **hw_getobjectbyquery()**.

hw_GetAndLock (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Retourne les attributs, et verrouille l'objet.

```
string hw_getandlock (resource connection, resource objectID)
```

hw_getandlock() retourne les attributs, et verrouille l'objet *objectID*. Le verrouillage empêchera les autres utilisateurs d'y accéder, jusqu'à ce qu'il soit deverrouillé.

Voir aussi **hw_unlock()**, **hw_getobject()**.

hw_GetText (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Retourne un document texte.

```
int hw_gettext (resource connection, resource objectID, mixed [rootID/prefix])
```

hw_gettext() retourne le document de l'objet *objectID*. Si le document possède des ancrages qui peuvent être insérés, ils seront déjà insérés. L'option rootID/prefix peut être une chaîne ou un entier. Si c'est un entier, il détermine la méthode d'insertion des liens dans le document. Par défaut, il vaut 0 et les liens seront construits en fonction du nom de l'objet cible. Cela sert beaucoup dans les applications web. Si un lien pointe sur un objet avec le nom 'film_internet' le lien HTML sera <A HREF="/internet_movie">. La position réelle de la source et de la cible dans la hiérarchie seront ignorés. Vous devrez modificer votre site web pour qu'il réécrive les URL, comme par exemple '/mon_script.php3/film_internet'. 'mon_script.php3' devra analyser $PATH_INFO et savoir recherche le document '/mon_script.php3/film_internet'. Si vous ne voulez pas de ce comportement, vous pouvez affecter à rootID/prefix n'importe quel prefixe. Dans ce cas, ce sera une chaîne.

Si *rootID/prefix* est un entier différent de 0 le lien sera construit avec tous les noms de la hiérarchie, en commencant à l'objet d'identifiant rootID/prefix, et séparé par des slash. Si, par exemple, le document 'film_internet' est situé à 'a-b-c-internet_movie' et '-' qui sert de séparateur hiérarchique de niveau sur le serveur Hyperwave et le document source est situé dans 'a-b-d-source' alors, le lien HTML serait: <A HREF="../c/internet_movie">. Cela est très pratique si vous voulez télécharger tout le contenu d'un serveur sur un disque, et faire une carte du système sur votre disque.

hw_gettext() n'est opérationnelle qu'avec des documents de pur texte. Elle n'ouvrira pas de canal spécial de transfert, et ainsi, bloquera le script le temps du transfert.

Voir aussi **hw_pipedocument()**, **hw_free_document()**, **hw_documentbodytag()**, **hw_documentsize()**, **hw_outputdocument()**.

hw_GetObjectByQuery (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Recherche un objet.

```
array hw_getobjectbyquery (resource connection, string query, int max_hits)
```

hw_getobjectbyquery() recherche un objet sur tout le serveur et retourne un tableau d' object ids. Le nombre maximum d'objet est limité par *max_hits*. Si *max_hits* vaut -1 il n'y a pas de limite.

La requête ne fonctionnera qu'avec des attributs indexés.

Voir aussi **hw_getobjectbyqueryobj()**.

hw_GetObjectByQueryObj (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Recherche un objet.

```
array hw_getobjectbyqueryobj (resource connection, string query, int max_hits)
```

hw_getobjectbyqueryobj() recherche un objet sur tout le serveur et retourne un tableau d' object records. Le nombre maximum d'objet est limité par *max_hits*. Si *max_hits* vaut -1 il n'y a pas de limite.

La requête ne fonctionnera qu'avec des attributs indexés.

Voir aussi **hw_getobjectbyquery()**.

hw_GetObjectByQueryColl (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Recherche un objet dans un groupe.

```
array hw_getobjectbyquerycoll (resource connection, resource objectID, string query, int
max_hits)
```

hw_getobjectbyquerycoll() recherche un objet sur tout le groupe objectID et retourne un tableau d' object records. Le nombre maximum d'objet est limité par *objectID*. Si *objectID* vaut -1 il n'y a pas de limite.

La requête ne fonctionnera qu'avec des attributs indexés.

Voir aussi **hw_getobjectbyquerycollobj()**.

hw_GetObjectByQueryCollObj (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Recherche un objet dans un groupe.

```
array hw_getobjectbyquerycollobj (resource connection, resource objectID, string query,
int max_hits)
```

hw_getobjectbyquerycollobj() recherche un objet sur tout le groupe objectID et retourne un tableau d' object records. Le nombre maximum d'objet est limité par *objectID*. Si *objectID* vaut -1 il n'y a pas de limite.

La requête ne fonctionnera qu'avec des attributs indexés.

Voir aussi **hw_getobjectbyquerycoll()**.

hw_GetChildDocColl (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

ids des documents fils d'un groupe.

```
array hw_getchilddoccoll (resource connection, resource objectID)
```

hw_getchilddoccoll() retourne un tableau avec les id des documents fils d'une collection.

Voir aussi **hw_getchildcoll()**.

hw_GetChildDocCollObj (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Attributs des documents fils d'un groupe.

```
array hw_getchilddoccollobj (resource connection, resource objectID)
```

hw_getchilddoccollobj() retourne un tableau contenant les attributs des documents fils du groupe *objectID*.

Voir aussi **hw_childrenobj()**, **hw_getchildcollobj()**.

hw_GetAnchors (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Identifiants des ancrages d'un document.

```
array hw_getanchors (resource connection, resource objectID)
```

hw_getanchors() retourne un tableau contenant les identifiants des ancrages du document *objectID*.

hw_GetAnchorsObj (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Attributs des ancrages d'un document.

```
array hw_getanchorsobj (resource connection, resource objectID)
```

hw_getanchorsobj() retourne un tableau d'attributs des ancrages du document *objectID*.

hw_Mv (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Déplace un objet.

```
int hw_mv (resource connection, array object id array, int source id, int destination id)
```

hw_mv() déplaces les objets dont les identifiants sont passés dans le tableau *source id*, depuis le *source id* dans le *destination id*. Si destination id vaut 0, les objets ne seront plus insérés dans le groupe (ni dans le serveur). Dans ce cas, si une instance était la dernière instance d'un objet, l'objet sera effacé. Si vous voulez effacer toutes les instances d'un coup, utilisez **hw_deleteobject()**.

La valeur retournée est le nombre d'objet déplacés.

Voir aussi **hw_cp()**, **hw_deleteobject()**.

hw_Identify (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Identifie un utilisateur.

```
int hw_identify (string username, string password)
```

hw_identify() identifie un utilisateur, dont le nom d'utilisateur est *username* et le mot de passe *password*. L'identification n'est valide que pour la session en cours. Je ne pense pas que cette fonction serve souvent. Dans la plus part des cas, il est plus simple de s'identifier lors de l'ouverture de la connexion.

Voir aussi **hw_connect()**.

hw_InCollections (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Vérifie qu'un identifiant d'objet est dans un groupe.

```
array hw_incollections (resource connection, array object_id_array, array
collection_id_array, int return_collections)
```

hw_incollections() vérifie qu'un ensemble d'objets (documents ou groupes) donnés par *object_id_array* fait partie des groupe listés par *object_id_array*. Lorsque le quatrième paramètre *return_collections* vaut 0, le sous ensemble d'identifiants qui font partie d'un groupe (i.e. les documents ou groupes qui sont fils d'un ou plusieurs groupe, ou leurs fils, récurusivement) est retourné sous la forme d'un tableau. Cette option permet de mettre en valeur la partie de l'arborescence qui contient le résultat d'une requête, dans un sens graphique.

hw_Info (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Informations à propos d'une connexion.

```
string hw_info (resource connection)
```

hw_info() retourne les informations de la connexion courante. La chaîne retournée a le format suivant : <Serverstring>, <Host>, <Port>, <Username>, <Port of Client>, <Byte swapping>

hw_InsColl (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Insère un groupe.

```
int hw_inscoll (resource connection, resource objectID, array object_array)
```

hw_inscoll() insère un nouveau groupe, avec les attributs *object_array* dans le groupe *objectID*.

hw_InsDoc (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Insère un document.

```
int hw_insdoc (resource connection, int parentID, string object_record, string text)
```

hw_insdoc() insère un nouveau document avec les attributs *object_record*, dans le groupe *parentID*. Cette fonction insère soit un objet avec ses seuls attributs, soit pur objet ascii, avec *text* si il est fourni. Si vous voulez insérer un document de type général, utilisez plutôt **hw_insertdocument()**.

Voir aussi **hw_insertdocument()**, **hw_inscoll()**.

hw_InsertDocument (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Insère un document dans un groupe.

```
int hw_insertdocument (resource connection, int parent_id, int hw_document)
```

hw_insertdocument() insère un document dans le groupe *parent_id*. Le document doit avoir été créé auparavant avec **hw_new_document()**. Assurez vous que les attributs du nouveau document contiennent au moins les attributs suivants :

Type, DocumentType, Title et Name. Vous aurez aussi parfois besoin de MimeType. La fonction retourne l'identifiant de l'objet inséré, ou bien FALSE.

Voir aussi **hw_pipedocument()**.

hw_InsertObject (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Insère un object record.

```
int hw_insertobject (resource connection, string object rec, string parameter)
```

hw_insertobject() insère un objet dans le serveur. L'objet peut être n'importe quel objet Hyperwave valide. Reportez vous à la documentation HG-CSP pour plus de détails sur les paramètres.

Note: Si vous voulez insérer un ancre, l'attribut Position doit être mis à la valeur start/end (début ou fin) ou encore 'invisible'. Les positions invisibles sont nécessaire si l'annotation n'a pas de liens correspondant dans le texte de l'annotation.

Voir aussi **hw_pipedocument()**, **hw_insertdocument()**, **hw_insdoc()** et **hw_inscoll()**.

hw_mapid (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Représente un id globale en un id virtuel local.

```
int hw_mapid (resource connection, int server id, int object id)
```

hw_mapid() représente l'id d'un objet global de n'importe quel serveur Hyperwave, même si vous ne vous y êtes pas connecté avec **hw_connect()**, avec un id d'objet local virtuel. Cet id d'objet local peut alors être utilisé comme n'importe quel id d'objet : par exemple on peut obtenir l'enregistrement d'objet avec la fonction **hw_getobject()**. L'id du serveur est la première partie de l'id global (GOid) de l'objet, qui est en fait une adresse IP.

Note: Afin d'utiliser cette fonction, vous devez lever le flag F_DISTRIBUTED, ce qui ne peut être fait qu'à la compilation. Par défaut, il n'est pas levé. Lisez les commentaires dans le fichier hg_comm.c

hw_Modifyobject (PHP 3>= 3.0.7, PHP 4)

Modifie les attributs d'objet record.

```
int hw_modifyobject (resource connection, int object_to_change, array remove, array add,
int mode)
```

hw_modifyobject() permet d'effacer, d'ajouter ou de modifier les attributs d'un objet. L'objet est reperé par son identifiant *object_to_change*. Le premier tableau, *remove*, est la liste des attributs à effacer. Le deuxième tableau *add* est la liste des attributs à ajouter. Afin de modifier un attribut, il vous faudra dont l'effacer, puis l'ajouter à nouveau. **hw_modifyobject()** effacera toujours les attributs avant de les ajouter, à moins que la valeur de l'attribut à effacer ne soit pas une chaîne, ou un tableau.

Le dernier paramètre détermine si la modification est récursive ou pas. 1 signigie que la modification est récursive. Si un objet ne peut pas être modifié, il sera ignoré. **hw_error()** n'indiquera alors pas toujours d'erreur, même si certains objets n'ont pas pu être modifié.

Les clés des deux tableaux sont les noms des attributs. La valeurs de chaque élément peut être un tableau, ou une chaîne ou n'importe quoi d'autre. Dans le cas du tableau, la valeur de l'attribut est construite en séparant chaque élément par un point virgule. Dans le cas de la chaîne, elle sert directement de valeur. Une chaîne vide provoquera un effacement de l'attribut. Dans le cas oú la valeur n'est ni un tableau, ni une chaîne, aucune opération ne sera effectuée. Cela est nécessaire si vous voulez ajouter un attribut complètement une nouvelle valeur pour un attribut existant. Si le tableau d'effacement contenait

une chaîne vide comme attribut, le serveur tenterait d'effacer l'attribut, ce qui échouerai de toute manière, car cet attribut n'existe pas. L'ajout de cet attribut échouerai aussi. Affecter la valeur de 0 à cet attribut ne l'effacerait pas, et l'ajout fonctionnerait.

Si vous voulez changer l'attribut 'Nom' de valeur courante 'livres' en 'articles' vous devrez faire deux tableaux, et appeler **hw_modifyobject()**.

Exemple 1. Modification d'un attribut

```
<?php
       // $connect est une connexion valide
       // $objid est l'identifiant de l'objet
       $remarr = array("Name" => "books");
       $addarr = array("Name" => "articles");
       $hw_modifyobject($connect, $objid, $remarr, $addarr);
?>
```

Afin d'effacer/ajouter une paire nom=valeur aux attributs d'un objet, utilisez simplement les tableaux d'effacement et d'ajout, et laissez le dernier/troisième paramètre vide. Si l'attribut est le premier de ce nom à ajouter, donnez une valeur entière à cet élément.

Exemple 2. Ajouter un nouvel attribut

```
<?php
       // $connect st une connexion Hyperwave valide
       // $objid est l'identifiant de l'objet à modifier
       $remarr = array("Name" => 0);
       $addarr = array("Name" => "articles");
       $hw_modifyobject($connect, $objid, $remarr, $addarr);
?>
```

> **Note :** Les attributs multilingues, (tels que 'Title'), peuvent être modifiés de deux manières : soit en fournissant la valeur de ces attributs de manière native (langue :valeur), soit en fournissant un tableau avec les éléments de chaque langue, comme décrit ci-dessus. L'exemple deviendrai alors :

Exemple 3. Modifier l'attribut de Titre (Title)

```
<?php
       $remarr = array("Title" => "en:Books");
       $addarr = array("Title" => "en:Articles");
       $hw_modifyobject($connect, $objid, $remarr, $addarr);
?>
```

ou

Exemple 4. Modifier l'attribut Title

```
<?php
       $remarr = array("Title" => array("en" => "Books"));
       $addarr = array("Title" => array("en" => "Articles", "ge"=>"Artikel"));
       $hw_modifyobject($connect, $objid, $remarr, $addarr);
?>
```

Pour supprimer l'entrée française 'Livres' et ajouter l'entrée 'Articles' et l'entrée allemande 'Artikel'.

Exemple 5. Suppression d'un attribut

```
<?php
       $remarr = array("Title" => "");
       $addarr = array("Title" => "en:Articles");
       $hw_modifyobject($connect, $objid, $remarr, $addarr);
?>
```

Note : Cet exemple va effacer tous les attributs avec le nom 'Title' et ajouter un nouvel attribut 'Title'. Cela peut être pratique pour effacer des attributs récursivement.

Note : Si vous devez effacer tous les attributs avec un certains nom, vous devez passer une chaîne vide comme valeur.

Note : Seuls les attributs 'Title', 'Description' et 'Keyword' gère correctement le préfixe de langue. Pour les autres attributs qui ne portent pas de préfixe de language, le préfixe 'xx' sera assigné.

Note : L'attribut 'Name' est un peu particuluer. Dans certains cas, il ne peut pas être complétement effacé. Vous aurez alors le message 'Change of base attribute' (l'apparition de cette erreur n'est pas très claire). Ainsi, vous aurez à ajouter une nouvelle entrée pour Name puis, effacer l'ancien.

Note : Il ne faut pas encadrer cette fonction par des appels à **hw_getandlock()** et **hw_unlock()**. **hw_modifyobject()** le fait de manière interne.

Retourne TRUE si aucune erreur ne survient, et FALSE sinon.

hw_New_Document (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Crée un nouveau document.

```
int hw_new_document (string object_record, string document_data, int document_size)
```

hw_new_document() retourne un nouveau document Hyperwave avec comme données *document_data* et comme attributs *object_record*. La longueur de *document_data* doit être données dans *document_size*. Cette fonction n'insère pas l'objet dans le serveur Hyperwave.

Voir aussi **hw_free_document()**, **hw_documentsize()**, **hw_documentbodytag()**, **hw_outputdocument()**, **hw_insertdocument()**.

hw_Objrec2Array (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Convertit les attributs d'un objet en tableau.

```
array hw_objrec2array (string object_record)
```

hw_objrec2array() convertit les attributs *object_record* d'un objet en un tableau. Les clés du tableau seront les noms des attributs. Les attributs multiples comme par exemple 'Title', dans différentes langues, seront rassemblées dans un autre tableau. Une clé est la partie gauche d'un attribut. Actuellement, seuls les attributs 'Title', 'Description' et 'Keyword' sont traités correctement.

Voir aussi **hw_array2objrec()**.

hw_OutputDocument (PHP 3>= 3.0.3, PHP 4 4.0b1 only)

Affiche hw_document.

```
int hw_outputdocument (int hw_document)
```

hw_outputdocument() affiche *hw_document* sans la balise BODY.

hw_pConnect (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Crée une connexion persistante.

```
int hw_pconnect (string host, int port, string username, string password)
```

hw_pconnect() retourne un index de connexion en cas de succès, et FALSE si la connexion n'a pas pu être créée. **hw_pconnect()** ouvre une connexion persistante à un serveur Hyperwave. Tous les arguments doivent être entre guillemets, hormis le numéro de port *port*. Les arguments *username* et *password* sont optionnels, et peuvent être ignorés. Dans ce cas, aucune authentification ne sera faite, (connexion anonyme). Cette fonction retourne un index de connexion qui sera utilisée par les autres fonctions Hyperwave. Vous pouvez ouvrir plusieurs connexions simultanées.

Voir aussi **hw_connect()**.

hw_PipeDocument (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Retourne un document.

```
int hw_pipedocument (resource connection, resource objectID)
```

hw_pipedocument() retourne le document Hyperwave d'objet id *objectID*. Si le document a des ancrages, ils seront insérés. Le document sera transmis via une connexion de données spéciale, qui ne bloque pas la connexion de contrôle (ie, le serveur n'attends pas la fin du transfert pour rendre la main).

Voir aussi **hw_gettext()** (insertions), **hw_free_document()**, **hw_documentsize()**, **hw_documentbodytag()** et **hw_outputdocument()**.

hw_Root (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Object id de la racine.

```
int hw_root (void)
```

hw_root() retourne l' object id de la racine. Actuellement, cette identifiant est toujours 0. L'ensemble des fils de la racine est celui du serveur courant.

hw_Unlock (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Déverrouille un objet.

```
int hw_unlock (resource connection, resource objectID)
```

hw_unlock() déverrouille un document, et laisse l'accès aux autres utilisateurs.

Voir aussi **hw_getandlock()**.

hw_Who (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Liste des utilisateurs actuellement identifiés.

```
int hw_who (resource connection)
```

hw_who() retourne un tableau contenant la liste des utilisateurs actuellement connectés au serveur Hyperwave. Chaque élément du tableau est lui même un tableau, qui contient l'identifiant de l'élément, le nom, le système, la date de connexion (onSinceDate), l'heure de connexion (onSinceTime), la durée de connexion (TotalTime) et self. 'self' contient l'élément est l'utilisateur qui a appelé la fonction.

hw_Username (unknown)

Nom de l'utilisateur actuellement identifié.

```
string hw_getusername (resource connection)
```

hw_getusername() retourne le nom d'utilisateur utilisé par la connexion.

XXXIV. ICAP

Pour faire fonctionner ces fonctions, vous devez compiler PHP avec l'option `--with-icap`. Il vous faudra aussi la librairie icap. Récupérez la dernière version à http://icap.chek.com/ et décompilez-la, puis installez la.

icap_open (PHP 4 >= 4.0b4)

Ouvre une connexion ICAP.

```
resource icap_open (string calendar, string username, string password, string options)
```

icap_open() retourne un flot ICAP en cas de succès, FALSE en cas d'erreur.

icap_open() ouvre une connexion ICAP au serveur de calendrier *calendar*. Si le paramètre *options* est spécifié, passe *options* à la boîte aux lettres(????).

icap_close (unknown)

Ferme le flot ICAP.

```
int icap_close (resource icap_stream [, int flags])
```

icap_close() ferme le flot ICAP *icap_stream*.

icap_fetch_event (PHP 4 >= 4.0b4)

Recherche un événement dans le calendrier.

```
int icap_fetch_event (resource stream_id, int event_id [, int options])
```

icap_fetch_event() recherche un évenement dans le calendrier spécifié par *id*.

Retourne un objet événement dont les attributs sont :

- int id - ID de l'événement.
- int public - TRUE si l'événement est public, FALSE si il est privé.
- string category - Catégorie de l'événement.
- string title - Titre de l'événement.
- string description - Description de l'événement.
- int alarm - Nombre de minutes avant d'envoyer une alerte pour cet événement.
- object start - Objet contenant une date et une heure.
- object end - Objet contenant une date et une heure.

Tous les objets de date et heure sont construits comme suit :

- int year - année
- int month - mois
- int mday - jour du mois
- int hour - heure
- int min - minutes
- int sec - secondes

icap_list_events (PHP 4 >= 4.0RC1)

Retourne une liste d'événement entre deux dates.

```
array icap_list_events (resource stream_id, int begin_date [, int end_date])
```

icap_list_events() retourne un tableau d'identificants d'événéments, compris entre deux dates.
icap_list_events() prend une date de début et une date de fin. Un tableau d'identifiants est retourné.
Tous les objets de date et heure sont construits comme suit :

- int year - année
- int month - mois
- int mday - jour du mois
- int hour - heure
- int min - minutes
- int sec - secondes

icap_store_event (PHP 4 >= 4.0b4)

Enregistre un événement dans un agenda ICAP.

```
string icap_store_event (resource stream_id, object event)
```

icap_store_event() enregistre un événement dans un calendrier ICAP.
Un événement est un objet avec les attributs suivants :

- int id - ID de l'événement.
- int public - `TRUE` si l'événement est public, `FALSE` si il est privé.
- string category - Catégorie de l'événement.
- string title - Titre de l'événement.
- string description - Description de l'événement.
- int alarm - Nombre de minutes avant d'envoyer une alerte pour cet événement.
- object start - Objet contenant une date et une heure.
- object end - Objet contenant une date et une heure.

Tous les objets de date et heure sont construits comme suit :

- int year - année
- int month - mois
- int mday - jour du mois
- int hour - heure
- int min - minutes
- int sec - secondes

Retourne TRUE en cas de succès, et FALSE en cas d'erreur.

icap_delete_event (PHP 4 >= 4.0b4)

Efface un événement dans un agenda ICAP.

```
string icap_delete_event (resource stream_id, int uid)
```

icap_delete_event() efface l'événement d'identifiant *uid*.
Retourne TRUE.

icap_snooze (PHP 4 >= 4.0b4)

Eteind l'alarme d'un événement.

```
string icap_snooze (resource stream_id, int uid)
```

icap_snooze() éteind l'alarme de l'événement identifié par l'UID *uid*.
Retourne TRUE.

icap_list_alarms (PHP 4 >= 4.0b4)

Retourne une liste d'événements qui ont une alarme prévue à une date.

```
int icap_list_alarms (resource stream_id, array date, array time)
```

icap_list_alarms() retourne un tableau d'identifiants, qui ont une alarme de prévue à la date *alarm_date*.
icap_list_alarms() prend une date, et retourne un tableau d'identifiants.
Tous les objets de date et heure sont construits comme suit :

- int year - année
- int month - mois
- int mday - jour du mois
- int hour - heure
- int min - minutes
- int sec - secondes

XXXV. Iconv

Ce module est une interface vers la librairie iconv. Pour pouvoir l'utiliser, vous devez compiler PHP avec l'option `--with-iconv`. Pour cela, vous devez avoir la fonction iconv() dans votre librairie standard C, ou bien la librairie libiconv installée sur votre système. La librairie libiconv est disponible à http://clisp.cons.org/~haible/packages-libiconv.html.

L'extension iconv convertit des fichiers entre divers jeux de caractères. Les jeux supportés dépendent de l'implémentation de iconv() sur votre système. Notez que cette fonction ne fonctionne pas toujours bien sur tous les systèmes. Dans ce cas, vous devez installer la librairie tout de même.

iconv (PHP 4 >= 4.0.5)

Convertit une chaîne dans un jeu de caractères

```
string iconv (string in_charset, string out_charset, string str)
```

iconv() convertit la chaîne *string* depuis le jeu de caractères *in_charset* vers le jeu de caractères *out_charset*. Elle retourne la chaîne ainsi convertie, ou bien FALSE, en cas d'échec.

Exemple 1. Exemple avec iconv()

```
<?php
  echo iconv("ISO-8859-1","UTF8","Ceci est un test.");
?>
```

iconv_get_encoding (PHP 4 >= 4.0.5)

Lit le jeu de caractères courant

```
array iconv_get_encoding ([string type])
```

iconv_get_encoding() retourne le paramètrage courant du gestionnaire **ob_iconv_handler()**, sous la forme d'un tableau, ou bien FALSE, en cas d'échec.

Voir aussi **iconv_set_encoding()** et **ob_iconv_handler()**.

iconv_set_encoding (PHP 4 >= 4.0.5)

Modifie le jeu courant de caractères courant

```
array iconv_set_encoding (string type, string charset)
```

iconv_set_encoding() modifie le jeu de caractères courant, et remplace la valeur courante du paramétre *type* par *charset* et TRUE en cas de succès et FALSE, en cas d'échec.

Exemple 1. Exemple iconv_set_encoding()

```
<?php
  iconv_set_encoding("internal_encoding", "UTF-8");
  iconv_set_encoding("output_encoding", "ISO-8859-1");
?>
```

Voir aussi **iconv_get_encoding()** et **ob_iconv_handler()**.

ob_iconv_handler (PHP 4 >= 4.0.5)

Gestionnaire de sortie pour maitriser le jeu de caractères de sortie

```
array ob_iconv_handler (string contents, int status)
```

ob_iconv_handler() convertit la chaîne utilisant le jeu de caractères *`internal_encoding`* en une chaîne utilisant le jeu de caractères *`output_encoding`*.

`internal_encoding` et *`output_encoding`* doivent être définis par **iconv_set_encoding()** ou dans le fichier de configuration.

Exemple 1. Exemple avec ob_iconv_handler()

```
<?php
  ob_start("ob_iconv_handler"); // start output buffering
?>
```

Voir aussi **iconv_get_encoding()** et **iconv_set_encoding()**.

XXXVI. Images

Vous pouvez utiliser les fonctions PHP pour obtenir les tailles des images aux formats JPEG, GIF, PNG et SWF, et si vous avez la librairie GD (disponible à http://www.boutell.com/gd/) vous pourrez aussi créer et manipuler ces images.

Les formats des images que vous pourrez manipuler dépendent de la version de GD que vous installerez, et de toute autre librairie dont GD a besoin pour traiter à ces images. Les versions antérieures à la version 1.6 supportent le GIF, mais pas le PNG. Pour les versions plus récentes, c'est le contraire.

Pour accéder aux images en JPEG, vous devez installer la librairie jpeg-6b (disponible à ftp://ftp.uu.net/graphics/jpeg/), puis, recompiler GD pour qu'elle utilise jpeg-6b. Vous devrez aussi compiler PHP avec `--with-jpeg-dir=/path/to/jpeg-6b`.

Pour ajouter le support des polices Type 1, vous devez installer t1lib (disponible à ftp://sunsite.unc.edu/pub/Linux/libs/graphics/), puis ajouter l'option `--with-t1lib[=dir]`.

getimagesize (PHP 3, PHP 4 >= 4.0b1)

Retourne la taille d'une image GIF, JPG ou PNG.

```
array getimagesize (string filename [, array imageinfo])
```

getimagesize() va déterminer la taille des images de type GIF, JPG, PNG ou SWF et en retourner les dimensions avec le type d'image, et une chaîne type "height/width", à placer dans une balise HTML ou `IMG` normale.

getimagesize() retourne un tableau de 4 éléments. L'index 0 contient la largeur. L'index 1 contient la longueur. L'index 2 contient le type de l'image : 1 = GIF, 2 = JPG, 3 = PNG. L'index 3 contient la chaîne à placer dans les balises HTML : "height=xxx width=xxx".

Exemple 1. getimagesize()

```
<?php
  $size = getimagesize("img/flag.jpg");
?>
<IMG SRC="img/flag.jpg"
<?php
  echo $size[3];
?»
```

Exemple 2. getimagesize() avec une URL

```
<?php
  $size = getimagesize("http://www.php.net/gifs/logo.gif");
?>
```

Avec les images JPEG, deux en-têtes supplémentaires sont retournés : `channel` et `bits`. `channel` vaudra 3 avec les images RGB, et 4 avec les images CMYK. `bits` est le nombre de bits de chaque couleur.

Si l'accès à *`filename`* est impossible, ou si ce n'est pas une image valide, **getimagesize()** retournera `NULL` et générera une alerte.

Le paramètre optionnel *`imageinfo`* permet d'extraire des informations supplémentaires du fichier image. Actuellement, cette option va retourner différents marqueurs JPG APP dans un tableau associatif. Certains programmes utilisent ces marqueur APP pour préciser les informations dans les balises HTML. Un marqueur commun est le marqueur APP13, décrit à http://www.iptc.org/. Vous pouvez utiliser la fonction **iptcparse()** pour analyser ce marqueur, et obtenir des informations intelligibles.

Exemple 3. getimagesize() qui retourne IPTC

```
<?php
  $size = getimagesize("testimg.jpg",&$info);
  if (isset($info["APP13"])) {
    $iptc = iptcparse($info["APP13"]);
    var_dump($iptc);
  }
?>
```

Note : getimagesize() ne requiert pas la bibliothèque GD.

Note : Le support URL a été ajouté en PHP 4.0.5.

ImageAlphaBlending (PHP 4 >= 4.0.6)

Modifie le mode de blending d'une image

```
int imagealphablending (resource im, boolean blendmode)
```

imagealphablending() fournit deux modes de dessins des images en vraies couleurs (truecolors). En mode "blending", le canal alpha de chaque couleur est fournie à chaque fonction de dessin, tel que **imagesetpixel()** peut déterminer sa transparence. GD va alors automatiquement mixer la couleur à ce point, et stocker le résultat dans l'image. Le pixel résultant est alors opaque. En mode non-mixant, la couleur est copiée littéralement avec ses informations de canal alpha, et remplace le pixel de destination. Le mixage n'est pas disponible avec les images à palette. Si *blendmode* vaut TRUE, alors le mode de mixage sera activé, sinon il sera désactivé.

Note : imagealphablending() a été ajoutée en PHP 4.0.6 et nécessite GD 2.0.1.

ImageArc (PHP 3, PHP 4 >= 4.0b1)

Dessine une ellipse partielle.

```
int imagearc (resource im, int cx, int cy, int w, int h, int s, int e, int col)
```

imagearc() dessine une ellipse partielle, centrée sur *cx*, *cy* (le coin en haut à gauche est l'origine (0,0)) dans l'image référencée par *im*. *w* et *h* spécifient la largeur et la hauteur de l'ellipse, tandis que le début et la fin de l'arc sont donnés en degrés, par les arguments *s* et *e*.

imagefilledarc (PHP 4 >= 4.0.6)

Dessine une ellipse partielle et la remplit

```
int imagefilledarc (int im, int cx, int cy, int w, int h, int s, int e, int col, int style)
```

imagefilledarc() dessine une ellipse partielle, centrée sur le point (*cx*, *cy*). Le coin supérieur gauche est (0, 0), dans l'image *im*. *w* et *h* spécifient respectivement la largeur et la hauteur de l'ellipse, tandis que les points de début et de fin sont représentés par *s* et *e*, en degrés. L'argument *style* est un champ de bits, combiné avec l'opérateur OR :

1. IMG_ARC_PIE
2. IMG_ARC_CHORD
3. IMG_ARC_NOFILL
4. IMG_ARC_EDGED

IMG_ARC_PIE et IMG_ARC_CHORD sont mutellemement exclusives; IMG_ARC_CHORD ne fait que connecter les angles de début et de fin avec une ligne droite, tandis que IMG_ARC_PIE produit une ligne courbe. IMG_ARC_NOFILL indique que l'arc (ou corde) doit être dessiné mais pas rempli. IMG_ARC_EDGED, utilisé conjointement avec IMG_ARC_NOFILL, indique que les angles de début et de fin doivent être connectés au centre. Cette fonction est recommandée pour faire les graphiques de type camenbert.

Note : imagefilledarc() a été ajoutée en PHP 4.0.6 et nécessite GD 2.0.1.

ImageEllipse (PHP 4 >= 4.0.6)

Dessine une ellipse

```
int imageellipse (resource im, int cx, int cy, int w, int h, int col)
```

imageellipse() dessine une ellipse centrée sur le point (*cx*, *cy*). Le coin supérieur gauche est aux coordonnées (0,0). L'image de dessin est *im*. *w* et *h* spécifient respectivement la largeur et la hauteur de l'ellipse. La couleur de dessin de l'ellipse est *color*.

Note : imageellipse() a été ajoutée en PHP 4.0.6 et nécessite GD 2.0.1.

ImageFilledEllipse (PHP 4 >= 4.0.6)

Dessine une ellipse pleine

```
int imagefilledellipse (resource im, int cx, int cy, int w, int h, int col)
```

imagefilledellipse() dessine une ellipse centrée sur le point (*cx*, *cy*). Le coin supérieur gauche est aux coordonnées (0,0). L'image de dessin est *im*. *w* et *h* spécifient respectivement la largeur et la hauteur de l'ellipse. La couleur de remplissage de l'ellipse est *color*.

Note : imagefilledellipse() a été ajoutée en PHP 4.0.6 et nécessite GD 2.0.1.

ImageChar (PHP 3, PHP 4 >= 4.0b1)

Dessine un caractère horizontalement.

```
int imagechar (resource im, resource font, int x, int y, string c, int col)
```

imagechar() dessine le premier caractère de la chaîne *c* dans l'image *id* avec le coin supérieur gauche placé à la position *x*,*y* (le coin en haut à gauche est l'origine (0,0)) avec la couleur *col*. Si la police est 1, 2, 3, 4 ou 5, une police intégrée sera utilisée (plus le chiffre est grand, plus grande est la police).

Voir aussi **imageloadfont()**.

ImageCharUp (PHP 3, PHP 4 >= 4.0b1)

Dessine un caractère verticalement.

```
int imagecharup (resource im, resource font, int x, int y, string c, int col)
```

imagecharup() dessine le premier caractère de la chaîne *c* dans l'image *id* avec le coin supérieur gauche placé à la position (*x*,*y*) (le coin en haut à gauche est l'origine (0,0)), avec la couleur *col*. Si la police est 1, 2, 3, 4 ou 5, une police intégrée sera utilisée (plus le chiffre est grand, plus grande est la police).

Voir aussi **imageloadfont()**.

ImageColorAllocate (PHP 3, PHP 4 >= 4.0b1)

Alloue une couleur pour une image.

```
int imagecolorallocate (resource im, int red, int green, int blue)
```

imagecolorallocate() retourne un identifiant de couleur, représentant la couleur composée avec les couleurs RGB (*red*, *green*, *blue*). L'argument *im* est le résultat de la fonction **imagecreate()**. **imagecolorallocate()** doit être appelée pour créer chaque couleur qui sera représentée par *im*.

```
<?php
  $white = imagecolorallocate($im, 255,255,255);
  $black = imagecolorallocate($im, 0,0,0);
?>
```

ImageColorDeAllocate (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Désalloue une couleur pour une image

```
int imagecolordeallocate  (resource im, int index)
```

imagecolordeallocate() désalloue une couleur précédemment allouée avec la fonction **imagecolorallocate()**.

```
<?php
  $white = imagecolorallocate($im, 255, 255, 255);
  imagecolordeallocate($im, $white);
?>
```

ImageColorAt (PHP 3, PHP 4 >= 4.0b1)

Retourne l'index de la couleur d'un pixel donné.

```
int imagecolorat (resource im, int x, int y)
```

imagecolorat() retourne l'index de la couleur du pixel situé aux coordonnées (*x*, *y*), dans l'image *im*.

Voir aussi **imagecolorset()** et **imagecolorsforindex()**.

ImageColorClosestAlpha (PHP 4 >= 4.0.6)

Retourne la couleur la plus proche, en tenant compte du canal alpha

```
int imagecolorclosestalpha (resource im, int red, int green, int blue, int alpha)
```

imagecolorclosestalpha() retourne l'index de la couleur, dans la palette de l'image *im*, la plus proche de la couleur spécifiée par les autres paramètres, au format RGB et de canal alpha *alpha*.

Voir aussi **imagecolorexactalpha()**.

> **Note : imagecolorclosestalpha()** a été ajoutée en PHP 4.0.6 et nécessite GD 2.0.1.

ImageColorClosest (PHP 3, PHP 4 >= 4.0b1)

Retourne l'index de la couleur la plus proche d'une couleur donnée.

```
int imagecolorclosest (resource im, int red, int green, int blue)
```

imagecolorclosest() retourne l'index de la couleur de la palette qui est la plus proche de la valeur RGB passée.

La "distance" entre la couleur souhaitée et les couleurs de la palette est calculée en considérant l'espace RGB comme un espace à 3 dimensions.

Voir aussi **imagecolorexact()**.

ImageColorExact (PHP 3, PHP 4 >= 4.0b1)

Retourne l'index de la couleur donnée.

```
int imagecolorexact (resource im, int red, int green, int blue)
```

imagecolorexact() retourne l'index de la couleur spécifiée dans la palette de l'image *im*.

Si la couleur n'existe pas dans cette palette, **imagecolorexact()** retourne -1.

Voir aussi **imagecolorclosest()**.

ImageColorExactAlpha (PHP 4 >= 4.0.6)

Retourne l'index d'une couleur avec son canal alpha

```
int imagecolorexactalpha (resource im, int red, int green, int blue, int alpha)
```

imagecolorexactalpha() retourne l'index de la couleur fournie au format RGB et son canal alpha *alpha*, dans l'image *im*.

Si la couleur n'existe pas dans la palette de l'image, **imagecolorexactalpha()** retourne -1.

Voir aussi **imagecolorclosestalpha()**.

> **Note : imagecolorexactalpha()** a été ajoutée en PHP 4.0.6 et nécessite GD 2.0.1.

ImageColorResolve (PHP 3>= 3.0.2, PHP 4 >= 4.0b1)

Retourne l'index de la couleur donnée, ou la plus proche possible.

```
int imagecolorresolve (resource im, int red, int green, int blue)
```

imagecolorresolve() retourne un index de couleur à tous les coups. Soit il arrive à trouver la couleur demandée dans la palette, soit il recherche la couleur la plus proche.

Voir aussi **imagecolorclosest()**.

ImageColorResolveAlpha (PHP 4 >= 4.0.6)

Retourne un index de couleur ou son alternative la plus proche, y compris le canal alpha

```
int imagecolorresolvealpha (resource im, int red, int green, int blue, int alpha)
```

imagecolorresolvealpha() retourne toujours un index de couleur, disponible dans la palette de l'image *im* : soit c'est la couleur exacte, soit c'est la meilleure approximation.

Voir aussi **imagecolorclosestalpha()**.

Note : imagecolorresolvealpha() a été ajoutée en PHP 4.0.6 et nécessite GD 2.0.1.

ImageGammaCorrect (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Applique une correction gamma à l'image

```
int imagegammacorrect (resource im, double inputgamma, double outputgamma)
```

imagegammacorrect() applique une correction gamma à l'image GD *im*. Le facteur d'entrée est *inputgamma*, et le facteur de sortie est *outputgamma*.

ImageColorSet (PHP 3, PHP 4 >= 4.0b1)

Change la couleur dans une palette à l'index donné.

```
boolean imagecolorset (resource im, int index, int red, int green, int blue)
```

imagecolorset() permet d'attribuer à un index d'une palette une couleur spécifique. C'est une fonction très pratique pour effectuer du remplissage de couleur sans le faire réellement.

Voir aussi **imagecolorat()**.

ImageColorsForIndex (PHP 3, PHP 4 >= 4.0b1)

Retourne la couleur associée à un index.

```
array imagecolorsforindex (resource im, int index)
```

imagecolorsforindex() retourne un tableau associatif avec les couleurs rouge (red) , vert (green), bleu (blue) qui contiennent les valeurs de la couleur correspondante.

Voir aussi **imagecolorat()** et **imagecolorexact()**.

ImageColorsTotal (PHP 3, PHP 4 >= 4.0b1)

Calcule le nombre de couleurs d'une palette.

```
int imagecolorstotal (resource im)
```

imagecolorstotal() retourne le nombre de couleurs de la palette.

Voir aussi **imagecolorat()** et **imagecolorsforindex()**.

ImageColorTransparent (PHP 3, PHP 4 >= 4.0b1)

Définit la couleur transparente.

```
int imagecolortransparent (resource im [, int col])
```

imagecolortransparent() permet de choisir la couleur transparente d'une image, et de lui donner la valeur de col. *im* est un identifiant d'image, retourné par **imagecreate()** et *col* est un identifiant de couleur retourné par **imagecolorallocate()**.

L'identifiant de la nouvelle (ou courante) couleur transparente est retourné.

ImageCopy (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Copie une partie d'une image

```
int imagecopy  (resource dst_im, resource src_im, int dst_x, int dst_y, int src_x, int
src_y, int src_w, int src_h)
```

Copie une partie de l'image *src_im* sur l'image de destination *dst_im*, en commençant aux coordonnées *src_x*, *src_y* et sur la largeur de *src_w* et la hauteur de *src_h*. La portion ainsi définie sera copiée et placée aux coordonnées *dst_x* et *dst_y*.

ImageCopyMerge (PHP 4 >= 4.0.1)

Copie et fusionne une partie d'une image

```
int imagecopymerge (resource dst_im, resource src_im, int dst_x, int dst_y, int src_x, int
src_y, int src_w, int src_h, int pct)
```

imagecopymerge() copie une partie de l'image *src_im* dans l'image de destination *dst_im* en commençant aux coordonnées (*src_x*, *src_y*), avec la largeur *src_w* et la hauteur *src_h*. La zone de l'image ainsi définie sera copiée aux coordonnées (*dst_x*, *dst_y*), dans l'image de destination. Les deux images seront fusionnées suivant le paramètre *pct*, qui peut valoir de 0 à 100. Si *pct* = 0, aucune action n'est faite, alors que si *pct* = 100, **imagecopymerge()** se comporte exactement comme **imagecopy()**.

> **Note : imagecopymerge()** a été ajoutée en PHP 4.0.6.

ImageCopyMergeGray (PHP 4 >= 4.0.6)

Copie et fusionne une partie d'une image en niveaux de gris

```
int imagecopymergegray  (resource dst_im, resource src_im, int dst_x, int dst_y, int
src_x, int src_y, int src_w, int src_h, int pct)
```

imagecopymergegray() copie une partie de l'image *src_im* dans l'image de destination *dst_im* commençant aux coordonnées (*src_x*, *src_y*), avec la largeur *src_w* et la hauteur *src_h*. La zone de l'image ainsi définie sera copiée aux coordonnées (*dst_x*, *dst_y*), dans l'image de destination. Les deux images seront fusionnées suivant le paramètre

pct, qui peut valoir de 0 à 100. Si *pct* = 0, aucune action n'est faite, alors que si *pct* = 100, **imagecopymerge()** se comporte exactement comme **imagecopy()**.

imagecopymergegray() est identique à la fonction **imagecopymerge()**, hormis le fait que lors de la fusion, le "hue" de l'image sera conservé grâce à la conversion de la zone dans l'image de destination en gris, avant l'opération de copie.

Note : imagecopymergegray() a été ajoutée en PHP 4.0.6.

ImageCopyResized (PHP 3, PHP 4 >= 4.0b1)

Copie et redimensionne une partie d'une image.

```
int imagecopyresized (resource dst_im, resource src_im, int dstX, int dstY, int srcX, int
srcY, int dstW, int dstH, int srcW, int srcH)
```

imagecopyresized() copie une partie rectangulaire d'une image dans une autre image de destination. *dst_im* est l'image de destination, *src_im* est l'image source. Si les dimensions de la source et de la destination ne sont pas égales, un étirement adéquat est effectué pour faire correspondre les deux. Les coordonnées fournies sont définies par rapport au coin supérieur gauche. Cette fonction peut être utilisée pour recopier des régions à l'intérieur d'une même image, si *dst_im* et *src_im* sont identiques : mais si les régions se chevauchent, le résultat risque d'être incohérent.

Voir aussi **imagecopyresampled()**.

ImageCopyResampled (PHP 4 >= 4.0.6)

Copie, redimensionne, rééchantillonne une image

```
int imagecopyresampled (resource dst_im, resource src_im, int dstX, int dstY, int srcX,
int srcY, int dstW, int dstH, int srcW, int srcH)
```

imagecopyresampled() copie une zone rectangulaire de l'image *src_im* vers l'image *src_im*. Durant la copie, la zone est rééchantillonnée de manière à conserver la clarté de l'image durant une réduction. *Dst_im* est l'image de destination, *src_im* est l'image source. Si les hauteurs et largeur des source et destination diffèrent, l'image copiée sera étirée de manière appropriée. Les coordonnées sont celles du coin supérieur gauche. **imagecopyresampled()** peut servir à copier des zones d'une image vers elle-même, mais si les régions se chevauchent, les résultats sont imprévisibles.

Voir aussi **imagecopyresized()**.

Note : imagecopyresampled() a été ajoutée en PHP 4.0.6 et nécessite GD 2.0.1.

ImageCreate (PHP 3, PHP 4 >= 4.0b1)

Crée une nouvelle image à palette.

```
resource imagecreate (int x_size, int y_size)
```

imagecreate() retourne un identifiant d'image représentant une image vide, de largeur *x_size* et longueur *y_size*.

imagecreatefromgif (PHP 3, PHP 4 >= 4.0b1)

Crée une nouvelle image à partir d'un fichier ou d'une URL.

```
resource imagecreatefromgif (string filename)
```

imagecreatefromgif() retourne un identifiant d'image qui représente l'image obtenue à partir du fichier dont le nom est donné.

imagecreatefromgif() retourne une chaîne vide en cas d'échec. Il va aussi retourner une erreur qui va afficher un lien brisé dans un navigateur. Pour simplifier le débuggage, utilisez le code suivant, qui retourne une erreur GIF :

Exemple 1. Exemple de gestion des erreurs durant la création d'image (gracieusement offert par vic@zymsys.com)

```
<?php
function loadgif($imgname){
  $im = @imagecreatefromgif($imgname); /* Tentative d'ouverture */
  if ($im == "") { /* échec ? */
    $im = ImageCreate(150,30); /* Crée une image vide */
    $bgc = ImageColorAllocate($im,255,255,255);
    $tc  = ImageColorAllocate($im,0,0,0);
    imagefilledrectangle($im,0,0,150,30,$bgc);
    imagestring($im,1,5,5,"Erreur lors du chargement du fichier $imgname",$tc);
    /* Affiche un message d'erreur */
  }
  return $im;
}
?>
```

> **Note :** Etant donné que toutes les fonctions de gestion des GIF ont été supprimées de la bibliothèque GD version 1.6, cette fonction n'est pas disponible si vous utilisez cette version de la librairie.

ImageCreateTrueColor (PHP 4 >= 4.0.6)

Crée une nouvelle image en vraies couleurs

```
resource imagecreatetruecolor (int x_size, int y_size)
```

imagecreatetruecolor() retourne une ressource représentant une image noire de largeur *x_size*, et de hauteur *y_size*.

> **Note : imagecreatetruecolor()** a été ajoutée en PHP 4.0.6 et nécessite GD 2.0.1.

ImageTrueColorToPalette (PHP 4 >= 4.0.6)

Convertit une image en vraies couleurs en image à palette

```
void imagetruecolortopalette (resource im, boolean dither, int ncolors)
```

imagetruecolortopalette() convertit l'image en vraies couleurs *im* en image à palette. Le code de cette fonction est directement tiré de la librairie du "Independent JPEG Group", qui est tout simplement génial. Le code a été modifié pour préserver l'essentiel du canal alph dans la nouvelle palette, en plus de conserver les couleurs du mieux possible. Mais cela

ne fonctionne pas toujours comme voulu. Il est alors préférable de générer un résultat en vraies couleurs, ce qui a toujours le meilleur rendu.

Si *dither* vaut TRUE, cela indique que l'image doit être ditherée : l'image sera un peu plus granuleuse, mais l'approximation des couleurs sera meilleure.

ncolors est le nombre maximal de couleurs dans la palette finale.

> **Note : imagetruecolortopalette()** a été ajoutée en PHP 4.0.6 et nécessite GD 2.0.1.

ImageCreateFromJPEG (PHP 3>= 3.0.16, PHP 4 >= 4.0RC1)

Crée une nouvelle image JPEG à partir d'un fichier ou d'une URL

```
resource imagecreatefromjpeg (string filename)
```

imagecreatefromjpeg() retourne un identifiant d'image représentant une image obtenue à partir du fichier *filename*.

imagecreatefromjpeg() retourne une chaîne vide en cas d'échec. Elle affiche aussi un message d'erreur, qui s'affiche comme un lien brisé dans un navigateur web. Pour faciliter le débuggage, voici une erreur JPEG:

Exemple 1. Exemple de gestion d'erreur lors de la création d'image (gracieusement offert par vic@zymsys.com)

```
<?php
function loadjpeg($imgname) {
  $im = @imagecreatefromjpeg($imgname); /* Tentative d'ouverture */
  if (!$im) { /* Vérification */
    $im = imagecreate(150, 30); /* Création d'une image blanche */
    $bgc = imagecolorallocate($im, 255, 255, 255);
    $tc  = imagecolorallocate($im, 0, 0, 0);
    imagefilledrectangle($im, 0, 0, 150, 30, $bgc);
// Affichage d'un message d'erreur
    imagestring($im, 1, 5, 5, "Erreur de chargement de l'image $imgname", $tc);
  }
  return $im;
}
?>
```

ImageCreateFromPNG (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Crée une nouvelle image PNG à partir d'un fichier ou d'une URL

```
resource imagecreatefrompng (string filename)
```

imagecreatefrompng() retourne un identifiant d'image représentant une image obtenue à partir du fichier *filename*.

imagecreatefromjpeg() retourne une chaîne vide en cas d'échec. Elle affiche aussi un message d'erreur, qui s'affiche comme un lien brisé dans un navigateur web. Pour faciliter le débuggage, voici une erreur PNG:

Exemple 1. Exemple de gestion d'erreur lors de la création d'image (gracieusement offert par vic@zymsys.com)

```
<?php
function LoadPNG($imgname) {
  $im = @imagecreatefrompng($imgname); /* Tentative d'ouverture */
  if (!$im) { /* Vérification */
    $im = imagecreate(150, 30); /* Création d'une image blanche */
        $bgc = imagecolorallocate($im, 255, 255, 255);
        $tc  = imagecolorallocate($im, 0, 0, 0);
        imagefilledrectangle($im, 0, 0, 150, 30, $bgc);
        /* Affichage d'un message d'erreur */
        imagestring($im, 1, 5, 5, "Erreur de chargement de l'image $imgname", $tc);
    }
    return $im;
}
?>
```

ImageCreateFromWBMP (PHP 4 >= 4.0.1)

Crée une image depuis un fichier WBMP

```
resource imagecreatefromwbmp (string filename)
```

imagecreatefromwbmp() retourne une ressource d'image PHP, représentant l'image *filename*.

imagecreatefromwbmp() retourne une chaîne vide en cas d'erreur. Il retourne aussi un message d'erreur qui s'affiche comme un lien mort dans un navigateur. Pour aider au débuggage, l'exemple suivant va produire une erreur WBMP:

Exemple 1. Exemple de gestion des erreurs durant la création d'une image WBMP (gracieusement proposé par vic@zymsys.com)

```
function loadwbmp($imgname) {
  $im = @imagecreatefromwbmp($imgname); /* Tentative d'ouverture */
  if (!$im) { /* Vérification que cela s'est bien passé */
    $im  = imagecreate(20, 20); /* Crée une image blanche */
    $bgc = imagecolorallocate($im, 255, 255, 255);
    $tc  = imagecolorallocate($im, 0, 0, 0);
    imagefilledrectangle($im, 0, 0, 10, 10, $bgc);
// Affiche le message d'erreur
    imagestring($im, 1, 5, 5, "Erreur de chargement de $imgname", $tc);
  }
  return $im;
}
```

ImageCreateFromString (PHP 4 >= 4.0.4)

Crée une image à partir d'une chaîne

```
resource imagecreatefromstring (string string)
```

imagecreatefromstring() retourne un identifiant d'image représentant la chaîne *string*.

ImageDashedLine (PHP 3, PHP 4 >= 4.0b1)

Dessine une ligne pointillée.

```
int imagedashedline (resource im, int x1, int y1, int x2, int y2, int col)
```

imagedashedline() dessine une ligne pointillée entre les points (*x1*,*y1*) et (*x2*,*y2*) (le coin supérieur droit est l'origine (0,0)) dans l'image *im*, avec la couleur *col*.

Voir aussi **imageline()**.

ImageDestroy (PHP 3, PHP 4 >= 4.0b1)

détruit une image.

```
int imagedestroy (resource im)
```

imagedestroy() libère toute la mémoire associée à l'image *im*. *im* est un identifiant d'image valide retourné par **imagecreate()**.

ImageFill (PHP 3, PHP 4 >= 4.0b1)

Remplit.

```
int imagefill (resource im, int x, int y, int col)
```

imagefill() effectue un remplissage avec la couleur *col*, dans l'image *im*, à partir du point de coordonnées (*x*, *y*) (le coin supérieur gauche est l'origine (0,0)).

ImageFilledPolygon (PHP 3, PHP 4 >= 4.0b1)

Dessine un polygone rempli.

```
int imagefilledpolygon (resource im, array points, int num_points, int col)
```

imagefilledpolygon() dessine un polygone rempli dans l'image *im*. *points* est un tableau PHP qui contient les sommets des polygones sous la forme :. points[0] = x0, points[1] = y0, points[2] = x1, points[3] = y1, etc. *num_points* est le nombre total de sommets.

ImageFilledRectangle (PHP 3, PHP 4 >= 4.0b1)

Dessine un rectangle rempli.

```
int imagefilledrectangle (resource im, int x1, int y1, int x2, int y2, int col)
```

imagefilledrectangle() dessine un rectangle de couleur *col* dans l'image *im*, en commençant par le sommet supérieur gauche (*x1*, *y1*) et finissant au sommet inférieur droit (*x2*, *y2*). Le coin supérieur gauche est l'origine (0, 0).

ImageFillToBorder (PHP 3, PHP 4 >= 4.0b1)

Remplit avec une région avec une couleur spécifique.

```
int imagefilltoborder (resource im, int x, int y, int border, int col)
```

imagefilltoborder() remplit avec la couleur *col* toute la région à l'intérieur de la région limitée par la couleur *border*. Le point de départ est (*x*,*y*) (le coin supérieur gauche est l'origine (0,0)).

ImageFontHeight (PHP 3, PHP 4 >= 4.0b1)

Retourne la hauteur de la police.

```
int imagefontheight (resource font)
```

imagefontheight() retourne la hauteur de la police *font* en pixels.

Voir aussi **imagefontwidth()** et **imageloadfont()**.

ImageFontWidth (PHP 3, PHP 4 >= 4.0b1)

Retourne la largeur de la police.

```
int imagefontwidth (resource font)
```

imagefontwidth() retourne la largeur de la police *font* en pixels.

Voir aussi **imagefontheight()** et **imageloadfont()**.

ImageGif (PHP 3, PHP 4 >= 4.0b1)

Envoie une image GIF vers un navigateur ou un fichier.

```
int imagegif (resource im, string filename)
```

imagegif() crée un fichier image GIF avec le nom *filename* d'après l'image *im*. L'argument *im* est un identifiant valide retourné par la fonction **imagecreate()**.

Le format de l'image sera GIF87a, à moins que l'image n'ait une couleur transparente (mise en place grâce à la fonction **imagecolortransparent()**)), ce qui fera qu'elle sera au format GIF89a.

Le nom du fichier est optionnel, et dans ce cas, l'image sera transmise directement à la sortie standard. En envoyant une en-tête de type `image/gifcontent-type` , (grâce à la fonction **header()**), vous pouvez créer des images avec des scripts PHP.

> **Note :** Etant donné que toutes les fonctions GIF ont été supprimées de la bibliothèque GD version 1.6, cette fonction ne sera pas accessible si vous avez cette version de la librairie.
>
> Le code suivant vous permet d'écrire des scripts PHP plus portables : le type de GD est automatiquement détecté. Il remplace la séquence `Header("Content-type: image/gif"); ImageGif($im);` par un code plus souple :

```
<?php
  if (function_exists("imagegif")) {
    header("Content-type: image/gif");
    imagegif($im);
  }
  elseif (function_exists("imageJpeg")) {
```

```
    header("Content-type: image/jpeg");
    imagejpeg($im, "", 0.5);
  }
  elseif (function_exists("imagePng")) {
    header("Content-type: image/png");
    imagepng($im);
  } elseif (function_exists("imagewbmp")) {
    header("Content-type: image/vnd.wap.wbmp");
    imagewbmp($im);
  }  else {
    die("Pas de support graphique avec PHP sur ce serveur");
  }
?>
```

Note : En PHP 4, à partir de la version 4.0.2, vous pouvez utiliser la fonction **imagetypes()** à la place de **function_exists()** pour vérifier que certains formats d'images sont supportés :

```
<?php
  if (function_exists("imagegif")) {
    header("Content-type: image/gif");
    imagegif($im);
  }
  elseif (function_exists("imageJpeg")) {
    header("Content-type: image/jpeg");
    imagejpeg($im, "", 0.5);
  }
  elseif (function_exists("imagePng")) {
    header("Content-type: image/png");
    imagepng($im);
  } elseif (function_exists("imagewbmp")) {
    header("Content-type: image/vnd.wap.wbmp");
    imagewbmp($im);
  } else {
    die("Pas de support graphique avec PHP sur ce serveur");
  }
?>
```

Voir aussi **imagepng()**, **imagewbmp()**, **imagejpeg()**, **imagetypes()**.

ImagePNG (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Envoie une image PNG vers un navigateur ou un fichier.

```
int imagepng (resource im [, string filename])
```

imagepng() envoie l'image GD (*im*) au format PNG sur la sortie standard (typiquement, le navigateur web), ou si *filename* est fourni, l'envoie dans un fichier.

```
<?php
  $im = imagecreatefrompng("test.png");
  imagepng($im);
?>
```

Le nom du fichier est optionnel, et dans ce cas, l'image sera transmise directement à la sortie standard. En envoyant une image de type `image/png content-type` (grâce à la fonction **header()**), vous pouvez créer des images PNG avec des scripts PHP.

Voir aussi **imagegif()**, **imagewbmp()**, **imagejpeg()**, **imagetypes()**.

ImageJPEG (PHP 3>= 3.0.16, PHP 4 >= 4.0RC1)

Envoie une image JPEG vers un navigateur ou un fichier.

```
int imagejpeg (resource im [, string filename [, int quality]])
```

imagejpeg() envoie l'image GD (*`im`*) au format JPEG sur la sortie standard (typiquement, le navigateur web), ou si *`filename`* est fourni, l'envoie dans un fichier. *`im`* a été créé par **imagecreate()**.

Le nom du fichier est optionnel, et dans ce cas, l'image sera transmise directement à la sortie standard. En envoyant une image de type `image/jpeg content-type` (grâce à la fonction **header()**), vous pouvez créer des images JPEG avec des scripts PHP.

> **Note :** Le support JPEG n'est disponible que si PHP est compilé avec GD-1.8 ou plus récent.

`quality` est optionnel, et prend des valeurs entières de 0 (pire qualité, petit fichier) et 100 (meilleure qualité, gros fichier). Par défaut, la valeur est à 100.

Voir aussi **imagepng()**, **imagewbmp()**, **imagegif()** et **imagetypes()**.

ImageWBMP (PHP 3>= 3.0.15, PHP 4 >= 4.0.1)

Affiche une image WBMP

```
int imagewbmp (resource im [, string filename])
```

imagewbmp() crée l'image WBMP dans le fichier *`filename`*, à partir de l'image *`im`*. Le paramètre *`im`* a été créé avec la fonction **imagecreate()**.

`filename` est optionnel, et s'il est omis, l'image sera envoyée directement au client. En plaçant l'en-tête image/vnd.wap.wbmp, dans le champs "content-type", vous pourrez afficher une image WBMP.

> **Note :** Le support WBMP n'est disponible que si PHP a été compilé avec GD-1.8 ou plus récent.

Voir aussi **imagepng()**, **imagegif()**, **imagejpeg()** et **imagetypes()**.

ImageInterlace (PHP 3, PHP 4 >= 4.0b1)

Active ou désactive l'entrelacement.

```
int imageinterlace (resource im [, int interlace])
```

imageinterlace() active ou désactive le bit d'entrelacement. Si l'entrelacement est à 1, l'image *`im`* sera interlacée, et sinon, elle ne le sera pas.

imageinterlace() retourne l'état courant d'entrelacement de l'image.

ImageLine (PHP 3, PHP 4 >= 4.0b1)

Dessine une ligne.

```
int imageline (resource im, int x1, int y1, int x2, int y2, int col)
```

imageline() dessine une ligne depuis le point (*x1*,*y1*) jusqu'au point (*x2*,*y2*) (le coin supérieur gauche est l'origine (0,0)) dans l'image *im* et avec la couleur *col*.

Voir aussi **imagecreate()** et **imagecolorallocate()**.

ImageLoadFont (PHP 3, PHP 4 >= 4.0b1)

Charge une nouvelle police.

```
resource imageloadfont (string file)
```

imageloadfont() charge une nouvelle police utilisateur et retourne un identifiant sur cette police. Cet identifiant sera toujours supérieur à 5, pour éviter les conflits avec les polices standard PHP.

Le format des polices dépend actuellement du système d'exploitation. Ce qui signifie qu'il vous faut générer des fichiers de polices pour la machine qui fait tourner PHP.

Tableau 1. Format de fichier de police.

Position	Type de données C	Description
Octets 0-3	int	Nombre de caractères de la police
Octets 4-7	int	Valeur du premier caractère de la police (souvent 32 pour espace)
Octets 8-11	int	Largeur en pixels des caractères
Octets 12-15	int	Hauteur en pixels des caractères
Octets 16-	char	Tableau avec les données des caractères, un octet par pixel pour chaque caractère, avec un total de (nombre_caractères*largeur*hauteur) octets.

Voir aussi **imagefontwidth()** et **imagefontheight()**.

ImagePolygon (PHP 3, PHP 4 >= 4.0b1)

Dessine un polygone.

```
int imagepolygon (resource im, array points, int num_points, int col)
```

imagepolygon() dessine un polygone dans l'image *im*. *points* est un tableau PHP qui contient les sommets du polygone sous la forme : points[0] = x0, points[1] = y0, points[2] = x1, points[3] = y1, etc. num_points est le nombre de sommets.

Voir aussi **imagecreate()**.

ImagePSBBox (PHP 3>= 3.0.9, PHP 4 >= 4.0RC1)

Retourne le rectangle entourant un texte et dessiné avec une police PostScript Type1.

```
array imagepsbbox (string text, resource font, int size, int space, int width, float angle)
```

size est exprimé en pixels.

space permet de changer la valeur par défaut du charactère espace. Cette valeur est ajoutée lors des dessins, et donc, peut être négative.

tightness permet de contrôler la quantité d'espace entre les caractères. Cette quantité est ajoutée lors des dessins, et peut donc être négative.

angle est en degrés.

Les paramètres *space* et *tightness* sont exprimés en unité d'espacement de caractères, avec 1 unité vaut 1/1000 d'un em carré (NDT : kesako?).

Les paramètres *space*, *tightness* et *angle* sont optionnels.

Le rectangle entourant est calculé en utilisant les informations disponibles sur les tailles de caractères, et, malheureusement, il a tendance à être légèrement différent du résultat réel final. Si l'angle est de 0 degré, vous pouvez-vous attendre à avoir besoin d'un rectangle d'au moins un pixel plus grand dans toutes les directions.

imagepsbbox() retourne un tableau contenant les éléments suivants :

0	Abscisse inférieure gauche
1	Ordonnée inférieure gauche
2	Abscisse supérieure droite
3	Ordonnée supérieure droite

Voir aussi **imagepstext()**.

ImagePSEncodeFont (PHP 3>= 3.0.9, PHP 4 >= 4.0RC1)

Change le codage vectoriel d'un caractère dans une police.

```
int imagepsencodefont (resource font, string encodingfile)
```

imagepsencodefont() charge le codage vectoriel d'un caractère depuis un fichier et change le codage vectoriel de la police correspondante. Etant donné que les polices PostScript ne disposent pas des caractères au-delà de 127, vous aurez sÛrement besoin de les changer si vous utilisez une autre langue que l'anglais. Le format exact est décrit dans la documentation T1libs. T1lib est disponible en deux formes : IsoLatin1.enc et IsoLatin2.enc.

Si vous commencez à utiliser cette fonction régulièrement, une meilleure solution est de définir un encodage, et de l'utiliser avec set ps.default_encoding dans le fichier de configuration pour utiliser par défaut l'encodage correct.

ImagePSFreeFont (PHP 3>= 3.0.9, PHP 4 >= 4.0RC1)

Libère la mémoire occupée par une police PostScript Type 1.

```
void imagepsfreefont (resource font)
```

Voir aussi **imagepsloadfont()**.

ImagePSLoadFont (PHP 3>= 3.0.9, PHP 4 >= 4.0RC1)

Charge une police PostScript Type 1 depuis un fichier.

```
resource imagepsloadfont (string filename)
```

Au cas où tout a bien marché, un index de police va être retourné, et pourra être utilisé pour des opérations ultérieures. Sinon, la fonction retourne FALSE et affiche un message décrivant ce qui est erroné.

```
<?php
  header("Content-type: image/jpeg");
  $im =    imagecreate(350, 45);
  $noir =  imagecolorallocate($im, 0, 0, 0);
  $Blanc = imagecolorallocate($im, 255, 255, 255);
  $font =  imagepsloadfont("bchbi.pfb");
  imagepstext($im, "Test ... Ca marche!", $font, 32, $white, $black, 32, 32);
  imagepsfreefont($font);
  imagejpeg($im, "");
  imagedestroy ($im);
?<
```

Voir aussi **imagepsfreefont()**.

ImagePsExtendFont (PHP 3>= 3.0.9, PHP 4 >= 4.0RC1)

Etend ou condense une police de caractères

```
boolean imagepsextendfont (resource font, double extend)
```

imagepsextendfont() étend ou condense la police de caractères *font*. Si la valeur de *extend* est inférieure à 1, ce sera une condensation.

ImagePsSlantFont (PHP 3>= 3.0.9, PHP 4 >= 4.0RC1)

Incline une police de caractères

```
boolean imagepsslantfont (resource font, double slant)
```

imagepsslantfont() met en italique la police de caractères *font* avec le coefficient *slant*.

ImagePSText (PHP 3>= 3.0.9, PHP 4 >= 4.0RC1)

Dessine un texte sur une image avec une police PostScript Type1.

```
array imagepstext (resource im, string text, resource font, int size, int foreground, int
background, int x, int y, int [space], int [tightness], float [angle], int
[antialias_steps])
```

size est exprimé en pixels.

foreground est la couleur dans laquelle le texte va être dessiné. *background* est la couleur d'anti aliasing. Aucun pixel avec la couleur *background* n'est dessiné, ce qui fait que l'arrière-plan n'a pas besoin d'être dans une couleur fixe.

Les coordonnées données (*x*, *y*) définissent l'origine du premier caractère (grossièrement, le coin inférieur gauche du caractère). Ceci est différent de la fonction **imagestring()**, où (x, y) définissait le coin supérieur gauche du premier caractère. Reportez-vous à la documentation PostScript pour avoir des détails à propos des polices et de leurs tailles.

space permet de changer la taille par défaut du caractère d'espacement. Cette valeur peut être négative.

tightness permet de contrôler la quantité d'espace entre deux caractères. Cette valeur peut être négative.

angle est en degrés.

antialias_steps permet de contrôler le nombre de couleurs du texte anti-aliasé. Les valeurs autorisées sont 4 et 16. 16 est recommandé pour les polices de moins de 20 pixels, car l'effet est alors visible. Avec les tailles plus grandes, utilisez de préférence 4, qui est moins gourmande en ressources.

Les paramètres *space* et *tightness* sont exprimés en unité d'espaces caractère, ce qui vaut 1/1000ème d'un em-carré (? ? ?).

Les paramètres *space*, *tightness*, *angle* et *antialias* sont optionnels.

imagepstext() retourne un tableau contenant les éléments suivants :

0	Abscisse inférieure gauche
1	Ordonnée inférieure gauche
2	Abscisse supérieure droite
3	Ordonnée supérieure droite

Voir aussi **imagepsbbox()**.

ImageRectangle (PHP 3, PHP 4 >= 4.0b1)

Dessine un rectangle.

```
int imagerectangle (resource im, int x1, int y1, int x2, int y2, int col)
```

imagerectangle() dessine un rectangle dans la couleur *col*, dans l'image *im*, et en commençant au point supérieur gauche (*x1*, *y1*), et en finissant au point inférieur droit (*x2*,*y2*). Le coin supérieur gauche est l'origine (0,0).

ImageSetPixel (PHP 3, PHP 4 >= 4.0b1)

Dessine un pixel.

```
int imagesetpixel (resource im, int x, int y, int col)
```

imagesetpixel() dessine un pixel au point (*x*,*y*) (le coin supérieur gauche est l'origine (0,0)) dans l'image *im*, et avec la couleur *col*.

Voir aussi **imagecreate()** et **imagecolorallocate()**.

imagesetbrush (PHP 4 >= 4.0.6)

Modifie la brosse pour le dessin des lignes

```
int imagesetbrush (resource im, resource brush)
```

imagesetbrush() remplace la brosse courante pour le dessin des lignes par *brush*. Cette brosse sera alors utilisée avec les fonctions **imageline()** et **imagepolygon()**.

> **Note :** Vous n'avez rien à faire lorsque vous en avez terminé avec une brosse, mais si vous détruisez l'image de brosse, vous ne DEVEZ plus utiliser les options IMG_COLOR_BRUSHED et IMG_COLOR_STYLEDBRUSHED des fonctions **imageline()** et **imagepolygon()**, avant d'avoir créé une nouvelle brosse.

> **Note : imagesetbrush()** a été ajoutée en PHP 4.0.6.

ImageSetTile (PHP 4 >= 4.0.6)

Modifie l'image utilisée pour le carrelage

```
int imagesettile (resource im, resource tile)
```

imagesettile() remplace l'image de carrelage courante par l'image *tile*, à utiliser dans tous les remplissages (comme avec les fonctions **imagefill()** et **imagefilledpolygon()**) lors des remplissages avec l'option `IMG_COLOR_TILED`.

Une image de carrelage est une image utilisée pour remplir une zone, de manière répétitive. N'importe quelle image GD peut servir d'image de remplissage. L'utilisation de la couleur transparente (gérée avec la fonction **imagecolortransparent()**) permet à certaines zones d'apparaître à travers le carrelage.

> **Note :** Vous n'avez rien à faire lorsque vous en avez terminé avec une brosse, mais si vous détruisez l'image de brosse, vous ne DEVEZ plus utiliser l'option `IMG_COLOR_TILED` des fonctions **imagefill()** et **imagefilledpolygon()**, avant d'avoir créé une nouvelle brosse.

> **Note : imagesettile()** a été ajoutée en PHP 4.0.6.

ImageSetThickness (PHP 4 >= 4.0.6)

Modifie l'épaisseur d'un trait

```
void imagesetthickness (resource im, int thickness)
```

imagesetthickness() modifie l'épaisseur du trait des lignes de l'image *im*. Cette épaisseur intervient dans les dessins de polygones, ellipses, cercles, rectangles, etc... *thickness* est en pixels.

> **Note : imagesetthickness()** a été ajoutée en PHP 4.0.6 et nécessite GD 2.0.1.

ImageString (PHP 3, PHP 4 >= 4.0b1)

Dessine une chaîne horizontale.

```
int imagestring (resource im, int font, int x, int y, string s, int col)
```

imagestring() dessine une la chaîne sur une ligne horizontale, dans l'image *im*, aux coordonnées (*x*,*y*) (le coin supérieur gauche est l'origine (0,0)) dans la couleur *col*. Si l'argument de police vaut 1, 2, 3, 4 ou 5, une des polices par défaut sera utilisée).

Voir aussi **imageloadfont()**.

ImageStringUp (PHP 3, PHP 4 >= 4.0b1)

Dessine une chaîne verticale.

```
int imagestringup (resource im, int font, int x, int y, string s, int col)
```

imagestringup() dessine une chaîne sur une ligne verticale dans l'image *im* aux coordonnées (*x*, *y*) (l'origine est le coin supérieur gauche (0,0)) dans la couleur *col*. Si la police utilisée est 1, 2, 3, 4 ou 5, une police par défaut sera utilisée.

Voir aussi **imageloadfont()**.

ImageSX (PHP 3, PHP 4 >= 4.0b1)

Retourne la largeur d'une image.

```
int imagesx (resource im)
```

imagesx() retourne la largeur de l'image référencée par *im*.

Voir aussi **imagecreate()** et **imagesy()**.

ImageSY (PHP 3, PHP 4 >= 4.0b1)

Retourne la hauteur de l'image.

```
int imagesy (resource im)
```

imagesy() retourne la hauteur de l'image référencée par *im*.

Voir aussi **imagecreate()** et **imagesx()**.

ImageTTFBBox (PHP 3>= 3.0.1, PHP 4 >= 4.0b1)

Retourne le rectangle entourant un texte et dessiné avec une police TrueType.

```
array imagettfbbox (int size, int angle, string fontfile, string text)
```

imagettfbbox() calcule et retourne le rectangle entourant le texte *text*, écrit avec une police truetype.

text

La chaîne à mesurer.

size

La taille de la police en pixel.

fontfile

Le nom de la police TrueType (peut aussi être une URL.)

angle

Angle en degré dans lequel le texte *text* va être mesuré.

imagettfbbox() retourne une tableau avec 8 éléments, représentant les 4 sommets du rectangle ainsi défini.

0	Coin inférieur gauche, abscisse
1	Coin inférieur gauche, ordonnée
2	Coin inférieur droit, abscisse
3	Coin inférieur droit, ordonnée
4	Coin supérieur droit, abscisse
5	Coin supérieur droit, ordonnée
6	Coin supérieur gauche, abscisse
7	Coin supérieur gauche, ordonnée

Les positions des points sont relatives au texte *text*, indépendamment de l'angle : coin supérieur gauche faire référence au coin supérieur gauche du texte écrit horizontalement.

imagettfbbox() requiert les bibliothèques GD et Freetype.

Voir aussi **imagettftext()**.

ImageTTFText (PHP 3, PHP 4 >= 4.0b1)

Dessine un texte avec une police TrueType.

```
array imagettftext (resource im, int size, int angle, int x, int y, int col, string
fontfile, string text)
```

imagettftext() dessine la chaîne *text* dans l'image *im*, en commençant aux coordonnées (*x*,*y*) (le coin supérieur gauche est l'origine (0,0)), avec un angle de *angle*, et dans la couleur *col*, en utilisant la police TrueType identifiée par *fontfile*.

Les coordonnées (*x*,*y*) serviront de référence pour le premier caractère (en gros, le coin inférieur gauche du caractère). C'est différent de **imagestring()**, qui utilise le coin supérieur droit.

angle est donné en degrés, avec degré 0 pour un texte horizontal, et en comptant les angles dans le sens inverse des aiguilles d'une montre (sens direct).

fontfile est le chemin jusqu'à la police TrueType à utiliser.

text est le texte à dessiner, incluant aussi des séquences de caractères UTF-8 (de la forme: {) pour générer des caractères au-delà de 255.

col est l'index de la couleur dans la palette. Utiliser des index négatifs, revient à supprimer l'anti-aliasing.

imagettftext() retourne un tableau de 8 éléments représentant les 4 points marquants les limites du texte. L'ordre des points est :supérieur gauche, supérieur droit, inférieur droit, inférieur gauche. Les points sont nommés relativement au texte à l'horizontale**imagecolorexact()**.

Cet exemple va générer une image GIF noire de 400x30 pixels, avec les mots "Test en cours...Oméga: Ω" en police blanche, type Arial.

Exemple 1. Exemple avec imagettftext()

```
<?php
  header("Content-type: image/gif");
  $im = imagecreate(400,30);
  $black = imagecolorallocate($im, 0,0,0);
  $white = imagecolorallocate($im, 255,255,255);
  imagettftext($im, 20, 0, 10, 20, $white, "/path/arial.ttf",
  "Test en cours... Oméga: &#937;");
  imagegif($im);
```

```
  imagedestroy($im);
?>
```

imagettftext() requiert les bibliothèques GD ainsi que FreeType (http://www.freetype.org/).

Voir aussi **imagettfbbox()**.

ImageTypes (PHP 3 CVS only, PHP 4 >= 4.0.2)

Retourne les types d'images supportés par la version courante de PHP

```
int imagetypes (void)
```

imagetypes() retourne un champs de bits correspondant aux formats d'images supportés par la version de GD utilisée. Les valeurs suivantes sont valables : `IMG_GIF` | `IMG_JPG` | `IMG_PNG` | `IMG_WBMP`. Pour vous assurer du support PNG, faites ceci :

Exemple 1. Exemple avec ImageTypes

```
<?php
  if (imagetypes() & IMG_PNG) {
    echo "Le type PNG est supporté";
  }
?>
```

read_exif_data (PHP 4 >= 4.0.1)

Lit les en-têtes EXIF d'une image JPEG

```
array read_exif_data (string filename)
```

read_exif_data() lit les en-têtes EXIF de l'image JPEG nommée *`filename`*. Elle retourne un tableau associatif où les index sont les noms d'en-têtes EXIF, et les valeurs sont leur valeur associée. Les en-têtes EXIF sont souvent disponibles dans les images générées par les appareils photos numériques, mais chaque constructeur marque ses images d'une manière qui lui est propre : il est impossible de savoir quelles en-têtes seront présents.

Exemple 1. Exemple avec read_exif_data()

```
<?php
  $exif = read_exif_data('p0001807.jpg');
  while(list($k,$v)=each($exif)) {
    echo "$k: $v<br>\n";
  }
?>
```

Cet exemple va afficher :

```
FileName: p0001807.jpg
FileDateTime: 929353056
FileSize: 378599
CameraMake: Eastman Kodak Company
CameraModel: KODAK DC265 ZOOM DIGITAL CAMERA (V01.00)
DateTime: 1999:06:14 01:37:36
```

```
Height: 1024
Width: 1536
IsColor: 1
FlashUsed: 0
FocalLength:  8.0mm
RawFocalLength: 8
ExposureTime:  0.004 s (1/250)
RawExposureTime: 0.0040000001899898
ApertureFNumber: f/ 9.5
RawApertureFNumber: 9.5100002288818
FocusDistance: 16.66m
RawFocusDistance: 16.659999847412
Orientation: 1
ExifVersion: 0200
```

Note : read_exif_data() n'est disponible que sous PHP 4 , compilé avec --enable-exif.

read_exif_data() ne requiert pas la librairie GD.

XXXVII. IMAP

Pour avoir accès à ces fonctions, vous devez compiler PHP avec l'option `--with-imap`. Il faut avoir installé la librairie C-client. Chargez sa dernière version sur le serveur ftp://ftp.cac.washington.edu/imap/ et compilez la. Puis, copiez le fichier `c-client/c-client.a` dans `/usr/local/lib` ou n'importe quel autre dossier qui soit dans le chemin de link. Enfin, copiez les fichiers `c-client/rfc822.h`, `mail.h` et `linkage.h` dans `/usr/local/include` ou n'importe quel autre dossier qui soit dans le chemin d'inclusion.

Ces fonctions ne sont pas limitées au protocole IMAP, malgré leur nom. La librairie sur laquelle elles sont développées supporte aussi NNTP, POP3 et les méthodes d'accès aux boîtes aux lettres locales. Reportez-vous à la fonction **imap_open()** pour plus d'informations.

Ce document ne peut entrer dans les détails de toutes les sujets abordés. Plus d'informations sont disponibles avec la documentation de la librairie C (`docs/internal.txt`) ainsi que les RFC suivantes :

- RFC821 (http://www.faqs.org/rfcs/rfc821.html): Simple Mail Transfer Protocol (SMTP).
- RFC822 (http://www.faqs.org/rfcs/rfc822.html): Standard for ARPA internet text messages.
- RFC2060 (http://www.faqs.org/rfcs/rfc2060.html): Internet Message Access Protocol (IMAP) Version 4rev1.
- RFC1939 (http://www.faqs.org/rfcs/rfc1939.html): Post Office Protocol Version 3 (POP3).
- RFC977 (http://www.faqs.org/rfcs/rfc977.html): Network News Transfer Protocol (NNTP).
- RFC2076 (http://www.faqs.org/rfcs/rfc2076.html): Common Internet Message Headers.
- RFC2045 (http://www.faqs.org/rfcs/rfc2045.html) , RFC2046 (http://www.faqs.org/rfcs/rfc2046.html) , RFC2047 (http://www.faqs.org/rfcs/rfc2047.html) , RFC2048 (http://www.faqs.org/rfcs/rfc2048.html) & RFC2049 (http://www.faqs.org/rfcs/rfc2049.html): Multipurpose Internet Mail Extensions (MIME).

Une étude approfondie est aussi disponibles dans les livres suivants (en anglais): Programming Internet Email (http://www.oreilly.com/catalog/progintemail/noframes.html) par David Wood et Managing IMAP (http://www.oreilly.com/catalog/mimap/noframes.html) par Dianna Mullet & Kevin Mullet.

imap_8bit (PHP 3, PHP 4 >= 4.0b1)

Convertit une chaîne à 8 bits en une chaîne à guillemets.

```
string imap_8bit (string string)
```

imap_8bit() convertit la chaîne à 8 bits en une chaîne à guillemets.

imap_8bit() retourne une chaîne à guillemets.

Voir aussi **imap_qprint()**.

imap_alerts (PHP 3>= 3.0.12, PHP 4 >= 4.0b4)

Retourne toutes les alertes

```
array imap_alerts (void)
```

imap_alerts() retourne tous les messages d'alerte IMAP générés depuis le dernier appel à **imap_alerts()** ou depuis le début de la page. Lorsque **imap_alerts()** est appelé, la pile d'alertes est vidée.

imap_append (PHP 3, PHP 4 >= 4.0b1)

Ajoute une chaîne dans une boîte aux lettres.

```
int imap_append (resource imap_stream, string mbox, string message, stringflags)
```

imap_append() ajoute un message dans la boîte aux lettres *mbox*. Si l'option *flags* est utilisée, *flags* sera aussi écrit dans la boîte aux lettres.

imap_append() retourne TRUE en cas de succès, et FALSE en cas d'erreur.

Lors des échanges avec le serveur Cyrus IMAP, vous devrez utiliser "\r\n" comme terminaison de ligne, à la place de "\n" ou l'opération échouera.

Exemple 1. Exemple avec imap_append()

```
<?php
$stream = imap_open("{your.imap.host}INBOX.Drafts","username", "password");
$check = imap_check($stream);
print "Nombre de message avant ajout : ". $check->Nmsgs."\n";
imap_append($stream,"{your.imap.host}INBOX.Drafts"
                  ,"From: me@my.host\r\n"
                  ."To: you@your.host\r\n"
                  ."Subject: test\r\n"
                  ."\r\n"
                  ."Ceci est un message de test. Ignorez le\r\n"
                  );
$check = imap_check($stream);
print "Nombre de message après ajout : ". $check->Nmsgs."\n";
imap_close($stream);
?>
```

imap_base64 (PHP 3, PHP 4 >= 4.0b1)

Décode un texte encodé en BASE64.

```
string imap_base64 (string text)
```

imap_base64() décode un texte encodé en BASE64. Le texte décodé est retourné sous la forme d'une chaîne.

imap_binary (PHP 3>= 3.0.2, PHP 4 >= 4.0b1)

Convertit une chaîne à 8 bits en une chaîne à base64.

```
string imap_binary (string string)
```

imap_binary() convertit la chaîne à 8 bits *string* en une chaîne à base64.

imap_binary() retourne la chaîne codée.

Voir aussi **imap_base64()**.

imap_body (PHP 3, PHP 4 >= 4.0b1)

Lit le corps d'un message.

```
string imap_body (resource imap_stream, int msg_number, int flags)
```

imap_body() retourne le corps du message numéro *msg_number* de la boîte aux lettres courante. L'option *flags* est un masque qui peut contenir les valeurs suivantes :

- FT_UID - msgno est un UID
- FT_PEEK - Ne pas lever le drapeaux \Seen (Message lu) s'il n'est pas déjà levé.
- FT_INTERNAL - La chaîne renvoyée est au format interne, et ne va pas canoniser les CRLF.

imap_body() va retourner une copie brute du corps du message. Pour extraire les sous parties MIME du message, utilisez **imap_fetchstructure()** pour analyser la structure, et **imap_fetchbody()** pour extraire une copie d'une des sous-partie.

imap_check (PHP 3, PHP 4 >= 4.0b1)

Vérifie le courrier de la boîte aux lettres courante.

```
object imap_check (resource imap_stream)
```

imap_check() retourne les informations à propos de la boîte aux lettres courante. **imap_check()** retourne FALSE en cas d'échec.

imap_check() vérifie le statut de la boîte aux lettres courante, sur le serveur *imap_stream*, et retourne les informations dans un objet avec les membres suivants :

- Date - Date de dernière modification du contenu de la boîte aux lettres
- Driver - protocole utilisé pour accéder à la boîte aux lettres: POP3, IMAP, NNTP.
- Mailbox - nom de la boîte aux lettres

- Nmsgs - nombre de messages de la boîte aux lettres
- Recent - nombre de messages récents de la boîte aux lettres

imap_clearflag_full (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Supprime un flag sur un message.

```
string imap_clearflag_full  (resource stream, string sequence, string flag, string
options)
```

imap_clearflag_full() efface le flag *flag* dans les messages de la séquence *sequence*, du flot imap *stream*.

Les options sont un masque de bit, qui accepte les valeurs suivantes :

ST_UID : la séquence contient des UIDs au lieu de numéro de séquence

imap_close (PHP 3, PHP 4 >= 4.0b1)

Termine un flot IMAP.

```
int imap_close (resource imap_stream, int flags)
```

imap_close() termine un flot IMAP. **imap_close()** prend un argument optionnel *flag*, CL_EXPUNGE, qui va retirer automatiquement de la liste la boîte aux lettres.

imap_createmailbox (PHP 3, PHP 4 >= 4.0b1)

Crée une nouvelle boîte aux lettres.

```
int imap_createmailbox (resource imap_stream, string mbox)
```

imap_createmailbox() crée une nouvelle boîte aux lettres nommée *mbox*. Les noms contenant des caractères spéciaux doivent être encodés.

imap_createmailbox() retourne TRUE en cas de succès, et FALSE en cas d'erreur.

Exemple 1. Exemple avec imap_createmailbox()

```
<?php
$mbox = imap_open("{your.imap.host}","utilisateur","mot_de_passe",OP_HALFOPEN)
    or die("connexion impossible: ".imap_last_error());
$name1 = "nouvellephpbox";
$name2 = imap_utf7_encode("nouvellephpboxéx");
$newname = $name1;
echo "Le nouveau nom sera '$name1'<br>\n";
# Nous allons créer maintenant une nouvelle boîte aux lettres "phptestbox"
# dans votre dossier inbox, vérifier son état et finalement, la supprimer
# pour remettre votre inbox dans son état initial.
if(@imap_createmailbox($mbox,imap_utf7_encode("{your.imap.host}INBOX.$newname"))){
 $status = @imap_status($mbox,"{your.imap.host}INBOX.$newname",SA_ALL);
```

```
 if($status) {
   print("Votre nouvelle boîte ’$name1’ est dans l’état suivant :<br>\n");
   print("Messages:   ". $status->messages   ."<br>\n";
   print("Récent:     ". $status->recent     ."<br>\n";
   print("Non lus:     ". $status->unseen     ."<br>\n";
   print("UID suivant:    ". $status->uidnext    ."<br>\n";
   print("UID validité:". $status->uidvalidity)."<br>\n";
   if(imap_renamemailbox($mbox,"{your.imap.host}INBOX.$newname","{your.imap.host}INBOX.$name2")) {
     echo "renommage de la boîte aux lettres ’$name1’ en ’$name2’<br>\n";
     $newname=$name2;
   } else {
     print "imap_renamemailbox sur la nou-
velle boîte aux lettres a échoué : ".imap_last_error()."<br>\n";
   }
 } else {
   print  "imap_status sur la nou-
velle boîte aux lettres a échoué : ".imap_last_error()."<br>\n";
 }
 if(@imap_deletemailbox($mbox,"{your.imap.host}INBOX.$newname")) {
   print "new mailbox supprimée pour remettre tout en état<br>\n";
 } else {
   print  "imap_deletemailbox ur la nou-
velle boîte aux lettres a échoué : ".implode("<br>\n",imap_errors())."<br>\n";
 }
} else {
 print  "Impossible de créer une nou-
velle boîte aux lettres : ".implode("<br>\n",imap_errors())."<br>\n";
}
imap_close($mbox);
?>
```

Voir aussi **imap_renamemailbox()**, **imap_deletemailbox()** et **imap_open()** pour connaître le format des noms de *mbox*.

imap_delete (PHP 3, PHP 4 >= 4.0b1)

Marque le fichier pour l’effacement, dans la boîte aux lettres courante.

```
int imap_delete (resource imap_stream, int msg_number [, int flags])
```

imap_delete() retourne TRUE.

imap_delete() marque le fichier *msg_number* pour l’effacement, dans la boîte aux lettres courante. Le paramètre optionnel *flags* ne prend qu’une seule valeur, *FT_UID*, qui indique à PHP qu’il faut traiter *msg_number* comme un *UID*. L’effacement réel n’interviendra que lors de l’appel de la fonction **imap_expunge()**.

Exemple 1. Exemple imap_delete()

```
<?php
$mbox = imap_open ("{your.imap.host}INBOX", "utilisateur", "mot_de_passe")
   or die ("connexion impossible: " . imap_last_error());
$check = imap_mailboxmsginfo ($mbox);
print "Nombre de messages avant effacement  : " . $check->Nmsgs . "<br>\n" ;
imap_delete ($mbox, 1);
$check = imap_mailboxmsginfo ($mbox);
print "Nombre de messages après effacement: " . $check->Nmsgs . "<br>\n" ;
imap_expunge ($mbox);
$check = imap_mailboxmsginfo ($mbox);
print "Nombre de messages après imap_expunge: " . $check->Nmsgs . "<br>\n" ;
imap_close ($mbox);
?>
```

imap_deletemailbox (PHP 3, PHP 4 >= 4.0b1)

Efface une boîte aux lettres.

```
int imap_deletemailbox (resource imap_stream, string mbox)
```

imap_deletemailbox() efface la boîte aux lettres (voir **imap_open()** pour connaître le format des noms de *mbox*).

imap_deletemailbox() retourne TRUE en cas de succès, et FALSE en cas d'erreur.

Voir aussi **imap_createmailbox()**, **imap_renamemailbox()** et **imap_open()** pour le format du paramètre *mbox*.

imap_errors (PHP 3>= 3.0.12, PHP 4 >= 4.0b4)

Retourne toutes les erreurs

```
array imap_errors (void)
```

imap_errors() retourne tous les messages d'erreurs IMAP générés depuis le dernier appel à **imap_errors()**, ou depuis le début de la page. Lorsque **imap_errors()** est appelés, la pile d'erreur est vidée.

imap_expunge (PHP 3, PHP 4 >= 4.0b1)

Efface tous les messages marqués pour l'effacement.

```
int imap_expunge (resource imap_stream)
```

imap_expunge() efface tous les messages marqués pour l'effacement par **imap_delete()**.

imap_expunge() retourne TRUE.

imap_fetch_overview (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Lit un sommaire des en-têtes de messages

```
array imap_fetch_overview (resource imap_stream, string sequence [, int flags])
```

imap_fetch_overview() lit les en-têtes des courriers électroniques de la séquence *sequence* et retourne un sommaire de leur contenu. *sequence* va contenir une séquence d'indice de message ou d'UIDs, si *flags* cotient FT_UID. La valeur retournée est un tableau d'objets, un par message d'en-tête décrit :

- subject - Le sujet du message
- from - Expéditeur
- date - Date d'expédition
- message_id - Identification du message
- references - est une référence sur l'id de ce message
- size - taille en octets

- uid - UID du message dans la boîte aux lettres
- msgno - numéro de séquence du message dans la boîte
- recent - Ce message est récent
- flagged - Ce message est marqué
- answered - Ce message a donné lieu à une réponse
- deleted - Ce message est marqué pour l'effacement
- seen - Ce message est déjà lu
- draft - Ce message est un brouillon

Exemple 1. Exemple avec imap_fetch_overview()

```
<?php
$mbox = imap_open("{votre.hote.imap}","utilisateur","mot_de_passe")
    or die("connexion impossible : ".imap_last_error());
$overview = imap_fetch_overview($mbox,"2,4:6",0);
if(is_array($overview)) {
       reset($overview);
       while( list($key,$val) = each($overview)) {
              print     $val->msgno
              . " - " . $val->date
              . " - " . $val->subject
              . "\n";
       }
}
imap_close($mbox);
?>
```

Voir aussi **imap_fetchstructure()**.

imap_fetchbody (PHP 3, PHP 4 >= 4.0b1)

Retourne une section extraite du corps d'un message.

```
string imap_fetchbody (resource imap_stream, int msg_number, string part_number [, flags
flags])
```

imap_fetchbody() va rechercher une section du corps du message, et la retourne sous la forme d'une chaîne. La section est une chaîne d'entiers, séparés par des virgules, qui servent d'index dans le corps du message, comme spécifié dans la norme IMAP4. Le texte n'est alors pas décodé par **imap_fetchbody()**.

L'option **imap_fetchbody ()** est un masque qui peut contenir les valeurs suivantes :

- FT_UID - msgono est un UID
- FT_PEEK - Ne pas lever le drapeau \Seen (Message lu) s'il n'est pas déjà levé.
- FT_INTERNAL - La chaîne renvoyée est au format interne, et ne va pas canoniser les CRLF.

imap_fetchheader (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Retourne l'en-tête d'un message.

```
string imap_fetchheader (resource imap_stream, int msgno, int flags)
```

imap_fetchheader() retourne l'en-tête brute et complète RFC 822 du message *msgno*, et le retourne sous la forme d'une chaîne.

Les options sont :

FT_UID L'argument *msgno* est un UID
FT_INTERNAL la chaîne renvoyée est au format "internal" ,
c'est-à-dire sans canonisation des CRLF
FT_PREFETCHTEXT RFC822.TEXT doit être pré
téléchargé en même temps que l'en-tête.
Cela réduit le RTT sur une connexion IMAP, si
le message complet est souhaité. (e.g. dans une opération
de sauvegarde dans un fichier).

imap_fetchstructure (PHP 3, PHP 4 >= 4.0b1)

Lit la structure d'un message.

```
object imap_fetchstructure  (resource imap_stream, int msg_number [, int flags])
```

imap_fetchstructure() la structure du message *msg_number*. **imap_fetchstructure()** dispose d'une option [flags], qui une seule valeur, *FT_UID*, pour indiquer que l'argument *msg_number* est un *UID*. **imap_fetchstructure()** retourne un objet avec des propriétés d'enveloppe, de date interne, de taille, de structure de flags et de corps, ainsi qu'un objet pour chaque attachement. La structure est la suivante :

Tableau 1. Objets retournés par imap_fetchstructure()

type	Type primaire de corps
encoding	Codage de transfert du corps
ifsubtype	TRUE s'il y a une chaîne de sous type
subtype	sous typeMIME
ifdescription	TRUE s'il y au ne chaîne de description
description	Chaîne de description du contenu
ifid	TRUE s'il y a une chaîne d'identification
id	Chaîne d'identification
lines	Nombre de lignes
bytes	Nombre d'octets
ifdisposition	TRUE s'il y a une chaîne de disposition
disposition	Chaîne de disposition
ifdparameters	TRUE s'il y a un tableau de paramètres dparameters
dparameters	tableau de disposition
ifparameters	TRUE si le tableau de paramètres existe
parameters	Tableau de paramètres MIME

parts	Tableau d’objet décrivant chaque partie du message

1. dparameters est un tableau d’objet où chaque objet à un "attribut" et une "valeur".
2. parameter est un tableau d’objet où chaque objet à un "attribut" et une "valeur".
3. parts est un tableau d’objets de même structure que l’objet supérieur, mais qui ne contient pas d’autres objets de même sorte.

Tableau 2. Type primaire de corps

0	text
1	multipart
2	message
3	application
4	audio
5	image
6	vidéo
7	autre

Tableau 3. Codage de transfert

0	7BIT
1	8BIT
2	BINARY
3	BASE64
4	QUOTED-PRINTABLE
5	OTHER

Voir aussi **imap_fetchstructure()**.

imap_get_quota (PHP 4 >= 4.0.5)

Lit les quotas des boîtes aux lettres

```
array imap_get_quota (resource imap_stream, string quota_root)
```

imap_get_quota() retourne un tableau contenant les valeurs de quota et courante de la boîte aux lettres *quota_root*. Le quota représente la taille maximale de votre boîte aux lettres. La valeur courante est l’espace actuellement utilisé par votre boîte aux lettres. **imap_get_quota()** retournera FALSE en cas d’échec.

imap_get_quota() ne fonctionne actuellement qu’avec les librairies c-client2000.

imap_stream doit avoir été créé avec la fonction **imap_open()**. Ce flot est nécessairement ouvert en tant qu’administrateur du serveur, pour que les droits nécessaires lui soit alloué. *quota_root* doit être de la forme : "user.nom", où "nom" est le nom de la boîte aux lettres que vous souhaitez analyser.

Exemple 1. Exemple avec imap_get_quota()

```
<?php
$mbox = imap_open("{votre.hote.imap}","mailadmin","mot de passe",OP_HALFOPEN)
     or die("Connexion impossible : ".imap_last_error());
$quota_value = imap_get_quota($mbox, "user.toto");
if(is_array($quota_value)) {
   print "Utilisation actuelle : " . $quota_value['usage'];
   print "Quota : " . $quota_value['limit'];
}
imap_close($mbox);
?>
```

Voir aussi **imap_open()** et **imap_set_quota()**.

imap_getmailboxes (PHP 3>= 3.0.12, PHP 4 >= 4.0b4)

Liste les boîtes aux lettres, et retourne le détail pour chacune.

```
array imap_getmailboxes (resource imap_stream, string ref, string pat)
```

imap_getmailboxes() retourne un tableau d'objets contenant les informations sur les boîtes aux lettres. Chaque objet a les attributs de *name*, qui contient le nom complet de la boîte aux lettres; *delimiter*, qui est le délimiteur hiérarchique; et *attributes*. *Attributes* est un masque de bits, qui contient :

- LATT_NOINFERIORS - Cette boîte aux lettres n'a pas d'"enfants" (il n'y a plus de boîtes aux lettres en dessous de celle-ci).
- LATT_NOSELECT - Ceci est juste un container, pas une boîte aux lettres (vous ne pouvez pas l'ouvrir).
- LATT_MARKED - Cette boîte aux lettres est marquée. Utilisé uniquement avec UW-IMAPD.
- LATT_UNMARKED - Cette boîte aux lettres n'est pas marquée. Utilisé uniquement avec UW-IMAPD.

ref ne devrait être que le serveur IMAP sous la forme `{imap_server:imap_port}`, et *pattern* spécifie la position dans la hiérarchie des boîtes aux lettres, où il faut commencer à charcher. Si vous voulez passer en revue toute la hiérarchier, passez '*' comme *pattern*.

Il y a deux caractères spéciaux que vous pouvez utiliser dans *pattern* : '*' et '%'. '*' signifie : toutes les boîtes aux lettres. Si vous passez *pattern* comme '*', vous obtiendrez la liste complète des boîtes aux lettres de la hiérarchie. '%' signifie qu'on ne s'interesse qu'au niveau courant. '%' passé à *pattern* ne retournera que les boîtes aux lettres de niveau supérieur; '~/mail/%'.Sous UW_IMAPD retournera toutes les boîtes aux lettres du dossier `~/mail directory`, mais pas leurs enfants.

Exemple 1. Exemple avec imap_getmailboxes()

```
<?php
$mbox = imap_open("{your.imap.host}","utilisateur","mot_de_passe",OP_HALFOPEN)
     or die("connexion impossible: ".imap_last_error());
$list = imap_getmailboxes($mbox,"{your.imap.host}","*");
if(is_array($list)) {
 reset($list);
 while (list($key, $val) = each($list))
 {
   print "($key) ";
   print imap_utf7_decode($val->name).",";
   print "'".$val->delimiter."',";
   print $val->attributes."<br>\n";
 }
} else
```

```
 print "imap_getmailboxes a échoué : ".imap_last_error()."\n";
imap_close($mbox);
?>
```

Voir aussi **imap_getsubscribed()**.

imap_getsubscribed (PHP 3>= 3.0.12, PHP 4 >= 4.0b4)

Liste toutes les boîtes aux lettres souscrites.

```
array imap_getsubscribed (resource imap_stream, string ref, string pattern)
```

imap_getsubscribed() est identique à **imap_getmailboxes()**, mais ne retourne que les boîtes aux lettres auxquelles l'utilisateur est inscrit.

imap_header (PHP 3, PHP 4 >= 4.0b1)

Lit l'en-tête d'un message.

```
object imap_header (resource imap_stream, int msg_number [, int fromlength [, int
subjectlength [, string defaulthost]]])
```

imap_header() est un alias de **imap_headerinfo()** et lui est identique en tous points.

imap_headerinfo (PHP 3, PHP 4 >= 4.0b1)

Lit l'en-tête du message

```
object imap_headerinfo (resource imap_stream, int msg_number [, int fromlength [, int
subjectlength [, string defaulthost]]])
```

imap_headerinfo() retourne un objet contenant divers éléments d'en-tête.

remail, date, Date, subject, Subject, in_reply_to, message_id,
newsgroups, followup_to, references
éléments d'en-tête :
Recent - 'R' si récent et lu
'N' si récent et non lu
' ' si non récent
Unseen - 'U' si non lu ET non récent
' ' si lu OU non lu et récent
Answered -'A' si répondu,
' ' si non répondu
Deleted - 'D' si effacé,
' ' si non effacé
Draft - 'X' si brouillon,
' ' si non brouillon
Flagged - 'F' si marqué,
' ' si non marqué
Notez bien que le comportement récent/non lu est un peu particulier :
si vous voulez savor si un message est non lu, vous devez le vérifier

avec
Unseen == 'U' || Recent == 'N'
toaddress (toute la ligne d'en-tête To: jusqu'à 1024 caractères)
to[] (retourne un objet avec tout l'en-tête To, contenant):
 personal
 adl
 mailbox
 host
fromaddress (toute la ligne d'en-tête from: jusqu'à 1024 caractères)
from[] (retourne un objet avec tout l'en-tête From, contenant):
 personal
 adl
 mailbox
 host
ccaddress (toute la ligne d'en-tête CC: jusqu'à 1024 caractères)
cc[] (retourne un objet avec tout l'en-tête CC, contenant):
 personal
 adl
 mailbox
 host
bccaddress (toute la ligne d'en-tête BCC: jusqu'à 1024 caractères)
bcc[] (retourne un objet avec tout l'en-tête BCC, contenant):
 personal
 adl
 mailbox
 host
reply_toaddress (oute la ligne d'en-tête Reply_to: jusqu'à 1024 caractères)
reply_to[] (retourne un objet avec tout l'en-tête Reply_to, contenant)
 personal
 adl
 mailbox
 host
senderaddress (toute la ligne d'en-tête Sender: jusqu'à 1024 caractères)
sender[] (retourne un objet avec tout l'en-tête Sender, contenant)
 personal
 adl
 mailbox
 host
return_path (toute la ligne d'en-tête Return-path: jusqu'à 1024 caractères)
return_path[] (retourne un objet avec tout l'en-tête Return-path, contenant)
 personal
 adl
 mailbox
 host
udate (Date du mail, au format UNIX)
fetchfrom (Ligne d'en-tête from formatée pour tenir dans *fromlength*
caractères)
fetchsubject (Ligne d'en-tête subject formatée pour tenir dans *subjectlength* caractères)

imap_headers (PHP 3, PHP 4 >= 4.0b1)

Retourne les en-têtes de tous les messages d'une boîte aux lettres.

```
array imap_headers (resource imap_stream)
```

imap_headers() retourne un tableau de chaîne contenant les en-tête des messages. Une chaîne par message.

imap_last_error (PHP 3>= 3.0.12, PHP 4 >= 4.0b4)

Retourne la dernière erreur (si elle existe) qui est survenu lors de la dernière requête.

```
string imap_last_error (void)
```

imap_last_error() retourne le texte complet de la dernière erreur IMAP (si elle existe) qui est survenu lors de la dernière requête. La pile d'erreur n'est pas touchée. Appeler **imap_last_error()** successivement dans nouvelles erreurs retournera la même erreur.

imap_listmailbox (PHP 3, PHP 4 >= 4.0b1)

Liste les boîtes aux lettres.

```
array imap_listmailbox (resource imap_stream, string ref, string pat)
```

imap_listmailbox() retourne un tableau contenant les noms des boîtes aux lettres.

Exemple 1. Exemple avec imap_listmailbox()

```
<?php
$mbox = imap_open("{your.imap.host}","utilisateur","mot_de_passe",OP_HALFOPEN)
     or die("connexion impossible: ".imap_last_error());
$list = imap_listmailbox($mbox,"{your.imap.host}","*");
if(is_array($list)) {
 reset($list);
 while (list($key, $val) = each($list))
   print imap_utf7_decode($val)."<br>\n";
} else
 print "imap_listmailbox a échoué: ".imap_last_error()."\n";
imap_close($mbox);
?>
```

imap_listsubscribed (PHP 3, PHP 4 >= 4.0b1)

Liste les boîtes aux lettres souscrites.

```
array imap_listsubscribed (resource imap_stream, string ref, string pattern)
```

imap_listsubscribed() retourne un tableau avec toutes les boîtes aux lettres auxquelles vous avez souscrit. Les arguments *ref* et *pattern* indiquent respectivement, le dossier où chercher et le nom des boîtes recherchées, sous la forme d'un masque.

imap_mail (PHP 3>= 3.0.14, PHP 4 >= 4.0b4)

Envoie un message mail

```
string imap_mail (string to, string subject, string message [, string additional_headers [,
string cc [, string bcc [, string rpath]]]])
```

imap_mail() est uniquement disponible sous PHP 3.

imap_mail_compose (PHP 3>= 3.0.5, PHP 4 >= 4.0b1)

Crée un message MIME

```
string imap_mail_compose (array envelope, array body)
```

Exemple 1. Exemple imap_mail_compose()

```
<?php
$envelope["from"]="musone@afterfive.com";
$envelope["to"]="musone@darkstar";
$envelope["cc"]="musone@edgeglobal.com";
$part1["type"]=TYPEMULTIPART;
$part1["subtype"]="mixed";
$filename="/tmp/imap.c.gz";
$fp=fopen($filename,"r");
$contents=fread($fp,filesize($filename));
fclose($fp);
$part2["type"]=TYPEAPPLICATION;
$part2["encoding"]=ENCBINARY;
$part2["subtype"]="octet-stream";
$part2["description"]=basename($filename);
$part2["contents.data"]=$contents;
$part3["type"]=TYPETEXT;
$part3["subtype"]="plain";
$part3["description"]="description3";
$part3["contents.data"]="contents.data3\n\n\n\t";
$body[1]=$part1;
$body[2]=$part2;
$body[3]=$part3;
echo nl2br(imap_mail_compose($envelope,$body));
?>
```

imap_mail_copy (PHP 3, PHP 4 >= 4.0b1)

Copie les messages spécifiés dans une boîte aux lettres.

```
int imap_mail_copy (resource imap_stream, string msglist, string mbox, int flags)
```

imap_mail_copy() copie les messages email spécifiés par *msglist* dans la boîte aux lettres nommée *mbox*. *msglist* est un intervalle, et pas seulement une liste numéros de message.

imap_mail_copy() retourne TRUE en cas de succès et FALSE en cas d'erreur.

flags est un masque, qui peut contenir une ou plusieurs des valeurs suivantes :

- CP_UID - la séquence de nombre contient des UIDS
- CP_MOVE - Efface les messages après copie.

imap_mail_move (PHP 3, PHP 4 >= 4.0b1)

Déplace les messages spécifiés dans une boîte aux lettres.

```
int imap_mail_move (resource imap_stream, string msglist, string mbox [, int flags])
```

imap_mail_move() déplace les messages spécifiés par *msglist* dans la boîte aux lettres *mbox*. *msglist* est un intervalle, et pas seulement une liste de messages.

flags est un champs de bit et peut contenir une seule valeur :

- CP_UID - La séquence de nombrs contient UIDS

imap_mail_move() retourne TRUE en cas de succès, et FALSE en cas d'erreur.

imap_mailboxmsginfo (PHP 3>= 3.0.2, PHP 4 >= 4.0b1)

Lit les informations à propos de la boîte aux lettres courante.

```
object imap_mailboxmsginfo (resource imap_stream)
```

imap_mailboxmsginfo() retourne les informations à propos de la boîte aux lettres courante. **imap_mailboxmsginfo()** retourne FALSE en cas d'échec.

imap_mailboxmsginfo() vérifie le statut courant de la boîte aux lettres sur le serveur, et retourne un objet avec les propriétés suivantes :

Tableau 1. Propriétés de boîte aux lettres

Date	Date de dernière modification du contenu de la boîte aux lettres
Driver	Pilote
Mailbox	Nom de la boîte aux lettres
Nmsgs	Nombre de messages
Recent	Nombre de messages récents
Unread	Nombre de messages non lus
Deleted	Nombre de messages effacés
Size	Taille de la boîte aux lettres

Exemple 1. Exemple avec imap_mailboxmsginfo()

```
<?php
$mbox = imap_open("{your.imap.host}INBOX","utilisateur", "mot_de_passe")
     or die("conexion impossible: ".imap_last_error());
$check = imap_mailboxmsginfo($mbox);
if($check) {
   print "Date: "    . $check->Date    ."<br>\n" ;
```

```
   print "Pilote: "  . $check->Driver  ."<br>\n" ;
   print "Mailbox: " . $check->Mailbox ."<br>\n" ;
   print "Messages: ". $check->Nmsgs   ."<br>\n" ;
   print "Récent: "  . $check->Recent  ."<br>\n" ;
   print "Non lus: "  . $check->Unread  ."<br>\n" ;
   print "Effacés: " . $check->Deleted ."<br>\n" ;
   print "Taille: "    . $check->Size    ."<br>\n" ;
} else {
   print "imap_check() a échoué: ".imap_last_error(). "<br>\n";
}
imap_close($mbox);
?>
```

imap_mime_header_decode (PHP 3>= 3.0.17, PHP 4 >= 4.0RC1)

Décode les éléments MIME d'une en-tête

```
array imap_mime_header_decode (string text)
```

imap_mime_header_decode() décode un message MIME qui contient des données non ASCII (Voir RFC2047 (http://www.faqs.org/rfcs/rfc2047.html)) Les éléments décodés sont retournés dans un tableau d'objets. Chacun de ces objets a deux propriétés : "charset" & "text". Si l'élément n'a pas été encodé, ou, en d'autres termes, sil il est en clair (plain US_ASCII), la propriété "charset" est mise à "default".

Exemple 1. Exemple imap_mime_header_decode()

```
<?php
$text="=?ISO-8859-1?Q?Keld_J=F8rn_Simonsen?= <keld@dkuug.dk>";
$elements=imap_mime_header_decode($text);
for($i=0;$i<count($elements);$i++) {
   echo "Charset: {$elements[$i]->charset}\n";
   echo "Texte: {$elements[$i]->text}\n\n";
}
?>
```

Dans l'exemple ci-dessus, on trouve deux éléments : le premier a été encodé en ISO-8859-1, et le second est en clair.

imap_msgno (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Retourne le numéro de séquence de message pour un UID donné.

```
int imap_msgno (resource imap_stream, int uid)
```

imap_msgno() retourne le numéro de séquence de message pour l'UID *uid*. C'est la fonction contraire de **imap_uid()**.

imap_num_msg (PHP 3, PHP 4 >= 4.0b1)

Retourne le nombre de message dans la boîte aux lettres courante.

```
int imap_num_msg (resource imap_stream)
```

imap_num_msg() retourne le nombre de message dans la boîte aux lettres courante.

Voir aussi **imap_num_recent()** et **imap_status()**.

imap_num_recent (PHP 3, PHP 4 >= 4.0b1)

Retourne le nombre de messages récents dans la boîte aux lettres courante.

```
int imap_num_recent (resource imap_stream)
```

imap_num_recent() retourne le nombre de message récents dans la boîte aux lettres courante.

Voir aussi **imap_num_msg()** et **imap_status()**.

imap_open (PHP 3, PHP 4 >= 4.0b1)

Ouvre un flot IMAP vers une boîte aux lettres.

```
int imap_open (string mailbox, string username, string password [, int flags])
```

imap_open() retourne un flot IMAP en cas de succès, et FALSE en cas d'erreur. **imap_open()** peut aussi être utilisée pour ouvrir des flots sur des serveurs POP3 et NNTP.

Un nom de boîte aux lettres est constitué d'une adresse de serveur, et d'une adresse de boîte sur ce serveur. Le mot réservé INBOX représente la boîte aux lettres de l'utilisateur courant. L'adresse du serveur, mise entre accolades '{' et '}', est constitué du nom du serveur ou de son adresse IP, d'une spécification de protocole (commençant par '/') et d'un port optionnel (spécifié avec ':'). Cette partie est obligatoire dans les paramètres de la boîte aux lettres. Les noms de boîtes aux lettres qui contiennent des caractères spéciaux (en dehors de l'espace ASCII) doivent être encodés avec **imap_utf7_encode()**.

Les options sont un masque de bit, qui peut prendre une ou plusieurs des valeurs suivantes :

- OP_READONLY - Ouvre une boîte aux lettres en lecture seule
- OP_ANONYMOUS - Ne pas utiliser, ou modifier le fichier .newsrc pour les news.
- OP_HALFOPEN - Pour les noms IMAP et NNTP, ouvre une connexion mais n'ouvre pas une boîte aux lettres.
- CL_EXPUNGE - Supprime automatiquement la boîte aux lettres de la liste, lors de la terminaison du flot.

Pour se connecter à un serveur IMAP, on peut utiliser la commande suivante :

```
<?php
$mbox  = imap_open("{localhost:143}INBOX","user_id","password");
?>
```

Pour se connecter à un serveur POP3 qui fonctionne sur le port 110 de la machine locale on peut utiliser la commande suivante :

```
<?php
$mbox = imap_open("{localhost:110/pop3}INBOX","user_id","password");
?>
```

Pour se connecter à un serveur IMAP SSL ou POP3 SSL, ajoutez /ssl après le protocole :

```
<?php
$mbox = imap_open ("{localhost:993/imap/ssl}INBOX", "user_id", "password");
?>
```

Pour se connecter à un serveur SSL IMAP ou POP3 avec un certificat ajoutez `/ssl/novalidate-cert` après le protocole :

```
<?php
$mbox = imap_open ("{localhost:995/pop3/ssl/novalidate-cert}", "user_id", "password");
?>
```

Pour se connecter à un serveur NNTP qui fonctionne sur le port 119 de la machine locale on peut utiliser la commande:

```
<?php
$nntp = imap_open("{localhost:119/nntp}comp.test","","");
?>
```

Pour se connecter à un serveur distant, remplacez "localhost" par le nom ou l'adresse IP de la machine.

Exemple 1. Exemple avec imap_open()

```
<?php
$mbox = imap_open ("{votre.hote.imap:143}", "nom_utilisateur", "mot de passe");
echo "<p><h1>Mailboxes</h1>\n";
$folders = imap_listmailbox ($mbox, "{votre.hote.imap:143}", "*");
if ($folders == FALSE) {
   echo "Appel échoué<br>\n";
} else {
   while (list ($key, $val) = each ($folders)) {
       echo $val."<br>\n";
   }
}
echo "<p><h1>en-têtes dans INBOX</h1>\n";
$headers = imap_headers ($mbox);
if ($headers == FALSE) {
   echo "Appel échoué<br>\n";
} else {
   while (list ($key,$val) = each ($headers)) {
       echo $val."<br>\n";
   }
}
imap_close($mbox);
?>
```

imap_ping (PHP 3, PHP 4 >= 4.0b1)

Vérifie que le flot IMAP est toujours actif.

```
int imap_ping (resource imap_stream)
```

imap_ping() retourne `TRUE` si le flot *`imap_stream`* existe toujours, et `FALSE` sinon.

imap_ping() vérifie que le flot IMAP est toujours actif, en lui envoyant un ping. Cette fonction permet de se rendre compte que du mail est arrivé : c'est même la méthode préconisée pour des tests périodiques de vérification du courrier. Cette fonction peut aussi servir à garder une connexion ouverte, avec les serveurs dotés d'un délai d'expiration.

imap_qprint (PHP 3, PHP 4 >= 4.0b1)

Convertit une chaîne à guillemets en une chaîne à 8 bits.

```
string imap_qprint (string string)
```

imap_qprint() convertit la chaîne à guillemets *string* en une chaîne à 8 bits.

imap_qprint() retourne une chaîne 8 bits (binaire).

Voir aussi **imap_8bit()**.

imap_renamemailbox (PHP 3, PHP 4 >= 4.0b1)

Renomme une boîte aux lettres.

```
int imap_renamemailbox (resource imap_stream, string old_mbox, string new_mbox)
```

imap_renamemailbox() renomme la boîte aux lettres *old_mbox* en *new_mbox*.

imap_renamemailbox() retourne TRUE en cas de succès, et FALSE en cas d'erreur.

Voir aussi **imap_createmailbox()**, **imap_deletemailbox()** et **imap_open()** pour le format de *mbox*.

imap_reopen (PHP 3, PHP 4 >= 4.0b1)

Ouvre un flot IMAP vers une nouvelle boîte aux lettres.

```
int imap_reopen (resource imap_stream, string mailbox [, string flags])
```

imap_reopen() réouvre la connexion spécifiée au serveur IMAP ou NNTP, avec une nouvelle boîtes aux lettres.

Les options sont des masques de bit, qui peuvent contenir les valeurs suivantes :

- OP_READONLY - Ouvre une boîte aux lettres en lecture seule
- OP_ANONYMOUS - Ne pas utiliser, ou modifier le fichier .newsrc pour les news
- OP_HALFOPEN - Pour les noms IMAP et NNTP, ouvre une connexion mais n'ouvre pas une boîte aux lettres.
- CL_EXPUNGE - Supprime automatiquement la boîte aux lettres de la liste, lors de la terminaison du flot. (voir **imap_delete()** et **imap_expunge()**).

imap_rfc822_parse_adrlist (PHP 3>= 3.0.2, PHP 4 >= 4.0b1)

Analyse une chaîne d'adresse.

```
string imap_rfc822_parse_adrlist  (string address, string default_host)
```

address analyse la chaîne *address* et essaie, pour chaque adresse, de retourner un tableau d'objets. Les 4 objets sont :

- mailbox - Le nom de la boîte aux lettres
- host - le nom de l'hôte

- personal - Le nom personnel
- adl - at domain source route (NDT : ???).

Exemple 1. Exemple avec imap_rfc822_parse_adrlist()

```
<?php
$address_string = "Hartmut Holzgraefe <hartmut@cvs.php.net>, postmas-
ter@somedomain.net, root";
$address_array  = imap_rfc822_parse_adrlist($address_string,"somedomain.net");
if(! is_array($address_array)) die("une erreur...\n");
reset($address_array);
while(list($key,$val)=each($address_array)){
 print "boîte   : ".$val->mailbox."<br>\n";
 print "hôte    : ".$val->host."<br>\n";
 print "personnel: ".$val->personal."<br>\n";
 print "adl     : ".$val->adl."<p>\n";
}
?>
```

imap_rfc822_parse_headers (PHP 4 >= 4.0RC1)

Analyse une en-tête mail

```
object imap_rfc822_parse_headers (string headers [, string defaulthost])
```

imap_rfc822_parse_headers() analyse la chaîne *headers*, et retourne un objet contenant differents éléments, similaires à la fonction **imap_header()**, hormis les flags, et autres éléments liés au serveur IMAP.

imap_rfc822_write_address (PHP 3>= 3.0.2, PHP 4 >= 4.0b1)

Retourne une adresse email proprement formatée

```
string imap_rfc822_write_address  (string mailbox, string host, string personal)
```

mailbox retourne une adresse email proprement formatée, à partir du nom de la boîte aux lettres de l'hôte *host*, et des informations personnelles *personal*.

Exemple 1. Exemple avec imap_rfc822_write_address()

```
<?php
print imap_rfc822_write_address("hartmut","cvs.php.net","Hartmut Holzgraefe")."\n";
?>
```

imap_scanmailbox (PHP 3, PHP 4 >= 4.0b1)

Lit la liste des boîtes aux lettres, et y recherche une chaîne.

```
array imap_scanmailbox (resource imap_stream, string string)
```

imap_scanmailbox() retourne un tableau contenant les noms des boîtes aux lettres qui contiennent la chaîne *`string`*. **imap_scanmailbox()** est simialaire à **imap_listmailbox()**, mais va aussi rechercher la chaîne *`string`* dans dans les données de la boîte aux lettres. Reportez-vous à **imap_getmailboxes()** pour une description des paramètres *`ref`* et *`pattern`*.

imap_search (PHP 3>= 3.0.12, PHP 4 >= 4.0b4)

Retourne un tableau de message après recherche.

```
array imap_search (resource imap_stream, string criteria, int flags)
```

imap_search() effectue une recherche dans la boîte aux lettres courante, sur le flot IMAP courant. *`criteria`* est une chaîne, délimitée par des espaces, dans laquelle les mots-clés suivants sont acceptés. Tous les arguments multi-mots doivent être entre guillemets :

- ALL - retourne tous les message qui vérifie le reste du critère.
- ANSWERED - tous les messages avec le flag \\ANSWERED
- BCC "string" - tous les messages avec la chaîne "string" dans le champs Bcc:
- BEFORE "date" - tous les messages avec Date: avant "date"
- BODY "string" - tous les messages avec "string" dans le corps
- CC "string" - tous les messages avec "string" dans le champs Cc:
- DELETED - tous les messages effacés
- FLAGGED - tous les messages avec le flag \\FLAGGED (parfois interprété comme Important ou Urgent)
- FROM "string" - tous les messages avec la chaîne "string" dan le champs From:
- KEYWORD "string" - tous les messags avec la chaîne "string" comme mot clé
- NEW - tous les nouveaux messages
- OLD - tous les anciens messages
- ON "date" - tous les messages avec la date "date" comme champs Date:
- RECENT - tous les messages avec le flag \\RECENT
- SEEN - tous les messages lus (avec le flag\\SEEN flag)
- SINCE "date" - tous les messages avec la date Date: après "date"
- SUBJECT "string" - tous les messages avec la chaîne "string" dans le champs Subject:
- TEXT "string" - tous les messages avec le texte "string"
- TO "string" - tous les messages avec la chaîne "string" dans le champs To:
- UNANSWERED - tous les messages non répondus
- UNDELETED - tous les messages non effacés
- UNFLAGGED - tous les messages non flaggés
- UNKEYWORD "string" - tous les messages dans le mot clés "string"
- UNSEEN - tous les messages non lus

Par exemple, pour rechercher les messages non répondus, envoyés par maman, vous pouvez utiliser : "UNANSWERED FROM maman". Les recherches semblent insensibles à la casse. Cette liste de critères est issue du code d'un client C UW et peut être incomplète ou inprécise. (voir aussi RFC2060, section 6.4.4).

Les valeurs pour les flags sont SE_UID, qui fait que le tableau réponse contient les UIDs plutôt que les numéros de séquence.

imap_set_quota (PHP 4 >= 4.0.5)

Modifie le quota d'une boîte aux lettres

```
int imap_set_quota (resource imap_stream, string quota_root, int quota_limit)
```

imap_set_quota() modifie le quota de la boîte aux lettres *quota_root*, en la fixant à *quota_limit*. **imap_set_quota()** requiert que *imap_stream* ait été ouvert avec un compte d'administrateur, pour avoir les droits nécessaires : elle ne fonctionnera avec aucun autre utilisateur.

imap_get_quota() ne fonctionne actuellement qu'avec les librairies c-client2000.

imap_stream doit avoir été créé avec la fonction **imap_open()**. Ce flot est nécessairement ouvert en tant qu'administrateur du serveur, pour que les droits nécessaires lui soit alloué. *quota_root* doit être de la forme : "user.nom", où "nom" est le nom de la boîte aux lettres que vous souhaitez analyser. *quota_limit* est la nouvelle taille maximum (en ko) de la boîte *quota_root*.

imap_set_quota() retourne TRUE en cas de succès, et FALSE sinon.

Exemple 1. Exemple avec imap_set_quota()

```
<?php
$mbox = imap_open ("{votre.hote.imap:143}", "mailadmin", "mot de passe");
if(!imap_set_quota($mbox, "user.toto", 3000)) {
   print "Erreur lors de la modification des quotas\n";
   return;
}
imap_close($mbox);
?>
```

Voir aussi **imap_open()** et **imap_set_quota()**.

imap_setflag_full (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Positionne un flag sur un message.

```
string imap_setflag_full (int stream, string sequence, string flag, string options)
```

imap_setflag_full() affecte le flag spécifié aux messages de la sequence donné.

Les flags que vous pouvez modifier sont "\\Seen", "\\Answered", "\\Flagged", "\\Deleted", "\\Draft" et "\\Recent" (comme défini dans la RFC2060).

Les options sont un masque de bits, et peuvent contenir les valeurs suivantes :

ST_UID la séquence contient des UIDs au lieu de numéro de séquence.

Exemple 1. Exemple avec imap_setflag_full()

```
<?php
$mbox = imap_open("{votre.hote.imap:143}","utilisateur","mot_de_passe")
    or die("can't connect: ".imap_last_error());
$status = imap_setflag_full($mbox,"2,5","\\Seen \\Flagged");
print gettype($status)."\n";
print $status."\n";
```

```
imap_close($mbox);
?>
```

imap_sort (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Trie des messages.

```
string imap_sort (int stream, int criteria, int reverse, int options)
```

imap_sort() retourne un tableau de nombre de message, triés suivant les paramètres suivants :

reverse vaut 1 pour signifier : tri inverse.

Les critères peuvent être un (et un seul) parmi les suivants :

SORTDATE Date du message
SORTARRIVAL Date d'arrivée
SORTFROM Nom de la première boîte aux lettres de l'adresse d'origine (From address)
SORTSUBJECT Sujet du message
SORTTO Nom de la première boîte aux lettres de destination (To address)
SORTCC Nom de la boîte aux lettres de copie cachée (cc address)
SORTSIZE Taille du message en octets

Les flags dont des masques de bits, d'un ou plusieurs des éléments suivants :

SE_UID Retourne l'UIDs à la place d'une séquence de nombres.
SE_NOPREFETCH Ne pas pré-télécharger les messages trouvés.

imap_status (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Retourne les informations de statut sur une boîte aux lettres autres que la boîte courante.

```
object imap_status (resource imap_stream, string mailbox, int options)
```

imap_status() retourne un objet contenant les informations de statut. Les options valables sont :

- SA_MESSAGES - met la valeur de status->messages au nombre de messages dans la boîtes aux lettres.
- SA_RECENT - met la valeur destatus->recent au nombre de messages récents dans la boîte aux lettres.
- SA_UNSEEN - met la valeur de status->unseen au nombre de messages non lus dans la boîte aux lettres.
- SA_UIDNEXT - met la valeur de status->uidnext à la prochaine valeur d'uid qui sera utilisée.
- SA_UIDVALIDITY - met la valeur de status->uidvalidity à une constante, qui change lorsque l'uid de la boîte aux lettres n'est plus valide.
- SA_ALL - fixe les valeurs de de toutes les précédents.

status->flags est aussi fixé : c'est un masque de bit qui peut contenir tous les flags ci-dessus.

Exemple 1. Exemple imap_status()

```
<?php
  $mbox = imap_open("{your.imap.host}","utilisateur","mot_de_passe",OP_HALFOPEN)
     or die("can't connect: ".imap_last_error());
  $status = imap_status($mbox,"{your.imap.host}INBOX",SA_ALL);
  if($status) {
    print("Messages:   ". $status->messages   )."<br>\n";
    print("Récents:     ". $status->recent     )."<br>\n";
    print("Non lus:     ". $status->unseen     )."<br>\n";
    print("UIDnext:    ". $status->uidnext    )."<br>\n";
    print("UIDvalidité:". $status->uidvalidity)."<br>\n";
  } else {
    print "imap_status a échoué : ".imap_last_error()."\n";
  }
  imap_close($mbox);
?>
```

imap_subscribe (PHP 3, PHP 4 >= 4.0b1)

Souscrit à une boîte aux lettres.

```
int imap_subscribe (resource imap_stream, string mbox)
```

imap_subscribe() souscrit à la boîte aux lettres *mbox*.

imap_subscribe() retourne TRUE en cas de succès, et FALSE en cas d'échec.

imap_uid (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Retourne l'UID d'un message.

```
int imap_uid (resource imap_stream, int msgno)
```

imap_uid() retourne l'UID pour le message *msgno*. Un UID est un identifant unique que ne change jamais, alors que le numéro du message dans la lsite des messages peut changer à toute modification de la boîte aux lettres. C'est la fonction contraire de **imap_msgno()**.

> **Note :** Cette fonctionnalité n'est pas supportées par les boîtes aux lettres POP3.

imap_undelete (PHP 3, PHP 4 >= 4.0b1)

Enlève la marque d'effacement d'un message.

```
int imap_undelete (resource imap_stream, int msg_number)
```

imap_undelete() enlève la marque d'effacement du message *msg_number*, placée avec **imap_delete()**.

imap_subscribe() retourne TRUE en cas de succès, et FALSE en cas d'erreur.

imap_unsubscribe (PHP 3, PHP 4 >= 4.0b1)

Termine la souscription à une boîte aux lettres.

```
int imap_unsubscribe (resource imap_stream, string mbox)
```

imap_unsubscribe() termine la souscription à la boîte aux lettres *mbox*.

imap_unsubscribe() retourne TRUE en cas de succès, et FALSE en cas d'erreur.

imap_utf7_decode (PHP 3>= 3.0.15, PHP 4 >= 4.0b4)

Décode une chaîne modifiée UTF-7.

```
string imap_utf7_decode (string text)
```

imap_utf7_decode() décode la chaîne UTF-7 *text* en données 8 bits.

imap_utf7_decode() retourne les données 8bits décodées, ou FALSE si la chaîne *text* n'est pas au format UTF-7. Cette fonction sert à décoder des noms de boîtes aux lettres qui contiennent des caractères internationaux hors de l'espace ASCII. Le standard UTF-7 est défini dans la RFC 2060 (http://www.faqs.org/rfcs/rfc2060.html), section 5.1.3 (l'original UTF-7 a été défini dans RFC1642 (http://www.faqs.org/rfcs/rfc1642.html)).

imap_utf7_encode (PHP 3>= 3.0.15, PHP 4 >= 4.0b4)

Convertit des données 8bit en texte UTF-7.

```
string imap_utf7_encode (string data)
```

imap_utf7_encode() retourne les données *data* 8bits encodées, ou FALSE si une erreur est survenue. Cette fonction sert à encoder des noms de boîtes aux lettres qui contiennent des caractères internationaux hors de l'espace ASCII. Le standard UTF-7 est défini dans la RFC 2060 (http://www.faqs.org/rfcs/rfc2060.html), section 5.1.3 (l'original UTF-7 a été défini dans RFC1642 (http://www.faqs.org/rfcs/rfc1642.html)).

imap_utf7_encode() retourne un texte UTF-7.

imap_utf8 (PHP 3>= 3.0.13, PHP 4 >= 4.0RC1)

Convertit du texte en UTF8

```
string imap_utf8 (string text)
```

imap_utf8() convertit le texte *text* en UTF8 (comme défini dans RFC2044 (http://www.faqs.org/rfcs/rfc2044.html)).

XXXVIII. Informix

Les pilotes d'accès à Informix pour Online (ODS) 7.x, SE 7.x, Universal Server (IUS) 9.x et IDS 2000 sont implémentés dans "functions/ifx.ec" et "functions/php3_ifx.h". Le support ODS 7.x est plutôt complet, et accepte les colonnes de type BYTE et TEXT. Le support IUS 9.x est partiellement fini, de nouveaux types sont disponibles, mais SLOB et CLOB sont toujours en cours de développement.

Notes de configuration : Vous avez besoin d'une version de ESQL/C pour compiler le pilote PHP d'Informix. Les versions ESQL/C 7.2x sont utilisables. ESQL/C fait partie du SDK Informix Client.

Avant que vous ne lanciez le script "configure", assurez-vous que la variable d'environnement "INFORMIXDIR" a été correctement paramétrée, et que $INFORMIXDIR/bin est dans votre PATH.

Le script de configuration va détecter automatiquement les librairies disponibles, et inclure les dossiers si vous lancer le script avec l'option `--with-informix=yes`. Vous pouvez ignorer cette détection en spécifiant "IFX_LIBDIR", "IFX_LIBS" et "IFX_INCDIR" dans votre environnement. Le script de configuration va aussi essayer de détecter la version de votre serveur Informix. Il modifiera alors la condition de compilation "HAVE_IFX_IUS" si votre serveur Informix est d'une version plus récente que 9.00.

Notes d'exécutions : Assurez-vous que les variables d'environnement INFORMIXDIR et INFORMIXSERVER sont accessibles au pilote PHP, et que le dossier bin INFORMIX est aussi dans la variable PATH. Vous pouvez le voir en lançant un script qui contient un appel à **phpinfo()** avant que vous ne commenciez à tester. La fonction **phpinfo()** affiche une liste des variables d'environnement. Cela fonctionne aussi bien en mode mod_php, qu'en mode CGI. Il vous faudra fixer les valeurs dans le script de démarrage d'Apache.

Les "Informix shared libraries" doivent aussi être accessibles au chargement (vérifiez LD_LINBRARY_PATH ou ld.so.conf/ldconfig).

Notes sur l'utilisation des BLOBs (TEXT et BYTE) : Les objets de type BLOBs sont normalement gérés par des identifiants de BLOB. Les requêtes de sélection retournent un identifiant de BLOB pour chaque colonne de type BYTE et TEXT. Vous pouvez en lire le contenu, avec des commandes de types "string_var = ifx_get_blob($BLOB_id);" ; si vous souhaitez ramener le BLOB en mémoire (avec: "ifx_blobinfile_mode(0);"). Si vous préférez recevoir le contenu d'une colonne BLOB dans un fichier, utilisez **ifx_blobinfile_mode()**, et ifx_get_blob($BLOB_id) vous retournera le nom du fichier. Utilisez les fonctions habituelles d'accès aux fichiers pour lire son contenu.

Pour les requêtes INSERT/UPDATE, vous devez créer les identifiants de BLOB par vous même, avec la fonction **ifx_create_blob()**. Puis, vous placez l'identifiant de BLOB dans un tableau, et remplacez la colonne par un point d'interrogation. Pour les UPDATE/INSERT, vous êtes responsable du contenu du BLOB, avec la fonction **ifx_update_blob()**.

Le comportement par défaut des colonnes de type BLOB peut être modifié en affectant de nouvelles valeurs aux variables de configuration (même à la volée) :

Variable de configuration : ifx.textasvarchar

Variable de configuration : ifx.byteasvarchar

Fonctions à utiliser lors de l'exécution :

ifx_textasvarchar(0) : Utilise l'identifiant de BLOB avec des colonnes de type TEXT, dans les requêtes SELECT

ifx_byteasvarchar(0) : Utilise l'identifiant de BLOB avec des colonnes de type BYTE, dans les requêtes SELECT

ifx_textasvarchar(1) : Retourne les colonnes de type TEXT sous la forme de VARCHAR, sans utiliser les identifiants de BLOB dans les requêtes SELECT.

ifx_byteasvarchar(1) : Retourne les colonnes de type BYTE sous la forme de VARCHAR, sans utiliser les identifiants de BLOB dans les requêtes SELECT.

Variable de configuration : ifx.BLOBinfile

Fonctions à utiliser lors de l'exécution :

ifx_blobinfile_mode(0) : Retourne les colonnes de type BYTE en mémoire, l'identifiant de BLOB vous donnera accès au contenu.

ifx_blobinfile_mode(1) : Retourne les colonnes de type BYTE dans un fichier, l'identifiant de BLOB vous donnera accès au nom de ce fichier.

En affectant la valeur de 1 à ifx_text/byteasvarchar, vous pouvez utiliser les colonnes de type TEXT et BYTE dans les requêtes SELECT comme des champs VARCHAR (mais plus long). Etant donné la gestion des chaînes par PHP, cette

technique conserve les données binaires. Les données retournées peuvent contenir n'importe quoi, et vous êtes responsable de la bonne manipulation de ces valeurs.

En affectant la valeur de 1 à **ifx_blobinfile_mode()**, utilisez le nom de fichier retourné par **ifx_get_blob()** pour accéder au contenu du BLOB. Notez bien que vous êtes tenu responsable de l'effacement des fichiers temporaires, créés par Informix. Chaque nouvelle ligne lue sur le serveur va créer un nouveau fichier temporaire, pour chaque colonne de type BYTE.

L'emplacement des fichiers temporaire peut être modifié, grâce à la variable "blobdir", (par défaut, ".", c'est-à-dire, le dossier courant). Une valeur telle que BLOBdir="tmpBLOB" simplifiera le nettoyage des fichiers temporaires, accidentellement oubliés (les noms commencent tous par "blb").

Suppression automatique des espaces (SQLCHAR et SQLNCHAR) : Elle peut être mise en place avec la variable de configuration.

ifx.charasvarchar : avec la valeur 1, les espaces de fin de champs seront automatiquement supprimés.

NULL values : Lorsque la variable de configuration ifx.nullformat (ou que la fonction **ifx_nullformat()**) est à un, les colonnes contenant la valeur NULL retourneront la chaîne "NULL", et sinon, retourneront une chaîne vide. Cela vous permet de faire la différence entre les colonnes vides et celles qui contiennent la valeur NULL.

ifx_connect (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Ouvre une connexion à un serveur Informix.

```
int ifx_connect (string [database], string [userid], string [password])
```

ifx_connect() retourne un identifiant de connexion, en cas de succès, et FALSE sinon.

ifx_connect() établit une connexion à un serveur Informix. Tous les arguments sont optionnels, et, s'ils viennent à manquer, les valeurs par défaut seront prises dans le fichier de configuration . (ifx.default_host pour l'hôte par défaut) (Les librairies Informix utiliseront la variable d'environnement $INFORMIXSERVER si ifx.default_host n'est pas définie). ifx.default_user pour l'utilisateur, et ifx.default_password comme mot de passe (si aucun n'a été défini).

Si un deuxième appel à **ifx_connect()** est fait avec les mêmes arguments, l'identifiant de connexion déjà ouvert sera retourné.

Le lien avec le serveur sera fermé dès que le script se termine, ce qui fait qu'il n'est pas nécessaire de terminer les connexions avec **ifx_close()**.

Voir aussi**ifx_pconnect()** et **ifx_close()**.

Exemple 1. Connection à un serveur Informix

```
<?php
$conn_id = ifx_pconnect ("mydb@ol_srv1", "imyself", "mypassword");
?>
```

ifx_pconnect (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Ouvre une connexion persistante à un serveur Informix.

```
int ifx_pconnect ([string database [, string userid [, string password]]])
```

ifx_pconnect() retourne un identifiant positif de connexion Informix, ou FALSE, en cas d'erreur.

ifx_pconnect() se comporte de manière très similaire à **ifx_connect()** avec deux différences importantes :

ifx_pconnect() se comporte exactement comme **ifx_connect()** lorsque PHP n'est pas un module Apache. Lors de la connexion, la fonction va chercher une connexion déjà ouverte avec le même hôte, le même nom d'utilisateur, et le même mot de passe. Si elle en trouve une, elle retournera un identifiant de cette connexion, au lieu d'en ouvrir une nouvelle.

Deuxièmement, la connexion au serveur SQL ne sera pas automatiquement refermée à la fin de l'exécution du script. Au contraire, le lien va rester ouvert (**ifx_close()** ne fermera pas les connexions établies avec **ifx_pconnect()**).

Ainsi, ce type de lien est appelé 'persistant'.

Voir aussi: **ifx_connect()**.

ifx_close (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Ferme une connexion à un serveur Informix.

```
int ifx_close ([int link_identifier])
```

ifx_close() retourne toujours TRUE.

ifx_close() ferme le lien au serveur de données Informix associé à l'identifant de connexion *link_identifier*. Si l'identifiant du lien n'est pas spécifié, la dernière connexion est utilisée.

Notez qu'il n'est généralement pas besoin d'appeler cette fonction, car les connexions non persistantes seront automatiquement fermées.

ifx_close() ne peut pas fermer une connexion ouverte avec **ifx_pconnect()**.

Voir aussi: **ifx_connect()** et **ifx_pconnect()**.

Exemple 1. Fermer une connexion Informix

```
<?php
$conn_id = ifx_connect ("mydb@ol_srv", "coucou", "cestmoi");
//... quelques requêtes diverses et variées ...
ifx_close($conn_id);
?>
```

ifx_query (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Envoie une requête Informix.

```
int ifx_query (string query [, int link_identifier [, int cursor_type, mixed
[blobidarray]]])
```

ifx_query() retourne un identifiant positif de résultat Informix en cas de succès, et `FALSE` en cas d'erreur.

L'entier de type "identifiant de résultat" est utilisé par4 d'autres fonctions pour lire les résultats. Pour un exemple, reportez-vous à **ifx_affected_rows()** pour connaître le nombre de lignes affectées.

ifx_query() envoie une requête au serveur actif courant, associé à l'identifiant de connexion *`link_identifier`*. Si *`link_identifier`* n'est pas fourni, la dernière connexion ouverte sera utilisée. Si aucune connexion n'a été ouverte, **ifx_query()** va essayer d'en créer une, en appelant **ifx_connect()**.

Exécute la requête *`query`* sur la connexion *`conn_id`*. Pour les requêtes de type SELECT, un pointeur est déclaré, et ouvert. L'option *`cursor_type`* permet de choisir le type de pointeur, "scroll" et/ou "hold". *`cursor_type`* accepte les deux valeurs séparées, et leur combinaison. Les requêtes d'autre type sont à exécution immédiate.

Le nombre de lignes affectées (estimé ou exact) est enregistré pour être lu avec **ifx_affected_rows()**.

Si vous avez une colonne de type BLOB (BYTE ou TEXT) dans une requête de modification, vous pouvez passer un paramètre *`BLOBidarray`* qui contiendra les identifiants des BLOB à modifier, et vous devrez remplacer cette colonne par un point d'interrogation (?) dans la requête.

Si le contenu d'une colonne de type TEXT (ou BYTE) vous pouvez aussi utiliser les fonctions **ifx_textasvarchar()** et **ifx_byteasvarchar()**. Cela vous permettra d'utiliser les colonnes TEXT (ou BYTE) comme des colonnes de type VARCHAR (mais plus long, tout de même), et vous n'aurez pas besoin de l'identifiant de BLOB.

Avec les fonctions **ifx_textasvarchar()** et **ifx_byteasvarchar()** (valeurs par défaut), les requêtes SELECT retourneront des identifiants de BLOB. Cet identifiant peut être une chaîne ou un fichier, suivant la configuration (voir plus loin).

Voir aussi : **ifx_connect()**.

Exemple 1. Afficher toutes les lignes de la table "ordres" sous la forme html

```
ifx_textasvarchar(1);      // Utilisation du mode "text mode" pour les BLOBs
$res_id = ifx_query("select * from orders", $conn_id);
if (! $res_id) {
    printf("Impossible de sélectionner des lignes dans : %s\n<br>%s<br>\n", ifx_error());
    ifx_errormsg();
    die;
}
ifx_htmltbl_result($res_id, "border=\"1\"");
ifx_free_result($res_id);
```

Exemple 2. Insertion de valeurs dans la table "catalogue"

```
<?php
// créer un identifiant de BLOB pour une colonne de type BYTE et une de type TEXT
$textid = ifx_create_blob(0, 0, "Colonne Text en mémoire");
$byteid = ifx_create_blob(1, 0, "Colonne Byte en mémoire");
// Stocke l'identifiant du BLOB dans le tableau BLOBid
$BLOBidarray[] = $textid;
$BLOBidarray[] = $byteid;
// éxécute la requête
$query = "insert into catalog (stock_num, manu_code, " .
         "cat_descr,cat_picture) values(1,'HRO',?,?)";
$res_id = ifx_query($query, $conn_id, $BLOBidarray);
if (! $res_id) {
//  ... erreur ...
}
// libération du résultat
ifx_free_result($res_id);
?>
```

ifx_prepare (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Prépare une requête SQL pour l'exécution.

```
int ifx_prepare (string query, int conn_id [, int cursor_def, mixed blobidarray])
```

ifx_prepare() retourne un entier identifiant de résultat *result_id* à utiliser avec **ifx_do()**. Modifie la valeur de *affected_rows*, pour accès ultérieur avec **ifx_affected_rows()**.

ifx_prepare() prépare la requête *query* sur la connexion *conn_id*. Pour les requêtes de type "select-type" un pointeur de résultat est déclaré et ouvert. L'option *cursor_type* permet de choisir le type de pointeur : "scroll" et/ou "hold". Les valeurs peuvent être combinées ensemble (IFX_SCROLL, IFX_HOLD).

Le nombre de ligne affectées (estimé ou exact) est enregistré, pour être lu avec la fonction **ifx_affected_rows()**.

Si vous avez une colonne de type BLOB (BYTE ou TEXT) dans une requête de modification, vous pouvez passer un paramètre *BLOBidarray* qui contiendra les identifiants des BLOB à modifier, et vous devrez remplacer cette colonne par un point d'interrogation (?) dans la requête.

Si le contenu d'une colonne de type TEXT (ou BYTE) vous pouvez aussi utiliser les fonctions **ifx_textasvarchar()** et **ifx_byteasvarchar()**. Cela vous permettra d'utiliser les colonnes TEXT (ou BYTE) comme des colonnes de type VARCHAR (mais plus long, tout de même), et vous n'aurez pas besoin de l'identifiant de BLOB.

Avec les fonctions **ifx_textasvarchar()** et **ifx_byteasvarchar()** (valeurs par défaut), les requêtes SELECT retourneront des identifiants de BLOB. Cet identifiant peut être une chaîne ou un fichier, suivant la configuration (voir plus loin).

Voir aussi **ifx_do()**.

ifx_do (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Exécute une requête SQL déjà préparée.

```
int ifx_do (int result_id)
```

ifx_do() retourne TRUE en cas de succès, FALSE en cas d'erreur.

Exécute une requête qui a déjà été préparée, ou crée un pointeur pour cela.

Ne libère pas *result_id* en cas d'erreur.

De plus, elle fixe la valeur de *result_id* pour accès ultérieur par **ifx_affected_rows()**.

Voir aussi : **ifx_prepare()**. Il y a un exemple.

ifx_error (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Retourne le code d'erreur de la dernière requête Informix.

```
string ifx_error (void)
```

ifx_error() retourne le code d'erreur de la dernière requête Informix. Les codes d'erreur Informix (SQLSTATE & SQLCODE) formaté comme suit :

x [SQLSTATE = aa bbb SQLCODE=cccc]

avec x = space : aucune erreur

E : erreur

N : il n'y a plus d'informations

W : Alerte

? : Indéfinie

Si le caractère vaut autre chose qu'un espace, SQLSTATE et SQLCODE décrit l'erreur avec plus de détails.

Reportez-vous au manuel Informix pour trouver la description de SQLSTATE et SQLCODE

ifx_error() retourne une chaîne avec un caractères, décrivant le résultat général de la commande, et aussi SQLSTATE et SQLCODE associé à la plus récente requête SQL exécutée. Le format de la chaîne est "(char) [SQLSTATE=(deux chiffres) (trois chiffres) SQLCODE=(un chiffre)]". Le premier caractère peut être ' ' (espace) (succès), 'W' (Alerte), 'E' (une erreur est survenue durant le traitement) ou 'N' (aucune donnée de retour).

Voir aussi: **ifx_errormsg()**.

ifx_errormsg (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Retourne le message d'erreur de la dernière requête Informix.

```
string ifx_errormsg ([int errorcode])
```

ifx_errormsg() retourne le plus récent message d'erreur ou, lorsque l'option *errorcode* est présent, le message d'erreur associé à *errorcode*.

Voir aussi: **ifx_error()**.

```
printf("%s\n<br>", ifx_errormsg(-201));
```

ifx_affected_rows (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Retourne le nombre de lignes affectées par une requête.

```
int ifx_affected_rows (int result_id)
```

ifx_affected_rows() retourne le nombre de lignes affectées par la requête associée à *result_id*.

result_id est un identifiant valide de résultat retourné par **ifx_query()** ou **ifx_prepare()**.

Pour les INSERT, UPDATE et DELETE, ce nombre est le nombre exact de lignes affectées (sqlerrd[2]). Pour les SELECT, ce n'est qu'une estimation (sqlerrd[0]). Ne vous y fiez pas.

ifx_affected_rows() est très pratique après **ifx_prepare()** pour limiter la taille des résultats.

Voir aussi : **ifx_num_rows()**.

Exemple 1. Nombre de lignes affectées

```
<?php
$rid = ifx_prepare ("select * from emp
                     where name like " . $name, $connid);
if (! $rid) {
   //... erreur ...
}
$rowcount = ifx_affected_rows ($rid);
if ($rowcount > 1000) {
    printf ("Trop de lignes trouvées (%d)\n<br>", $rowcount);
    die ("Ressayez avec une autre requête. <br>\n");
}
?>
```

ifx_getsqlca (PHP 3>= 3.0.8, PHP 4 >= 4.0b1)

Retourne le contenu de la variable sqlca.sqlerrd[0..5] après une requête.

```
array ifx_getsqlca (int result_id)
```

ifx_getsqlca() retourne une pseudo-ligne (tableau associatif) avec sqlca.sqlerrd[0] à sqlca.sqlerrd[5] après la requête associée *result_id*.

result_id est un identifiant valide de résultat retourné par **ifx_query()** ou **ifx_prepare()**.

Pour les requêtes INSERT, UPDATE et DELETE, les valeurs retournées sont celles fixées par le serveur après avoir exécuté la requête. Cela donne accès au nombre de ligne affectées, ainsi qu'au numéro de série d'insertion. Pour les requêtes de type SELECT, les valeurs retournées sont celles qui ont été préparées. Utiliser cette fonction économise l'exécution d'une requête "select dbinfo('sqlca.sqlerrdx')", étant donné qu'elle retourne les valeurs qui ont été sauvées par le pilote ifx au moment approprié.

Exemple 1. Lire les valeurs de sqlca.sqlerrd[x]

```
<?php
/* On suppose que la première colonne d'une table 'quelconque' est un numéro de série */
$qid = ifx_query("insert into sometable values(0, '2nd column', 'another col-
umn' ", $connid);
if (! $qid) {
//    ... erreur ...
}
$sqlca = ifx_getsqlca ($qid);
$serial_value = $sqlca["sqlerrd1"];
echo "Le numéro de série de la valeur insérée est : " . $serial_value<br>\n";
?>
```

ifx_fetch_row (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Retourne une ligne sous la forme d'un tableau énuméré.

```
array ifx_fetch_row (int result_id [, mixed position])
```

ifx_fetch_row() retourne un tableau associatif qui contient la ligne retournée, ou FALSE s'il ne reste plus de lignes à lire, ou s'il a eu une erreur.

Les colonnes de types BLOB sont retournées sous la forme d'un identifiant à utiliser avec **ifx_get_blob()** à moins que vous n'ayez utilisé la fonction **ifx_textasvarchar()** ou **ifx_byteasvarchar()**, et dans ce cas, les BLOBs seront retournés sous forme de chaîne. **ifx_fetch_row()** retourne FALSE en cas d'erreur.

result_id est un identifiant valide de résultat, retourné par **ifx_query()** ou **ifx_prepare()** (Requêtes SELECT seulement !).

position est un paramètre optionnel, pour une opération de lecture d'informations sur un pointeur de type "scroll": "NEXT", "PREVIOUS", "CURRENT", "FIRST", "LAST" ou encore un nombre. Si vous spécifiez un nombre, la ligne d'index absolu sera retournée. Ce paramètre est optionnel, et ne fonctionne qu'avec les pointeurs de type "scroll".

ifx_fetch_row() retourne une ligne de données d'un résultat associé à l'identifiant de résultat *result_id*. La ligne est retournée sous la forme d'un tableau associatif.

Les appels ultérieurs à **ifx_fetch_row()** retourneront la ligne suivante, ou FALSE s'il n'y a plus de ligne.

Exemple 1. Exemple avec ifx_fetch_row()

```
<?php
$rid = ifx_prepare ("select * from emp where name like " . $name,
                    $connid, IFX_SCROLL);
if (! $rid) {
//    ... erreur ...
}
$rowcount = ifx_affected_rows($rid);
if ($rowcount > 1000) {
    printf ("Trop de lignes dans le résultats. (%d)\n<br>", $rowcount);
    die ("Recommencez votre requête. <br>\n");
}
if (! ifx_do ($rid)) {
//   ... erreur ...
}
$row = ifx_fetch_row ($rid, "NEXT");
while (is_array($row)) {
    for(reset($row); $fieldname=key($row); next($row)) {
        $fieldvalue = $row[$fieldname];
        printf ("%s = %s,", $fieldname, $fieldvalue);
    }
    printf("\n<br>");
    $row = ifx_fetch_row ($rid, "NEXT");
}
ifx_free_result ($rid);
?>
```

ifx_htmltbl_result (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Lit toutes les lignes d'un tableau, et la met sous la forme d'un tableau HTML.

```
int ifx_htmltbl_result (int result_id [, string html_table_options])
```

ifx_htmltbl_result() lit toutes les lignes d'un tableau, et la met sous la forme d'un tableau HTML, ou FALSE en cas d'erreur.

Affiche les lignes avec des balises HTML. Le second argument permet de modifier les options de table.

Exemple 1. Affichage sous la forme d'une table HTML

```
<?php
$rid = ifx_prepare ("select * from emp where name like " . $name,
                    $connid, IFX_SCROLL);
if (! $rid) {
//   ... erreur ...
}
$rowcount = ifx_affected_rows ($rid);
if ($rowcount > 1000) {
    printf ("Trop de lignes dans le résultat : (%d)\n<br>", $rowcount);
    die ("Recommencez votre requête <br>\n");
}
if (! ifx_do($rid) {
//  ... erreur ...
}
ifx_htmltbl_result ($rid, "border=\"2\"");
ifx_free_result($rid);
?>
```

ifx_fieldtypes (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Liste les champs Informix SQL.

```
array ifx_fieldtypes (int result_id)
```

ifx_fieldtypes() retourne un tableau associatif avec les noms des champs comme clés, et les types SQL comme valeur. En cas d'erreur, retourne FALSE.

Exemple 1. Nom de champs et type SQL.

```
<?php
$types = ifx_fieldtypes ($resultid);
if (! isset ($types)) {
//  ... erreur ...
}
for ($i = 0; $i < count($types); $i++) {
    $fname = key($types);
    printf("%s :\t type =  %s\n", $fname, $types[$fname]);
    next($types);
}
?>
```

ifx_fieldproperties (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Liste les propriétés des champs SQL.

```
array ifx_fieldproperties (int result_id)
```

ifx_fieldproperties() retourne un tableau associatif avec les nom des champs comme clé, et les données de propriétés des champs comme valeur. **ifx_fieldproperties()** retourne FALSE en cas d'erreur.

ifx_fieldproperties() retourne les propriétés Informix SQL pour tous les champs d'une requête, sous la forme d'un tableau associatif. Les propriétés sont présentées sous la forme : "SQLTYPE;longueur ;précision;échelle;ISNULLABLE" avec SQLTYPE qui représente le type de données Informix tel que "SQLVCHAR" et ISNULLABLE = "Y" ou "N" (le champs peut contenir NULL ou pas : Oui ou Non).

Exemple 1. Exemple avec ifx_fieldproperties()

```
<?php
$properties = ifx_fieldtypes ($resultid);
if (! isset($properties)) {
//  ... erreur ...
}
for ($i = 0; $i < count($properties); $i++) {
    $fname = key ($properties);
    printf ("%s:\t type =  %s\n", $fname, $properties[$fname]);
    next ($properties);
}
?>
```

ifx_num_fields (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Retourne le nombre de colonnes dans une requête.

```
int ifx_num_fields (int result_id)
```

ifx_num_fields() retourne le nombre de colonnes dans la requête *result_id* ou FALSE en cas d'erreur.

Après avoir préparé ou exécuté une requête, cette fonction retourne le nombre de colonne dans la requête.

ifx_num_rows (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Compte le nombre de ligne déjà lues dans un résultat.

```
int ifx_num_rows (int result_id)
```

ifx_num_rows() compte le nombre de ligne déjà lues dans le résultat *result_id* après **ifx_query()** ou **ifx_do()**.

ifx_free_result (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Libère les ressources prises par un résultat.

```
int ifx_free_result (int result_id)
```

ifx_free_result() libère les ressources prises par le résultat *result_id*. **ifx_free_result()** retourne FALSE en cas d'erreur.

ifx_create_char (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Crée un objet char.

```
int ifx_create_char (string param)
```

ifx_create_char() crée un objet char. *param* sera le contenu de l'objet.

ifx_free_char (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Supprime un objet char.

```
int ifx_free_char (int bid)
```

ifx_free_char() supprime l'objet char *bid*. **ifx_free_char()** retourne FALSE en cas d'erreur, et sinon TRUE.

ifx_update_char (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Modifie le contenu d'un objet char.

```
int ifx_update_char (int bid, string content)
```

ifx_update_char() modifie le contenu de l'objet char repéré par son identifiant*bid*. *content* est une chaîne avec les nouvelles données. **ifx_update_char()** retourne FALSE en cas d'erreur, et sinon, TRUE.

ifx_get_char (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Retourne le contenu d'un objet char.

```
int ifx_get_char (int bid)
```

ifx_get_char() retourne le contenu de l'objet associé à l'identifiant *bid*.

ifx_create_blob (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Crée un objet BLOB.

```
int ifx_create_blob (int type, int mode, string param)
```

ifx_create_blob() crée un objet BLOB.

type: 1 = TEXT, 0 = BYTE

mode: 0 = L'objet BLOB place le contenu en mémoire ; 1 = L'objet BLOB place le contenu dans un fichier.

param: Si mode = 0: pointeur du contenu, si mode = 1: pointeur vers un fichier.

ifx_create_blob() retourne FALSE en cas d'erreur, et sinon, un identifiant de BLOB.

ifx_copy_blob (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Duplique un objet BLOB.

```
int ifx_copy_blob (int bid)
```

ifx_copy_blob() retourne FALSE en cas d'erreur, et sinon, l'identifiant du nouvel objet.

ifx_free_blob (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Supprime un objet BLOB.

```
int ifx_free_blob (int bid)
```

ifx_free_blob() supprime l'objet BLOB *bid*. **ifx_free_blob()** retourne FALSE en cas d'erreur, et sinon TRUE.

ifx_get_blob (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Retourne le contenu d'un objet BLOB.

```
int ifx_get_blob (int bid)
```

ifx_get_blob() retourne le contenu de l'objet BLOB associé à *bid*.

ifx_update_blob (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Modifie le contenu d'un objet BLOB.

```
boolean ifx_update_blob (int bid, string content)
```

ifx_update_blob() modifie le contenu de l'objet BLOB repéré par sont identifiant *bid*. *content* est une chaîne contenant les nouvelles données. **ifx_update_blob()** retourne FALSE en cas d'erreur, et sinon, TRUE.

ifx_blobinfile_mode (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Choisit le mode par défaut des objets BLOB pour toutes les requêtes SELECT.

```
void ifx_blobinfile_mode (int mode)
```

ifx_blobinfile_mode() modifie le mode par défaut des objets BLOB pour toutes les requêtes SELECT. Mode "0" chargera les BLOB de type Byte en mémoire ; Mode "1" sauvera les BLOB de type Byte dans un fichier.

ifx_textasvarchar (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Choisit le mode par défaut des objets text.

```
void ifx_textasvarchar (int mode)
```

ifx_textasvarchar() modifie le mode par défaut des objets TEXT. Le mode "0" retournera un identifiant de BLOB et le mode "1" retourne le BLOB sous la forme d'un (gros) varchar.

ifx_byteasvarchar (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Choisit le mode par défaut des objets BYTE.

```
void ifx_byteasvarchar (int mode)
```

ifx_byteasvarchar() modifie le mode par défaut des objets BYTE. Le mode "0" retournera l'identifiant de BLOB, et le mode "1" retourenra le contenu du TEXT sous la forme d'un VARCHAR.

ifx_nullformat (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Modifie le mode par défaut de lecture des valeurs.

```
void ifx_nullformat (int mode)
```

ifx_nullformat() modifie le mode par défaut de lecture des valeurs. Le mode "0" retourne "", et le mode "1" retourne "NULL".

ifxus_create_slob (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Crée un objet SLOB et l'ouvre.

```
int ifxus_create_slob (int mode)
```

ifxus_create_slob() crée un objet SLOB et l'ouvre. Les modes valides sont : 1 = LO_RDONLY, 2 = LO_WRONLY, 4 = LO_APPEND, 8 = LO_RDWR, 16 = LO_BUFFER, 32 = LO_NOBUFFER -> ou une combinaison des précédents. Vous pouvez aussi utiliser les constantes suivantes : IFX_LO_RDONLY, IFX_LO_WRONLY etc. **ifxus_create_slob()** retourne `FALSE` en cas d'erreur, et sinon, l'identifiant de l'objet SLOB.

ifx_free_slob (unknown)

Supprime un objet SLOB.

```
int ifxus_free_slob (int bid)
```

ifxus_free_slob() supprime un objet SLOB. *`bid`* est l'identifiant de l'objet SLOB. **ifxus_free_slob()** retourne `FALSE` en cas d'erreur, et sinon `TRUE`.

ifxus_close_slob (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Ferme un objet SLOB.

```
int ifxus_close_slob (int bid)
```

ifxus_close_slob() ferme l'objet SLOB représenté par son identifiant *`bid`*. **ifxus_close_slob()** retourne `FALSE` en cas d'erreur, et sinon, `TRUE`.

ifxus_open_slob (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Ouvre un objet SLOB.

```
int ifxus_open_slob (long bid, int mode)
```

ifxus_open_slob() ouvre un objet SLOB. *bid* est un identifiant d'objet SLOB. Les modes valides sont : 1 = LO_RDONLY, 2 = LO_WRONLY, 4 = LO_APPEND, 8 = LO_RDWR, 16 = LO_BUFFER, 32 = LO_NOBUFFER -> ou une combinaison des valeurs précédentes. **ifxus_open_slob()** retourne FALSE en cas d'erreur, et sinon, l'identifiant du nouvel objet.

ifxus_tell_slob (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Retourne le fichier courant, ou la position courante.

```
int ifxus_tell_slob (long bid)
```

ifxus_tell_slob() retourne le fichier courant, ou la position courante d'un objet SLOB ouvert. *bid* est un identifiant d'objet SLOB. **ifxus_tell_slob()** retourne FALSE en cas d'erreur, et sinon, la position du pointeur de fichier.

ifxus_seek_slob (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Fixe le fichier courant, ou la position courante.

```
int ifxus_seek_slob (long bid, int mode, long offset)
```

ifxus_seek_slob() modifie le fichier courant, ou la position du pointeur de fichier, pour un objet SLOB ouvert. *bid* est un identifiant d'objet SLOB. Les modes valides sont : 0 = LO_SEEK_SET, 1 = LO_SEEK_CUR, 2 = LO_SEEK_END et *offset* est un octet d'offset. **ifxus_seek_slob()** retourne FALSE en cas d'erreur, et sinon, la position du pointeur de fichier.

ifxus_read_slob (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Lit n bytes d'un objet SLOB.

```
int ifxus_read_slob (long bid, long nbytes)
```

ifxus_read_slob() lit *nbytes* octets de l'objet SLOB *bid*. *bid* est un identifiant d'objet SLOB existant, et *nbytes* est le nombre d'octets à lire. **ifxus_read_slob()** retourne FALSE en cas d'erreur, et sinon, une chaîne de caractères.

ifxus_write_slob (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Ecrit une chaîne dans un objet SLOB.

```
int ifxus_write_slob (long bid, string content)
```

ifxus_write_slob() écrit une chaîne dans un objet SLOB. *bid* est un identifiant d'objet SLOB et *content* sont les données à écrire. **ifxus_write_slob()** retourne FALSE en cas d'erreur, et sinon, le nombre d'octets écrits.

XXXIX. InterBase

Interbase est une base de données populaire, créée par Borland/Inprise. Pour plus d'informations sur Interbase, allez à http://www.interbase.com/. Par ailleurs, Interbase vient de rejoindre le mouvement Open Source!

Note : Le support intégral de InterBase 6 a été ajouté à PHP 4.0.

Cette base de données utilise les guillemets simples (') pour échapper les caractères, un peu comme le fait Sybase. Ajoutez à votre fichier `php.ini` la directive suivante :

```
magic_quotes_sybase = On
```

ibase_connect (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Ouvre une connexion à une base de données Interbase.

```
resource ibase_connect (string database [, string username [, string password [, string
charset [, int buffers [, int dialect [, string role]]]]]])
```

ibase_connect() établit une connexin avec un serveur InterBase. *database* doit être un chemin valide jusqu'à un fichier de base de donnés sur le serveur sur lequel il résde. Si le serveur est distant, il faut le préfixer avec un nom d'hôte 'hostname:' (TCP/IP), '//hostname/' (NetBEUI) ou 'hostname@' (IPX/SPX), en fonction du protocole de communication utilisé. *username* et *password* peuvent être spécifié dans les directives de configuration du PHP ibase.default_user et ibase.default_password. *charset* est le jeu de caractère par défaut de la base. *buffers* est le nombre de buffer de base à allouer pour le cache serveur. S'il est passé à 0 ou omis, le serveur choisira de lui-même. *dialect* sélectionne le dialecte SQL pour les requêtes exécutées avec cette connexion, et par défaut, il utilise le meilleur dialecte disponible.

Si un deuxième appel est fait avec **ibase_connect()**, en passant les mêmes arguments, une nouvelle connexion ne sera pas ouverte, mais la connexion déjà ouverte sera retournée. La connexion sera fermée dès que le script se termine, à moins qu'elle ne soit fermée explicitement avec **ibase_close()**, durant le script.

Exemple 1. Exemple ibase_connect()

```
<?php
    $dbh = ibase_connect($host, $username, $password);
    $stmt = 'SELECT * FROM tblname';
    $sth = ibase_query($dbh, $stmt);
    while ($row = ibase_fetch_object($sth)) {
        print $row->email . "\n";
    }
    ibase_close($dbh);
?>
```

Note : *buffers* a été ajouté en PHP 4-RC2.

Note : *dialect* a été ajouté en PHP 4-RC2. Il n'est opérationnel qu'avec les versions InterBase 6 et plus récentes.

Note : *role* a été ajouté en PHP 4-RC2. Il n'est opérationnel qu'avec les versions InterBase 5 et plus récentes.

Voir aussi **ibase_pconnect()**.

ibase_pconnect (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Ouvre une connexion persistante à une base de données Interbase.

```
resource ibase_pconnect (string database [, string username [, string password [, string
charset [, int buffers [, int dialect [, string role]]]]]])
```

ibase_pconnect() se comporte similairement à **ibase_connect()**, avec deux différences majeures : la première est que, lors de la connexion, la fonction va essayer de trouver une connexion (peristante) déjà ouverte. Si elle la trouve, cette dernière sera retournée, plutôt qu'une nouvelle connexion. Sinon, une nouvelle connexion sera ouverte. La deuxième est que la connexion ne sera pas fermée à la fin du script, mais restera ouverte pour utilisation ultérieure. (**ibase_close()** ne fermera pas une connexion ouverte avec **ibase_pconnect()**). Ce type de lien est alors dit 'persistant'.

Note : *buffers* a été ajouté en PHP 4-RC2.

Note : *dialect* a été ajouté en PHP 4-RC2. Il n'est opérationnel qu'avec les versions InterBase 6 et plus récentes.

Note : *role* a été ajouté en PHP 4-RC2. Il n'est opérationnel qu'avec les versions InterBase 5 et plus récentes.

Voir aussi **ibase_connect()** pour plus de détails sur les arguments de cette fonction.

ibase_close (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Ferme une connexion à une base de données Interbase.

```
int ibase_close ([resource connection_id])
```

ibase_close() ferme une connexion à une base de données Interbase. Cette fonction prend comme argument l'identifiant de connexion *connection_id* retourné par **ibase_connect()**. Si *connection_id* est omis, la dernière connexion ibase est fermée. Les transactions par défaut sont validées et les autres sont annulées.

ibase_query (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Exécute une requête sur une base Interbase

```
resource ibase_query ([resource link_identifier, string query [, int bind_args]])
```

ibase_query() exécute une requête sur une base Interbase, et retourne un identifiant de résultat, à utiliser avec **ibase_fetch_row()**, **ibase_free_result()** et/ou **ibase_free_query()**.

Note : Bien que ces fonctions supportent la liaison de variables avec des paramètres de requêtes, il n'y a pas d'intérêt spécial à les utiliser. Pour des exemples grandeur réelle, voyez **ibase_prepare()** et **ibase_execute()**.

ibase_fetch_row (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Lit une ligne dans une base Interbase

```
array ibase_fetch_row (resource result_identifier)
```

ibase_fetch_row() retourne la prochaine ligne spécifiée dans le résultat obtenu de **ibase_query()**.

ibase_fetch_object (PHP 3>= 3.0.7, PHP 4 >= 4.0RC1)

Lit une ligne dans une base Interbase dans un objet.

```
object ibase_fetch_object (resource result_id)
```

ibase_fetch_object() lit une ligne dans une base Interbase et la place dans un pseudo objet. **ibase_fetch_object()** prend comme argument l'identifiant de résultat *result_id* obtenu de **ibase_query()** ou **ibase_execute()**.

```
<php
```

```
    $dbh = ibase_connect($host, $username, $password);
    $stmt = 'SELECT * FROM tblname';
    $sth = ibase_query($dbh, $stmt);
    while ($row = ibase_fetch_object($sth)) {
        print $row->email . "\n";
    }
    ibase_close($dbh);
?>
```

Voir aussi **ibase_fetch_row()**.

ibase_field_info (PHP 3>= 3.0.7, PHP 4 >= 4.0RC1)

Lit les informations sur un champs

```
array ibase_field_info (resource result, int field_number)
```

ibase_field_info() retourne un tableau contenant les informations sur le champs numéro *field_number* après une requête de SELECT. Le tableau contient les index name (nom), alias, relation, length (taille), type.

```
<?php
// helio@helio.com.br  08-Dec-2000 02:53
$rs=ibase_query("Select * from unetable");
$coln = ibase_num_fields($rs);
for ($i=0 ; $i < $coln ; $i++) {
    $col_info = ibase_field_info($rs, $i);
    echo "nom: ".$col_info['name']."\n";
    echo "alias: ".$col_info['alias']."\n";
    echo "relation: ".$col_info['relation']."\n";
    echo "taille: ".$col_info['length']."\n";
    echo "type: ".$col_info['type']."\n";
    }
?>
```

ibase_free_result (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Libère un résultat.

```
int ibase_free_result (resource result_identifier)
```

ibase_free_result() libère les ressources liées au résultat *result_identifier*.

ibase_prepare (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Prépare une requête pour lier les paramètres et l'éxécuter ultérieurement.

```
int ibase_prepare ([resource link_identifier, string query])
```

ibase_prepare() prépare une requête pour l'éxécuter

ibase_execute (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Exécute une requête préparée.

```
resource ibase_execute (int query [, int bind_args])
```

ibase_execute() exécute une requête préparée (et éventuellement liée) par **ibase_prepare()**. **ibase_execute()** est beaucoup plus efficace que **ibase_query()**, si vous effectuez plusieurs fois la même requête, en ne changeant que quelques paramètres.

```
<?php
    $updates = array(
        1 => 'Eric',
        5 => 'Filip',
        7 => 'Larry'
    );
    $query = ibase_prepare("UPDATE FOO SET BAR = ? WHERE BAZ = ?");
    while (list($baz, $bar) = each($updates)) {
        ibase_execute($query, $bar, $baz);
    }
?>
```

ibase_trans (PHP 3>= 3.0.7, PHP 4 >= 4.0RC1)

Prépare une transaction

```
resource ibase_trans ([int trans_args [, resource link_identifier]])
```

ibase_trans() prépare une transaction

ibase_commit (PHP 3>= 3.0.7, PHP 4 >= 4.0RC1)

Valide une transaction

```
int ibase_commit ([resource link_identifier, resource trans_number])
```

ibase_commit() valide la transaction *trans_number*, qui a été préparée avec **ibase_trans()**.

ibase_rollback (PHP 3>= 3.0.7, PHP 4 >= 4.0RC1)

Annule une transaction

```
int ibase_rollback ([resource link_identifier, resource trans_number])
```

ibase_rollback() annule la transaction *trans_number* qui a été préparée avec **ibase_trans()**.

ibase_free_query (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Libère la mémoire réservée par une requête préparée.

```
int ibase_free_query (resource query)
```

ibase_free_query() libère la mémoire réservée par une requête préparée par **ibase_prepare()**.

ibase_timefmt (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Fixe le format de date pour les prochaines requêtes.

```
int ibase_timefmt (string format [, int columntype])
```

ibase_timefmt() fixe le format des colonnes de type dates, heure et timestamp, retournées par les requêtes. En interne, les colonnes sont formatées par la fonction C strftime() : reportez-vous à sa documentation pour connaître la structure de la chaîne de format. *columntype* est une des constantes suivantes : IBASE_TIMESTAMP, IBASE_DATE ou IBASE_TIME. Si elle est omise, la valeur par défaut est IBASE_TIMESTAMP, pour compatibilité ascendante.

```
<?php
    // Les colonnes TIME de InterBase 6 seront retournées avec
    // la forme '05 heures 37 minutes'.
    ibase_timefmt("%H heures %M minutes", IBASE_TIME);
?>
```

Vous pouvez aussi modifier les formats par défaut avec les directives PHP ibase.timestampformat, ibase.dateformat et ibase.timeformat.

Note : *columntype* a été ajouté en PHP 4.0. Il n'a aucun sens jusqu'à InterBase version 6 et plus récent.

Note : Une modification incompatible avec l'existant est apparue en PHP 4.0 lorsque la directive PHP ibase.timeformat a été renommée en ibase.timestampformat et les directives ibase.dateformat et ibase.timeformat ont été ajoutée, de manière à les adapter à leur fonction.

ibase_num_fields (PHP 3>= 3.0.7, PHP 4 >= 4.0RC1)

Retourne le nombre de colonnes dans un résultat.

```
int ibase_num_fields (resource result_id)
```

ibase_num_fields() retourne le nombre de colonnes dans un résultat.

```
<?php
    $dbh = ibase_connect ($host, $username, $password);
    $stmt = 'SELECT * FROM tblname';
    $sth = ibase_query($dbh, $stmt);
    if (ibase_num_fields($sth) > 0) {
        while ($row = ibase_fetch_object ($sth)) {
            print $row->email . "\n";
        }
    } else {
        die ("Aucun résultat dans votre requête");
    }
```

```
    ibase_close ($dbh);
?>
```

Note : ibase_timefmt() ne fonctionne pas encore sous PHP4.

ibase_errmsg (PHP 3>= 3.0.7, PHP 4 >= 4.0RC1)

Retourne un message d'erreur

```
string ibase_errmsg (void)
```

ibase_errmsg() retourne une chaîne contenant les messages d'erreurs.

XL. Ingres II

Ces fonctions permettent l'accès à un serveur de base de données Ingres II.

Pour pouvoir utiliser ces fonctions, vous devez compiler PHP avec le support Ingres, en utilisant l'option `--with-ingres`. Ceci nécessite les fichiers de bibliothèque de l'en-tête d'Open API qui sont inclus dans Ingres II. Si la variable d'environnement II_SYSTEM n'est pas correctement initialisée, vous devrez utiliser `--with-ingres=REP` pour spécifier le répertoire où a été installé Ingres.

Lorsque cette extension est utilisée avec Apache, si Apache ne démarre pas et émet l'erreur "PHP Fatal error: Unable to start ingres_ii module in Unknown on line 0", assurez-vous que la variable d'environnement II_SYSTEM est correctement initialisée. Il suffit souvent d'ajouter "export II_SYSTEM="/home/ingres/II" dans le script qui démarre Apache, juste avant le lancement de httpd.

> **Note :** Si vous avez déjà utilisé des extensions PHP permettant l'accès à d'autres serveurs de bases de données, notez qu'Ingres n'accepte pas de requêtes et/ou de transactions concurrentes sur la même connexion, et donc vous ne trouverez aucun identifiant de résultat ou de transaction dans cette extension. Le résultat d'une requête doit être traité avant d'envoyer une autre requête, et une transaction doit être validée ("commit") ou annulée ("roll back") avant de pouvoir en ouvrir une nouvelle (l'ouverture de transaction est fait automatiquement à l'envoi de la première requête).

ingres_connect (PHP 4 >= 4.0.2)

Ouvre une connexion à un serveur Ingres.

```
resource ingres_connect ([string database [, string username [, string password]]])
```

ingres_connect() retourne une ressource représentant un lien Ingres II en cas de succès, et FALSE sinon.

ingres_connect() établit une connexion avec la base de données Ingres désignée par *database*, qui suit la syntaxe *[node_id::]dbname[/svr_class]*.

Si certains paramétres sont manquants, **ingres_connect()** utilise les valeurs de *ingres.default_database*, *ingres.default_user* et *ingres.default_password* indiquées dans php.ini.

La connexion est fermée lorsque le script se termine ou en cas d'appel à **ingres_close()**.

Toutes les autres fonctions Ingres utilisent le dernier lien ouvert comme lien par défaut, il n'est donc nécessaire de conserver la valeur de retour qu'en cas d'utilisation de plus d'un lien en même temps.

Exemple 1. Exemple pour ingres_connect()

```
<?php
    $link = ingres_connect("mydb", "user", "pass")
        or die("Erreur de connexion");
    print("Connexion réussie");
    ingres_close($link);
?>
```

Exemple 2. Exemple pour ingres_connect() utilisant le lien par défaut

```
<?php
    ingres_connect("madb", "user", "pass")
        or die("Erreur de connexion");
    print("Connexion réussie");
    ingres_close();
?>
```

Voir aussi **ingres_pconnect()**, et **ingres_close()**.

ingres_pconnect (PHP 4 >= 4.0.2)

Ouvre une connexion persistante à un serveur Ingres.

```
resource ingres_pconnect ([string database [, string username [, string password]]])
```

ingres_pconnect() retourne une ressource représentant un lien (link) Ingres II en cas de succès, et FALSE sinon.

Voir **ingres_connect()** pour le détail des paramètres et des exemples. Il n'y a que 2 différences entre **ingres_pconnect()** et **ingres_connect()** : Tout d'abord, à la connexion, la fonction cherche un lien persistant déjà ouvert avec les mêmes paramètres. Si un tel lien est trouvé, un identificateur pour ce lien est retourné au lieu d'établir une nouvelle connexion. Ensuite, la connexion vers le serveur Ingres n'est pas fermée lorsque le script se termine. En fait, le lien reste ouvert pour pouvoir être réutilisé (**ingres_close()** ne ferme pas les liens établis avec **ingres_pconnect()**). C'est pourquoi ce type de lien est dit 'persistant'.

Voir aussi **ingres_connect()** et **ingres_close()**.

ingres_close (PHP 4 >= 4.0.2)

Ferme une connexion à un serveur Ingres.

```
boolean ingres_close ([resource link])
```

ingres_close() retourne TRUE en cas de succès, et FALSE sinon.

ingres_close() ferme la connexion au serveur Ingres associée au lien spécifié. Si le paramètre *link* n'est pas spécifié, le dernier lien ouvert est utilisé.

Habituellement l'appel à **ingres_close()** n'est pas nécessaire, car il ne ferme pas les connexions persistantes, et toutes les connexions non-persistantes sont automatiquement fermées à la fin du script.

Voir aussi **ingres_connect()** et **ingres_pconnect()**.

ingres_query (PHP 4 >= 4.0.2)

Envoie une requête SQL à un serveur Ingres II.

```
boolean ingres_query (string query [, resource link])
```

ingres_query() retourne TRUE en cas de succès, et FALSE sinon.

ingres_query() envoie la requête *query* au serveur Ingres. La requête doit être valide (voir le guide de référence SQL pour Ingres).

La requête s'ajoute à la transaction en cours. S'il n'y a pas de transaction ouverte, **ingres_query()** en ouvre une nouvelle. Pour fermer une transaction, vous pouvez soit appeler **ingres_commit()** pour valider les changements effectués sur la base de données ou **ingres_rollback()** pour les annuler. Lorsque le script se termine, toute transaction ouverte est annulée (par appel à **ingres_rollback()**). Vous pouvez aussi utiliser **ingres_autocommit()** avant d'ouvrir une transaction pour que chaque requête SQL soit validée immédiatement et automatiquement.

Certains types de requêtes SQL ne peuvent pas être envoyés par **ingres_query()** :

- CLOSE (voir **ingres_close()**).
- COMMIT (voir **ingres_commit()**).
- CONNECT (voir **ingres_connect()**).
- DISCONNECT (voir **ingres_close()**).
- get dbevent
- PREPARE TO COMMIT
- ROLLBACK (voir **ingres_rollback()**).
- savepoint
- SET AUTOCOMMIT (voir **ingres_autocommit()**).
- Les requêtes relatives aux curseurs ne sont pas supportées.

Exemple 1. Exemple pour ingres_query()

```
<?php
ingres_connect($database, $user, $password);
ingres_query("select * from table");
while ($row = ingres_fetch_row()) {
    echo $row[1];
    echo $row[2];
}
```

```
?>
```

Voir aussi **ingres_fetch_array()**, **ingres_fetch_object()**, **ingres_fetch_row()**, **ingres_commit()**, **ingres_rollback()** et **ingres_autocommit()**.

ingres_num_rows (PHP 4 >= 4.0.2)

Retourne le nombre de lignes affectées ou retournées par la dernière requête.

```
intingres_num_rows ([resource link])
```

Pour les requêtes DELETE, INSERT, UPDATE **ingres_num_rows()** retourne le nombre de lignes (tuples) affectées par la requête. Pour les autres requêtes, **ingres_num_rows()** retourne le nombre de lignes du résultat de la requête.

> **Note :** Cette fonction est conçue principalement pour obtenir le nombre de tuples modifiés dans la base de données. Si cette fonction est appelée avant d'utiliser **ingres_fetch_array()**, **ingres_fetch_object()** ou **ingres_fetch_row()**, le serveur efface les données du résultat et le script ne pourra plus les obtenir.
>
> Il faut dans ce cas récupérer les données du résultat en utilisant une de ces 3 fonctions dans une boucle jusqu'à ce qu'elle renvoie FALSE, ce qui indique qu'il n'y a plus de résultats à récupérer.

Voir aussi **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()** et **ingres_fetch_row()**.

ingres_num_fields (PHP 4 >= 4.0.2)

Retourne le nombre de champs renvoyés par la dernière requête.

```
intingres_num_fields ([resource link])
```

ingres_num_fields() retourne le nombre de champs du résultat renvoyé par le serveur Ingres après un appel à **ingres_query()**.

Voir aussi **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()** et **ingres_fetch_row()**.

ingres_field_name (PHP 4 >= 4.0.2)

Retourne le nom d'un champ dans le résultat d'une requête.

```
stringingres_field_name (int index [, resource link])
```

ingres_field_name() retourne le nom d'un champ dans le résultat d'une requête, ou FALSE en cas d'échec.

index est le numéro du champ et doit être compris entre 1 et la valeur donnée par **ingres_num_fields()**.

Voir aussi **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()** et **ingres_fetch_row()**.

ingres_field_type (PHP 4 >= 4.0.2)

Retourne le type d'un champ dans le résultat d'une requête.

```
stringingres_field_type (int index [, resource link])
```

ingres_field_type() retourne le type d'un champ dans le résultat d'une requête, ou FALSE en cas d'échec. Exemples de types renvoyés par cette fonction : "IIAPI_BYTE_TYPE", "IIAPI_CHA_TYPE", "IIAPI_DTE_TYPE", "IIAPI_FLT_TYPE", "IIAPI_INT_TYPE", "IIAPI_VCH_TYPE". Certains de ces types correspondent à plus d'un type SQL, selon la taille du champ (voir **ingres_field_length()**). Par exemple "IIAPI_FLT_TYPE" peut être un float4 ou un float8. Pour plus d'informations, voir le Guide de l'utilisateur d'Ingres/OpenAPI - Annexe C.

index est le numéro du champ et doit être compris entre 1 et la valeur donnée par **ingres_num_fields()**.

Voir aussi **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()** et **ingres_fetch_row()**.

ingres_field_nullable (PHP 4 >= 4.0.2)

Teste si un champ est annulable.

```
boolean ingres_field_nullable (int index [, resource link])
```

ingres_field_nullable() retourne TRUE si le champ peut recevoir la valeur NULL et FALSE dans le cas contraire.

index est le numéro du champ et doit être compris entre 1 et la valeur donnée par **ingres_num_fields()**.

Voir aussi **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()** et **ingres_fetch_row()**.

ingres_field_length (PHP 4 >= 4.0.2)

Retourne la taille d'un champ.

```
intingres_field_length (int index [, resource link])
```

ingres_field_length() retourne la taille d'un champ. Il s'agit du nombre d'octets utilisés par le serveur pour stocker ce champ. Pour plus d'informations, voir le Guide de l'utilisateur d'Ingres/OpenAPI - Annexe C.

index est le numéro du champ et doit être compris entre 1 et la valeur donnée par **ingres_num_fields()**.

Voir aussi **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()** et **ingres_fetch_row()**.

ingres_field_precision (PHP 4 >= 4.0.2)

Retourne la précision d'un champ.

```
intingres_field_precision (int index [, resource link])
```

ingres_field_precision() retourne la précision d'un champ. Cette valeur est utilisée uniquement pour les types de données SQL décimal, float et money. Pour plus d'informations, voir le Guide de l'utilisateur d'Ingres/OpenAPI - Annexe C.

index est le numéro du champ et doit être compris entre 1 et la valeur donnée par **ingres_num_fields()**.

Voir aussi **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()** et **ingres_fetch_row()**.

ingres_field_scale (PHP 4 >= 4.0.2)

Retourne l'échelle d'un champ.

```
intingres_field_scale (int index [, resource link])
```

ingres_field_scale() retourne l'échelle (scale) d'un champ. Cette valeur n'est utilisée que pour le type de données SQL décimal. Pour plus d'informations, voir le Guide de l'utilisateur d'Ingres/OpenAPI - Annexe C.

index est le numéro du champ et doit être compris entre 1 et la valeur donnée par **ingres_num_fields()**.

Voir aussi **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()** et **ingres_fetch_row()**.

ingres_fetch_array (PHP 4 >= 4.0.2)

Récupère une ligne de résultat dans un tableau.

```
arrayingres_fetch_array ([int result_type [, resource link]])
```

ingres_fetch_array() renvoie un tableau correspondant à la ligne récupérée, ou FALSE s'il n'y a plus de ligne à récupérer.

Cette fonction est une version améliorée de **ingres_fetch_row()**. En plus de stocker les données dans un tableau à indices numériques, elle peut aussi les enregistrer dans un tableau associatif, en utilisant les noms des champs comme indices.

Si plusieurs colonnes ont le même nom, la dernière colonne aura la priorité. Pour accéder aux autres colonnes du même nom, vous devez utiliser l'index numérique, ou faire un alias pour chaque colonne.

```
<?php
ingres_query(select t1.f1 as foo t2.f1 as bar from t1, t2);
$result = ingres_fetch_array();
$foo = $result["foo"];
$bar = $result["bar"];
?>
```

result_type peut valoir II_NUM pour un tableau à indices numériques, II_ASSOC pour un tableau associatif, ou II_BOTH (défaut) pour un tableau mixte (accessible selon les 2 méthodes).

Du point de vue de la rapidité, cette fonction est identique à **ingres_fetch_object()**, et presque aussi rapide que **ingres_fetch_row()** (la différence est insignifiante).

Exemple 1. Exemple pour ingres_fetch_array()

```
<?php
  ingres_connect($database, $user, $password);
  ingres_query("select * from table");
  while ($row = ingres_fetch_array()) {
    echo $row["user_id"]; // utilisation du tableau associatif
    echo $row["fullname"];
    echo $row[1];         // utilisation du tableau à indices numériques
    echo $row[2];
  }
?>
```

Voir aussi **ingres_query()**, **ingres_num_fields()**, **ingres_field_name()**, **ingres_fetch_object()** et **ingres_fetch_row()**.

ingres_fetch_row (PHP 4 >= 4.0.2)

Récupère une ligne de résultat dans un tableau énuméré.

```
array ingres_fetch_row ([resource link])
```

ingres_fetch_row() renvoie un tableau correspondant à la ligne récupérée, ou FALSE s'il n'y a plus de ligne à récupérer. La ligne est stockée dans un tableau à indices numériques, le premier champs étant à l'indice 1.

Les appels successifs à **ingres_fetch_row()** retournent les lignes suivantes du résultat, ou FALSE s'il n'y a plus de lignes.

Exemple 1. Exemple pour ingres_fetch_row()

```
<?php
ingres_connect($database, $user, $password);
ingres_query ("select * from table");
while ($row = ingres_fetch_row()) {
    echo $row[1];
    echo $row[2];
}
?>
```

Voir aussi **ingres_num_fields()**, **ingres_query()**, **ingres_fetch_array()** et **ingres_fetch_object()**.

ingres_fetch_object (PHP 4 >= 4.0.2)

Récupère une ligne de résultat dans un objet.

```
object ingres_fetch_object ([int result_type [, resource link]])
```

ingres_fetch_object() renvoie un objet correspondant à la ligne (tuple) récupérée, ou FALSE s'il n'y a plus de ligne à récupérer.

ingres_fetch_object() est similaire à **ingres_fetch_array()**, avec une différence : la valeur de retour est un objet et non un tableau. Indirectement, cela signifie qu'il n'est possible d'accéder aux données qu'avec les noms des champs, et pas avec leur numéro (les nombres ne sont pas des noms valides de propriété).

Le paramètre optionnel *result_type* est une constante qui peut prendre les valeurs II_ASSOC, II_NUM, et II_BOTH (par défaut).

Du point de vue de la rapidité, cette fonction est identique à **ingres_fetch_array()**, et presque aussi rapide que **ingres_fetch_row()** (la différence est insignifiante).

Exemple 1. Exemple pour ingres_fetch_object()

```
<?php
ingres_connect($database, $user, $password);
ingres_query("select * from table");
while ($row = ingres_fetch_object()) {
    echo $row->user_id;
    echo $row->fullname;
}
?>
```

Voir aussi **ingres_query()**, **ingres_num_fields()**, **ingres_field_name()**, **ingres_fetch_array()** et **ingres_fetch_row()**.

ingres_rollback (PHP 4 >= 4.0.2)

Annule une transaction.

```
boolean ingres_rollback ([resource link])
```

ingres_rollback() annule (roll back) la transaction ouverte, ce qui annule les modifications faites sur la base de données au cours de cette transaction.

Ceci ferme la transaction. Une nouvelle transaction peut être ouverte en envoyant une requête à l'aide de **ingres_query()**.

Voir aussi **ingres_query()**, **ingres_commit()** et **ingres_autocommit()**.

ingres_commit (PHP 4 >= 4.0.2)

Valide une transaction.

```
boolean ingres_commit ([resource link])
```

ingres_commit() valide la transaction ouverte, ce qui rend permanentes toutes les modifications faites sur la base de données au cours de cette transaction

Ceci ferme la transaction. Une nouvelle transaction peut être ouverte en envoyant une requête à l'aide de **ingres_query()**.

Vous pouvez aussi faire en sorte que le serveur valide automatiquement les changements après chaque requête en appelant **ingres_autocommit()** avant l'ouverture d'une transaction.

Voir aussi **ingres_query()**, **ingres_rollback()** et **ingres_autocommit()**.

ingres_autocommit (PHP 4 >= 4.0.2)

Active ou désactive le mode autocommit.

```
boolean ingres_autocommit ([resource link])
```

ingres_autocommit() est appelée juste avant l'ouverture d'une transaction (avant le premier appel à **ingres_query()** ou juste après un appel à **ingres_rollback()** ou **ingres_autocommit()**) pour changer l'état du mode "autocommit" (ce mode fonctionne comme une bascule). Lorsque le script débute le mode autocommit est toujours désactivé.

Lorsque le mode autocommit est activé, chaque requête est automatiquement commise (validée) par le serveur, comme si **ingres_commit()** était appelée après chaque appel à **ingres_query()**.

Voir aussi **ingres_query()**, **ingres_rollback()** et **ingres_commit()**.

XLI. IRC

Client pour IRC : Internet Relay Chat

Basé sur IRCG (http://php.net/~sas/schumann.cx/ircg/), de Sascha Schumann.

ircg_pconnect (PHP 4 >= 4.0.4)

Connecte à un serveur IRC

```
resource ircg_pconnect (string username [, string server_ip [, int server_port [, string
msg_format [, array ctcp_messages [, array user_settings]]]]])
```

ircg_pconnect() essaie d'établir une connexion avec le serveur IRC *server_ip*, et retourne une ressource de connexion pour utilisation ultérieure.

Le seul paramètre obligatoire est *username*, qui représente le nick (nom d'utilisateur en IRC) initial. *server_ip* et *server_port* sont optionnels, et par défaut, valent respectivement `127.0.0.1` (hôte local) et `6667`.

> **Note :** Actuellement, le paramètre *server_ip* n'effectue aucune résolution de nom, et n'accepte que les IP au format numérique.

Vous pouvez personnaliser l'affichage des messages IRC et les événements qui s'y rattachent avec les formats de messages, générés par la fonction **ircg_register_format_messages()**, en spécifiant le format dans *msg_format*.

ctcp_messages

user_settings

Voir aussi **ircg_disconnect()**, **ircg_is_conn_alive()** et **ircg_register_format_messages()**.

ircg_set_current (PHP 4 >= 4.0.4)

Prépare la connexion courante pour l'affichage

```
boolean ircg_set_current (resource connection)
```

ircg_set_current() sélectionne la connexion courante pour l'affichage dans le contexte d'exécution courant. Tous les messages envoyés par le serveur représenté par *connection* seront recopiés et envoyés à la sortie standard, avec le format standard, ou bien la chaîne de format spécifiée par la fonction **ircg_register_format_messages()** et créée par **ircg_lookup_format_messages()**.

Voir aussi **ircg_register_format_messages()** et **ircg_lookup_format_messages()**.

ircg_join (PHP 4 >= 4.0.4)

Rejoint un canal IRC

```
boolean ircg_join (resource connection, string channel)
```

ircg_join() rejoint le canal *channel* sur le serveur sur le serveur représenté par *connection*.

ircg_part (PHP 4 >= 4.0.4)

Quitte le canal

```
boolean ircg_part (resource connection, string channel)
```

ircg_part() quitte le canal *channel* sur le serveur sur le serveur représenté par *connection*.

ircg_msg (PHP 4 >= 4.0.4)

Envoie un message à un canal ou un utilisateur

```
boolean ircg_msg (resource connection, string recipient, string message [, boolean
suppress])
```

ircg_msg() envoie le message *message* à l'utilisateur ou au canal *recipient*, sur le serveur *connection*. Un destinataire *recipient* commençant par # ou & représente un canal IRC, et sinon, un utilisateur.

En donnant la valeur TRUE au paramètre *suppress*, vous éviterez que vos propres messages soient affichés sur votre connexion *connection*.

ircg_notice (PHP 4 >= 4.0.5)

Envoie une note (notice) à un utilisateur

```
boolean ircg_notice (resource connection, string , string message)
```

ircg_notice() envoie le message *message* à l'utilisateur *nick* sur le serveur *connection*. Consultez votre documentation IRC pour connaître la différence exacte entre un message MSG et une note NOTICE.

ircg_nick (PHP 4 >= 4.0.5)

Change de nom sur le serveur

```
boolean ircg_nick (resource connection, string nick)
```

ircg_nick() change le nom (nick) que vous portez sur la connexion *connection*, si ce nom n'est pas pris.

ircg_nick() retourne TRUE en cas de succès, et FALSE

ircg_topic (PHP 4 >= 4.0.5)

Modifie le sujet (topic) d'un canal

```
boolean ircg_topic (resource connection, string channel, string new_topic)
```

ircg_topic() change le sujet du canal de *channel* en *new_topic*, sur le serveur *connection*.

ircg_channel_mode (PHP 4 >= 4.0.5)

Modifie les flags du canal

```
boolean ircg_channel_mode (resource connection, string channel, string mode_spec, string
nick)
```

ircg_channel_mode() modifie les flags du canal *channel*, sur le serveur *channel*. Les nouveaux flags sont passés dans le paramètre *mode_spec* et sont appliqués à l'utilisateur *nick*.

Les flags sont activés ou désactivés en spécifiant un caractère de mode et en le préfixant par un caractère plus (+) ou un caractère moins (-). Par exemple, le mode opérateur est donné à un utilisateur avec la syntaxe '+o' et retiré au même utilisateur avec la syntaxe '-o', passé dans le paramètre *mode_spec*.

ircg_html_encode (PHP 4 >= 4.0.5)

Prépare l'affichage pour le HTML

```
boolean ircg_html_encode (string html_string)
```

...

Voir aussi:

ircg_whois (PHP 4 >= 4.0.5)

Requiert les informations sur un utilisateur

```
boolean ircg_whois (resource connection, string nick)
```

Voir aussi:

ircg_kick (PHP 4 >= 4.0.5)

Expulse un utilisateur d'un canal

```
boolean ircg_kick (resource connection, string channel, string nick, string reason)
```

ircg_kick() expulse l'utilisateur *nick* du canal *channel* sur le serveur *connection*. Le paramètre *reason* doit contenir une brève explication de la raison de cette expulsion.

ircg_ignore_add (PHP 4 >= 4.0.5)

Ajoute un utilisateur sur la liste des utilisateurs indésirables

```
boolean ircg_ignore_add (resource connection, string nick)
```

ircg_ignore_add() ajoute l'utilisateur *nick* dans votre liste d'utilisateurs indésirables, sur le serveur *connection*. Tous les messages qui vous sont envoyés par cet utilisateur seront ignorés.

Voir aussi **ircg_ignore_del()** .

ircg_ignore_del (PHP 4 >= 4.0.5)

Supprime un utilisateur de la liste des utilisateurs indésirables

```
boolean ircg_ignore_del (resource connection, string nick)
```

ircg_ignore_del() supprime l'utilisateur *nick* de la liste des utilisateurs indésirables, sur le serveur *connection*.

Voir aussi **ircg_ignore_add()**.

ircg_disconnect (PHP 4 >= 4.0.4)

Ferme la connexion avec un serveur

```
boolean ircg_disconnect (resource connection, string reason)
```

ircg_disconnect() ferme la connexion ouverter précédemment avec un serveur grâce à la fonction **ircg-pconnect()**, et représentée par *connection*.

Voir aussi **ircg_pconnect()**.

ircg_is_conn_alive (PHP 4 >= 4.0.5)

Vérifie l'état de la connexion

```
boolean ircg_is_conn_alive (resource connection)
```

ircg_is_conn_alive() retourne TRUE si la connexion *connection* est toujours active et fonctionnelle, et FALSE si la connexion n'est plus disponible.

ircg_lookup_format_messages (PHP 4 >= 4.0.5)

Sélectionne un format d'affichage pour les messages IRC

```
boolean ircg_lookup_format_messages (string name)
```

ircg_lookup_format_messages() sélectionne un format d'affichage pour les messages et les événements. Les formats peuvent être prédéfinis avec la fonction **ircg_register_format_messages()**. Un format par défaut, appelé `ircg` est toujours disponible.

Voir aussi **ircg_register_format_messages()**.

ircg_register_format_messages (PHP 4 >= 4.0.5)

Enregistre un nouveau format d'affichage des messages IRC

```
boolean ircg_register_format_messages (string name, array messages)
```

ircg_register_format_messages() vous permet de personnaliser l'affichage de vos messages IRC. Vous pouvez même enregistrer plusieurs formats et passer de l'un à l'autre à la volée avec **ircg_lookup_format_messages()**.

- Message public brut
- Message privé reçu
- Message privé envoyé
- Un utilisateur quitte le canal
- Un utilisateur rejoint le canal
- Un utilisateur est expulsé du canal

- Le sujet du canal est modifié
- Erreur
- Erreur fatale
- Rejoint la fin de la liste (??? : Join list end)
- Se quitte soi-même (??? : Self part)
- Un utilisateur s’expulse lui-même
- Un utilisateur quitte sa connexion
- Début de regroupement en masse
- Elément de regroupement en masse
- Fin de regroupement en masse
- Whois utilisateur
- Whois serveur
- Whois inactif
- Whois canal
- Fin de whois
- Changement de statut Voice pour un utilisateur
- Changement de statu d’opérateur pour un utilisateur
- Liste d’utilisateurs indésirables
- Fin de liste d’utilisateurs indésirables

- %f - origine
- %t - destination
- %c - canal
- %r - message brut
- %m - message encodé
- %j - message encodé js

- 1 - encodage mod
- 2 - nickname decode

Voir aussi **ircg_lookup_format_messages()**.

XLII. Java

Il y a deux moyens de connecter PHP et Java : soit en intégrant Java directement dans PHP, ce qui est la solution la plus stable et la plus efficace, ou en intégrant PHP dans un environnement de Servlet Java. La première solution est fournie par cette extension Java, et la dernière par le module SAPI qui s'interface avec un serveur de Servlet.

PHP 4 ext/java fournit un moyen simple mais efficace pour invoquer et créer des objets Java, depuis PHP. Une machine virtuelle est créée via JNI, et le tout fonctionne avec des processus fils. Les instructions d'installation pour cette extension sont sont disponibles dans le fichier : `php4/ext/java/README`.

Exemple 1. Exemple avec Java

```
<?php
  // crée une instance de Java class java.lang.System en PHP
  $system = new Java("java.lang.System");
  // accède aux propriétés
  print "Java version=".$system->getProperty("java.version")." <br>";
  print "Java vendor=" .$system->getProperty("java.vendor")."  <br>";
  print "OS=".$system->getProperty("os.name")." ".
              $system->getProperty("os.version")." on ".
              $system->getProperty("os.arch")." <br>";
  // Exemple avec java.util.Date
  $formatter = new Java("java.text.SimpleDateFormat",
                        "EEEE, MMMM dd, yyyy 'at' h:mm:ss a zzzz");
  print $formatter->format(new Java("java.util.Date"));
?>
```

Exemple 2. Exemple AWT

```
<?php
  // Cet exemple ne fonctionne qu'en mode CGI.
  $frame  = new Java("java.awt.Frame", "Zend");
  $button = new Java("java.awt.Button", "Bonjour au monde Java!");
  $frame->add("Nord", $button);
  $frame->validate();
  $frame->pack();
  $frame->visible = True;
  $thread = new Java("java.lang.Thread");
  $thread->sleep(10000);
  $frame->dispose();
?>
```

Notes:

- `new Java()` crée une nouvelle instance d'une classe, si un constructeur valable est disponible. Si aucun paramètre n'est passé, et le constructeur par défaut est utile pour accéder à ces classes telles que "java.lang.System", qui fournissent leur fonctionnalités via des méthodes statiques.
- Lors de l'accès aux membres d'une instance, PHP commencera par rechercher les membres Bean, puis les champs publics. En d'autres termes, "`print $date.time`" sera d'abord résolu par "`$date.getTime()`", puis par "`$date.time`";
- Les membres statiques et d'instance sont accessibles avec la même syntaxe. De plus, si un objet est de type "`java.lang.Class`", les membres statiques de la classe (champs et méthodes) sont accessibles.
- Les exceptions sont transformées en alertes PHP, et résultat NULL. Les alertes peuvent être supprimées en préfixant l'appel par l'opérateur "@". Les fonctions suivantes peuvent être utilisées pour lire et effacer la dernière erreur remontée :
  - **java_last_exception_get()**
  - **java_last_exception_clear()**

- Les surchargements de fonctions sont des problèmes épineux, étant donné les différences de type de valeurs entre les deux langages. L'extension Java de PHP utilise une métrique simple mais efficace pour déterminer la meilleur fonction à utiliser.

  De plus, les noms de méthodes ne sont pas sensibles à la casse en PHP, ce qui augmente le nombre de conflits potentiels.

  Une fois qu'une méthode est sélectionnée, les paramètres sont transtypés, avec une perte d'information potentielle non négligeable (par exemple, les nombres à virgules flottante en double précisions seront convertis en booléen).

- Traditionnellement en PHP, les tableaux et les tables de hashage peuvent être interchangées, et fonctionnent de la même façon. Notez que les tables de hashage de PHP ne peuvent être indexées qu'avec des entiers ou des chaînes, et que le type primitif de tableau de Java ne peut comporter de trous dans les index. Notez aussi que les valeurs sont passées par valeur, ce qui peut être coÛteux en mémoire et en temps.

L'interface PHP4 sapi/servlet est construite sur un mécanisme défini par l'extension Java, qui permet à PHP d'être exécuté comme une servlet. L'avantage immédiat d'un point de vue PHP est que les serveurs web qui supportent les servlets gèrent rigoureusement les machines virtuelles. Les instructions d'installation du module Servlet SAPI sont disponibles dans le fichier `php4/sapi/README`. Notes:

- Bien que ce code soit prévu pour fonctionner sur n'importe quel serveur à Servlet, il n'a été testé qu'avec le module Apache Jakarta/tomcat (jusqu'à aujourd'hui). Les remontées de bugs, les réussites et les patches nécessaires pour faire fonctionner ce code sur d'autres serveurs sont fortement encouragés!
- PHP a l'habitude de changer le dossier de travail. Le serveur SAPI/Servlet le changera à nouveau, mais tant que PHP fonctionnera, le moteur de servlet ne pourra pas charger de classes dans le CLASSPATH, si le dossier est spécifié avec un chemin relatif, ou ne pourra pas trouver le dossier d'administration et de compilation des tâches JSP.

java_last_exception_clear (PHP 4 >= 4.0.2)

Efface la dernière exception Java

```
void java_last_exception_clear ()
```

java_last_exception_get (PHP 4 >= 4.0.2)

Lit la dernière exception Java

```
exception java_last_exception_get ()
```

L’exemple ci-dessous montre l’utilisation du gestionnaire d’exceptions java :

Exemple 1. Gestionnaire d’exception Java

```
<?php
  $stack = new Java("java.util.Stack");
  $stack->push(1);
  // Cela doit marcher
  $result = $stack->pop();
  $ex = java_last_exception_get();
  if (!$ex) print "$result\n";
  // Cela doit échouer (le rapport d’erreurs est supprimé par >)
  $result = @$stack->pop();
  $ex = java_last_exception_get();
  if ($ex) print $ex->toString();
  // Efface la dernière exception
  java_last_exception_clear();
?>
```

XLIII. LDAP

Introduction à LDAP

LDAP est l'acronyme de Lightweight Directory Access Protocol, c'est à dire Protocole Léger d'Accès aux Dossiers. C'est un protocole utilisé pour accéder à des "serveurs de dossiers", des serveurs qui gèrent les informations de manière hiérarchique.

Le concept est similaire à la structure de votre disque dur, hormis le fait que la racine s'appelle ici : "The world" (le monde), et que les dossiers du premier niveau sont assimilés à des pays. Les niveaux inférieurs de la structure contiennent des entrées de sociétés, d'organisations ou de lieux tandis que les niveaux encore inférieurs sont des gens, voire des équipements ou des documents.

Pour accéder à un fichier sur votre disque, vous devez utiliser la syntaxe suivante :

```
/usr/local/myapp/docs
```

Le slash indique une division de la référence, et la séquence est lue de gauche à droite.

Avec tous les détails, une référence LDAP s'appelle un "nom distingué" ("distinguished name"), appelé aussi "nd" ("dn" en anglais). Par exemple :

```
cn=Jean Dupont,ou=Comptes,o=Ma Société,c=Fr
```

La virgule marque une division de la référénce, et la séquence est lue de droite à gauche. Vous pouvez la lire comme ceci :

```
country = Fr
organization = Ma Société
organizationalUnit = Comptes
commonName = Jean Dupont
```

De la même façon qu'il n'y a pas de règle universelle d'organisation d'un disque dur, un serveur de dossier peut supporter n'importe quelle structure du moment qu'elle a un sens pour ce qu'on en fait. Cependant, il existe quelques conventions : il est impossible d'écrire un code d'accès à un dossier sans en connaître sa structure, de la même façon que vous ne pouvez pas utiliser une base de données sans en connaître les tables.

Exemple complet

Recupérer toutes les entrées dont le nom commence par "S" dans un serveur, et afficher le nom et l'adresse email.

Exemple 1. Recherche LDAP

```
<?php
// Structure d'une commande simple :
// connexion, lien, recherche, interpretation de la recherche
// résultat, déconnexion
  echo "<?h3>LDAP query test<?/h3>";
  echo "Connexion ...";
  $ds=ldap_connect("localhost");  // Doit être un serveur LDAP valide!
  echo "Résultat de la connexion : ".$ds."<?p>";
  if ($ds) {
    echo "Lien ...";
    $r=ldap_bind($ds);     // Ceci est un lien "anonymous", typiquement
                           // en lecture seule. En cas d'accès, affiche
                           // " Lien résultat est"
    echo "Lien résultat est ".$r."<?p>";
    echo "Recherche de (sn=S*) ...";
    // Recherche dans les noms
    $sr=ldap_search($ds,"o=Ma Société, c=Fr", "sn=S*");
    echo "Résultat : ".$sr."<?p>";
    echo "Nombre d'entrée retournée : ".ldap_count_entries($ds,$sr)."<?p>";
```

```
    echo "Lecture des entrées...<?p>";
    $info = ldap_get_entries($ds, $sr);
    echo "Data for ".$info["count"]." items returned:<?p>";
    for ($i=0; $i<?$info["count"]; $i++) {
        echo "dn vaut : ". $info[$i]["dn"] ."<?br>";
        echo "première entrée cn vaut : ". $info[$i]["cn"][0] ."<?br>";
        echo "premièr email vaut: ". $info[$i]["mail"][0] ."<?p>";
    }
    echo "Déconnexion ";
    ldap_close($ds);
  } else {
    echo "<?h4>Impossible de se connecter à un serveur LDAP <?/h4>";
  }
?>
```

Utilisation des fonctions PHP LDAP

Il faut d'abord que les bibliothèques client LDAP soient compilées avec PHP. Vous pouvez vous procurer ces bibliothèques University of Michigan (ldap-3.3 package) ou chez Netscape (Netscape Directory SDK).

Avant d'utiliser les fonctions LDAP il faut savoir :

- Le nom ou l'adresse du serveur à utiliser
- Le "nd" dans le serveur (la partie du monde qui est sur ce serveur, ce qui peut correspondre à "o=Ma société,c=Fr")
- Eventuellement, un mot de passe pour accéder au serveur (de nombreux serveusr fournissent un accès anonyme ("anonymous bind") mais requièrent un mot de passe pour tous les autres).

Une séquence habituelle LDAP suivra le schéma suivant :

```
ldap_connect()   // établit une connexion à un serveur
  |
ldap_bind()      // nom de compte "login" ou anonyme
  |
éxécution de commandes sur le serveur, comme un listage, ou
une modification de données avec affichage
  |
ldap_close()     // "déconnexion"
```

Plus d'informations

Vous pouvez en apprendre encore plus, mais en anglais, aux adresses suivantes :

- Netscape (http://developer.netscape.com/tech/directory/)
- University of Michigan (http://www.umich.edu/~dirsvcs/ldap/index.html)
- OpenLDAP Project (http://www.openldap.org/)
- LDAP World (http://elvira.innosoft.com/ldapworld)

Le SDK Netscape contient un guide du programmeur au format HTML particulièrement pratique (en anglais).

ldap_add (PHP 3, PHP 4 >= 4.0b1)

Ajoute une entrée à un dossier LDAP.

```
int ldap_add (resource link_identifier, string dn, array entry)
```

ldap_add() retourne TRUE en cas de succès, ou FALSE en cas d'erreur.

ldap_add() sert à ajouter une entrée dans un dossier LDAP. Le ND de l'entrée sera ajouté à la *dn* du dossier spécifié. Le tableau *entry* spécifie les informations de la nouvelle entrée. Les valeurs de l'entrée sont indexées dans les attributs de l'entrée. Si un attribut a de multiples valeurs, elles seront indexées dans un tableau, à partir de l'index 0.

```
entree["attribut1"] = valeur
entree["attribut2"][0] = valeur1
entree["attribut2"][1] = valeur2
```

Exemple 1. Exemple complet avec lien authentifié

```
<?php
  $ds=ldap_connect("localhost");
// On suppose que le serveur LDAP est sur cet hôte
  if ($ds) {
    // liaison avec le nd approprié, pour avoir un accès en modification
    $r=ldap_bind($ds,"cn=root, o=Ma Société, c=Fr", "secret");
    // preparation des données
    $info["cn"]="John Jones";
    $info["sn"]="Jones";
    $info["mail"]="jonj@here.and.now";
    $info["objectclass"]="person";
    // Ajout des données dans le dossier
    $r=ldap_add($ds, "cn=John Jones, o=My Company, c=US", $info);
    ldap_close($ds);
  } else {
    echo "Impossible de se connecter au serveur LDAP ";
  }
?>
```

ldap_bind (PHP 3, PHP 4 >= 4.0b1)

Se lie à un serveur LDAP.

```
int ldap_bind (resource link_identifier [, string bind_rdn [, string bind_password]])
```

ldap_bind() lie un serveur LDAP avec le RDN *bind_rdn* et mot de passe *bind_password*. **ldap_bind()** retourne TRUE en cas de succès, et FALSE sinon.

ldap_bind() effectue une opération de liaison sur le serveur. *bind_rdn* et *bind_password* sont optionnels. S'ils manquent, la liaison se fera en mode anonyme.

ldap_close (PHP 3, PHP 4 >= 4.0b1)

Déconnecte d'un serveur LDAP.

```
int ldap_close (resource link_identifier)
```

ldap_close() retourne `TRUE` en cas de succès, et `FALSE` sinon.

ldap_close() ferme le lien au serveur LDAP associé à l'identifiant *`link_identifier`*.

Cet appel est identique à **ldap_unbind()**, en interne. Les API LDAP utilisent la fonction **ldap_unbind()** : il est probablement mieux que vous utilisiez cette fonction là plutôt que **ldap_close()**.

ldap_compare (PHP 4 >= 4.0.2)

Compare les valeurs des attributs trouvés dans un ND

```
int ldap_compare (resource link_identifier, string dn, string attribute, string value)
```

ldap_compare() retournr `TRUE` si *`value`* un fichier correspond à la recherche; retourne -1 si une erreur survient.

ldap_compare() sert à comparer la valeur *`value`* de l'attribut *`attribute`* aux valeurs du même atribut dans l'annuaire LDAP *dn*.

L'exemple suivant montre comment vérifier qu'un mot de passe correspond bien à celui qui est stocké dans l'annuaire.

Exemple 1. Vérification d'un mot de passe avec LDAP

```
<?php
  $ds=ldap_connect("localhost");
// on suppose que le serveur LDAP est sur le serveur local
  if ($ds) {
    // liaison
    if(ldap_bind($ds)) {
        // prépare les données
        $dn = "cn=Matti Meikku, ou=My Unit, o=My Company, c=FI";
        $value = "secretpassword";
        $attr = "password";
        // compare les valeurs
        $r=ldap_compare($ds, $dn, $attr, $value);
        if ($r === -1) {
            echo "Erreur: ".ldap_error($ds);
        } elseif ($r === TRUE) {
            echo "Mot de passe correct.";
        } elseif ($r === FALSE) {
            echo "Mot de passe erroné!";
        }
    } else {
        echo "Connexion impossible.";
    }
    ldap_close($ds);
  } else {
    echo "Impossible de se connecter au serveur LDAP.";
  }
?>
```

Note : ldap_compare() ne peut pas comparer des données binaires.

Note : Cette fonction a été ajoutée en 4.0.2.

ldap_connect (PHP 3, PHP 4 >= 4.0b1)

Se connecte à un serveur LDAP.

```
resource ldap_connect ([string hostname [, int port]])
```

ldap_connect() retourne un identifiant positif de serveur LDAP en cas de succès, ou bien FALSE en cas d'erreur.

ldap_connect() établit une connexion avec un serveur. Le serveur LDAP situé sur l'hôte *hostname* et *port*. Les deux arguments sont optionnels. Sans argument, l'identifiant de la dernière connexion ouverte sera retournée. Si seul *hostname* est spécifié, le port par défaut est 389.

Si vous utilisez OpenLDAP 2.x.x, vous pouvez spécifier une URL au lieu d'un nom d'hôte. Pour utiliser LDAP avec SSL, compilez OpenLDAP 2.x.x avec le support SSL, configurez PHP avec SSL, et utilisez ldaps://hostname/ comme nom d'hôte. Le paramètre de port *port* n'est pas utile lorsqu'utilisé avec des URL. Le support des URL et SSL a été ajouté en PHP 4.0.4.

ldap_count_entries (PHP 3, PHP 4 >= 4.0b1)

Compte le nombre d'entrées d'une recherche.

```
int ldap_count_entries (resource link_identifier, resource result_identifier)
```

ldap_count_entries() retourne le nombre d'entrées en cas de succès, et FALSE sinon.

ldap_count_entries() retourne le nombre d'entrées placées dans le résultat par les recherches précédentes. *result_identifier* identifie un résultat LDAP interne.

ldap_delete (PHP 3, PHP 4 >= 4.0b1)

Efface une entrée dans un dossier.

```
int ldap_delete (resource link_identifier, string dn)
```

ldap_delete() retourne TRUE en cas de succès, et FALSE sinon.

ldap_delete() efface une entrée dans un dossier LDAP spécifié par le nd *dn*.

ldap_dn2ufn (PHP 3, PHP 4 >= 4.0b1)

Convertit un ND dans un format plus accessible.

```
string ldap_dn2ufn (string dn)
```

ldap_dn2ufn() sert à mettre le nd *dn* dans un format plus agréable, notamment en supprimant les noms des types.

ldap_err2str (PHP 3>= 3.0.13, PHP 4 >= 4.0RC2)

Convertit un numéro d'erreur LDAP en message d'erreur.

```
string ldap_err2str (int errno)
```

ldap_err2str() retourne un message d'erreur.

ldap_err2str() retourne le message d'erreur lié au numér d'erreur *errno*. Même si les numéros d'erreur LDAP sont standardisés, différentes librairies retournent différents messages, ou parfois, des messages en langue locale. Ne vous fiez pas au message d'erreur, mais bien au numéro d'erreur.

Voir aussi **ldap_errno()** et **ldap_error()**.

Exemple 1. Enumérer tous les messages d'erreur LDAP

```
<?php
  for($i=0; $i<100; $i++) {
    printf("Error $i: %s<br>\n", ldap_str2err($i));
  }
?>
```

ldap_errno (PHP 3>= 3.0.12, PHP 4 >= 4.0RC2)

Retourne le numéro d'erreur LDAP de la dernière commande exécutée.

```
int ldap_errno (resource link_id)
```

ldap_errno() retourne le numéro d'erreur LDAP généré par la dernière commande.

ldap_errno() retourne le numéro d'erreur standard, généré par la dernière commande LDAP, pour la connexion *link_id*. Ce numéro peut être converti en message textuel avec **ldap_err2str()**.

A moins que vous n'abaissiez suffisamment le niveau d'erreur dans `php.ini` (ou `php3.ini`), ou que vous ne préfixiez vos commandes LDAP avec `@` (at) pour supprimer les affichages, les erreurs LDAP s'afficheront aussi dans le code PHP.

Exemple 1. Genérer et intercepter une erreur

```
<?php
// Cet exemple contient une erreur, que nous allons intercepter.
  $ld = ldap_connect("localhost");
  $bind = ldap_bind($ld);
// Erreur de syntaxe dans l'expression du filtre (errno 87),
// ce doit être "objectclass=*"
  $res =  @ldap_search($ld, "o=Myorg, c=DE", "objectclass");
  if (!$res) {
    printf("LDAP-Errno: %s<br>\n", ldap_errno($ld));
    printf("LDAP-Error: %s<br>\n", ldap_error($ld));
    die("Argh!<br>\n");
  }
  $info = ldap_get_entries($ld, $res);
  printf("%d entrées trouvées.<br>\n", $info["count"]);
?>
```

Voir aussi **ldap_err2str()** et **ldap_error()**.

ldap_error (PHP 3>= 3.0.12, PHP 4 >= 4.0RC2)

Retourne le message LDAP de la dernière commande LDAP.

```
string ldap_error (resource link_id)
```

ldap_error() retourne un message d'erreur.

ldap_error() retourne le message d'erreur lié à la connexion *link_id*. Même si les numéros d'erreur LDAP sont standardisés, différentes librairies retournent différents messages, ou parfois, des messages en langue locale. Ne vous fiez pas au message d'erreur, mais bien au numéro d'erreur.

A moins que vous n'abaissiez suffisamment le niveau d'erreur dans `php.ini` (ou `php3.ini`), ou que vous ne préfixiez vos commandes LDAP avec `@` pour supprimer les affichages, les erreurs LDAP s'afficheront aussi dans le code PHP.

Voir aussi **ldap_err2str()** et **ldap_errno()**.

ldap_explode_dn (PHP 3, PHP 4 >= 4.0b1)

Scinde un ND en plusieurs composants.

```
array ldap_explode_dn (string dn, int with_attrib)
```

ldap_explode_dn() sert à scinder le nd *dn* retourné par **ldap_get_dn()** en plusieurs composants. Chaque composant est reconnu sous le nom Nom Distinct Relatif (ou RDN, en anglais). **ldap_explode_dn()** retourne un tableau qui contient ces composants. *with_attrib* sert à préciser si le RDN est retourné avec ses attributs, ou seul. Pour obtenir le RDN et ses attributs, mettez *with_attrib* à 0 et pour n'avoir que les valeurs, mettez le à 1.

ldap_first_attribute (PHP 3, PHP 4 >= 4.0b1)

Retourne le premier attribut.

```
string ldap_first_attribute (resource link_identifier, resource result_entry_identifier,
int ber_identifier)
```

ldap_first_attribute() retourne le premier attribut en cas de succès, et `FALSE` sinon.

Le comportement est similaire pour les entrées. Les attributs sont lus séquentiellement dans une entrée particulière. **ldap_first_attribute()** retourne le premier attribut de l'entrée désignée par l'identifiant d'entrée. Les attributs suivants sont accessibles avec **ldap_next_attribute()**. *ber_identifier* est un identifiant de pointeur de mémoire interne. Il est passé par référence. Le même identifiant *ber_identifier* est passé à **ldap_next_attribute()**, qui modifie ce pointeur.

Voir aussi **ldap_get_attributes()**.

ldap_first_entry (PHP 3, PHP 4 >= 4.0b1)

Retourne l'identifiant du premier attribut.

```
int ldap_first_entry (resource link_identifier, resource result_identifier)
```

ldap_first_entry() retourne un identifiant sur la première entrée en cas de succès, et `FALSE` sinon.

Les entrées d'un résultat sont lues séquentiellement, en utilisant **ldap_first_entry()** et **ldap_next_entry()**. **ldap_first_entry()** retourne l'identifiant de la première entrée du résultat. Cet identifiant sera fourni à **ldap_next_entry()** pour accéder à la prochaine entrée.

Voir aussi **ldap_get_entries()**.

ldap_free_result (PHP 3, PHP 4 >= 4.0b1)

Libère la mémoire prise par un résultat.

```
int ldap_free_result (resource result_identifier)
```

ldap_free_result() retourne TRUE en cas de succès, et FALSE sinon.

ldap_free_result() libère la mémoire allouée en interne pour enregistrer le résultat pointé par *result_identifier*. A la fin de chaque script, la mémoire sera de toute manière libérée.

Généralement, il n'y a pas besoin de libérer la mémoire, et le mécanisme automatique de fin de script est suffisant. Cependant, dans les cas où le script effectue plusieurs recherches successives, où que les résultats retournés sont très grands, **ldap_free_result()** permet de réduire la consommation de mémoire.

ldap_get_attributes (PHP 3, PHP 4 >= 4.0b1)

Retourne les attributs d'une entrée d'un résultat.

```
array ldap_get_attributes (resource link_identifier, resource result_entry_identifier)
```

ldap_get_attributes() retourne un tableau multi-dimensionel en cas de succès, et FALSE sinon.

ldap_get_attributes() sert à simplifier la lecture des attributs et des valeurs d'une entrée dans un résultat. Le résultat est un tableau multi-dimensionnel, avec les attributs en clé, et les valeurs des attributs en valeurs.

Une fois que vous avez repéré une entré dans un dossier, vous pouvez lire les informations de cette entrée avec cette fonction. Vous pouvez utiliser cette fonction pour créer une application qui se déplace dans les dossiers, sans en connaître la structure au préalable. Dans de nombreux cas, vous ne chercherez qu'un attribut particulier (le email, par exemple) et vous ne vous intéresserez pas aux autres valeurs.

Exemple 1. Affichage de la liste des attributs d'une entrée

```
<?php
// $ds est l'identifiant de lien pour ce dossier
// $sr est un résultat de recherche valide, obtenu lors d'une recherche
// précédente
  $entry = ldap_first_entry($ds, $sr);
  $attrs = ldap_get_attributes($ds, $entry);
  echo $attrs["count"]." Attributs dans cette entrée:<p>";
  for ($i=0; $i<$attrs["compte"]; $i++)
    echo $attrs[$i]."<br>";
?>
```

Voir aussi **ldap_first_attribute()** et **ldap_next_attribute()**.

ldap_get_dn (PHP 3, PHP 4 >= 4.0b1)

Retourne un ND d'une entrée d'un résultat.

```
string ldap_get_dn (resource link_identifier, resource result_entry_identifier)
```

ldap_get_dn() retourne le DN de l'entrée en cas de succès, et FALSE sinon.

ldap_get_dn() sert à obtenir le ND d'une entrée d'un résultat.

ldap_get_entries (PHP 3, PHP 4 >= 4.0b1)

Retourne toutes les entrées.

```
array ldap_get_entries (resource link_identifier, resource result_identifier)
```

ldap_get_entries() retourne un tableau multi-dimensionnel en cas de succès, et FALSE sinon.

ldap_get_entries() sert à simplifier la lecture d'un résultat à plusieurs entrées. Toutes les informations sont retournées sous la forme d'un tableau multi-dimensionnel. La structure de ce tableau est la suivante :

Les attributs servent d'index et sont mis en minuscules (les attributs sont insensibles à la casse sur les serveurs, mais peuvent ne pas l'être quand ils sont utilisés comme index)

résultat ["compte"] = nombre d'entrées du résultat
résultat [0] : correspond aux détails de la première entrée :
résultat [i]["nd"] = ND de la i-ième entrée
résultat [i][" compte "] = nombre d'attributs de la i-ième entrée
résultat [i][j] = j-ième attribut de la i-ième entrée
résultat [i]["attribut"]["count"] = nombre de valeur pour l'attribut
résultat [i]["attribut"][j] = j-ième valeur de l'attribut

Voir aussi **ldap_first_entry()** et **ldap_next_entry()**.

ldap_get_option (PHP 4 >= 4.0.4)

Lit la valeur courante d'une option

```
boolean ldap_get_option (resource link_identifier, int option, mixed retval)
```

ldap_get_option() remplace la valeur courante de l'option *option* par *retval*, et retourne TRUE en cas de succès, FALSE sinon.

Le paramètre *option* peut prendre l'une des valeurs : LDAP_OPT_DEREF, LDAP_OPT_SIZELIMIT, LDAP_OPT_TIMELIMIT, LDAP_OPT_PROTOCOL_VERSION, LDAP_OPT_ERROR_NUMBER, LDAP_OPT_REFERRALS, LDAP_OPT_RESTART, LDAP_OPT_HOST_NAME, LDAP_OPT_ERROR_STRING, LDAP_OPT_MATCHED_DN. Elles sont décrites dans draft-ietf-ldapext-ldap-c-api-xx.txt (http://www.openldap.org/devel/cvsweb.cgi/~checkout~/doc/drafts/draft-ietf-ldapext-ldap-c-api-xx.txt)

ldap_get_option() n'est disponible que si vous utilisez OpenLDAP 2.x.x ou Netscape Directory SDK x.x. Elle a été ajoutée dans PHP 4.0.4.

Exemple 1. Vérification de la version du protocole

```
<?php
// $ds est un identifiant valide d'annuaire
  if (ldap_get_option($ds, LDAP_OPT_PROTOCOL_VERSION, $version))
    echo "La version du protocole est $version";
  else
    echo "Impossible de lire la version du protocole";
?>
```

Voir aussi **ldap_set_option()**.

ldap_get_values (PHP 3, PHP 4 >= 4.0b1)

Retourne toutes les entrées d’un résultat.

```
array ldap_get_values (resource link_identifier, resource result_entry_identifier, string attribute)
```

ldap_get_option() retourne un tableau de valeurs en cas de succès, et FALSE sinon.

ldap_get_values() sert à lire toutes les valeurs d’un attribut dans une entrée. L’entrée est référencée par *result_entry_identifier*. Le nombre de valeurs peut être trouvé à l’index "count" dans le résultat. Les valeurs sont accessibles par un index entier, qui commence à 0.

ldap_get_values() nécessite un pointeur de résultat *result_entry_identifier*, ce qui implique qu’il ait été précédé d’une recherche sur le serveur, et de l’obtention d’une entrée.

Votre application pourra utiliser des noms d’attributs en dur dans le code, ou bien, utiliser la fonction **ldap_get_attributes()** pour y accéder dynamiquement.

LDAP autorise plus d’une entrée par attribut, ce qui permet, par exemple, d’étiqueter tous les adresses email d’un utilisateur avec l’attribut "mail"

return_value["count"] = nombre de valeurs de l’attribut
return_value[0] = première valeur de l’attribut
return_value[i] = n-ième valeur de attribut

Exemple 1. Liste toutes les valeurs avec l’attribut "mail"

```
<?php
// $ds est l’identifiant de lien pour ce dossier
// $sr est un résultat de recherche valide, obtenu lors d’une recherche
// précédente
// $entry est un identifiant valide d’entrée
  $values = ldap_get_values($ds, $entry,"mail");
  echo $values["count"]." Adresse email dans ce résultat.<p>";
  for ($i=0; $i < $values["count"]; $i++)
    echo $values[$i]."<br>";
?>
```

ldap_get_values_len (PHP 3>= 3.0.13, PHP 4 >= 4.0RC2)

Retourne toutes les valeurs binaires à partir d’un identifiant de résultat.

```
array ldap_get_values_len (resource link_identifier, resource result_entry_identifier, string attribute)
```

ldap_get_values_len() retourne un tableau de valeurs pour l’attribut, ou bien FALSE en cas d’erreur.

ldap_get_values_len() sert à lire toutes les valeurs d’un attribut d’une entrée dans un résultat. L’entrée est spécifiée par *result_entry_identifier*. Le nombre de valeurs trouvées peut être retrouvé en indexant "count" dans le tableau résultat. On peut accéder aux valeurs individuelles avec un index numérique, commençant à 0.

ldap_get_values_len() est utilisée exactement comme **ldap_get_values()** mais elle gère les données binaires, et non pas les chaînes.

ldap_list (PHP 3, PHP 4 >= 4.0b1)

Recherche dans un seul niveau.

```
boolean ldap_list (resource link_identifier, string base_dn, string filter [, array
attributes [, int attrsonly [, int sizelimit [, int timelimit [, int deref]]]]])
```

ldap_list() retourne TRUE en cas de succès, et FALSE sinon.

ldap_list() effecture une recherche avec le filtre7 donnée, et limité un dossier.

LDAP_SCOPE_ONELEVEL indique que la recherche ne doit s'étendre que dans le dossier immédiatement sous le nd *base_dn*. (Equivalent à taper "ls" et obtenir la liste des fichiers et dossiers du dossier courant).

Cette appel prend un quatrième argument optionnel : un tableau contenant les attributs recherchés. Reportez-vous à **ldap_search()** pour plus de détails.

Exemple 1. Affiche une liste d'unités organisationnelle

```
<?php
// $ds est un identifiant de connexion valide.
  $basedn = "o=Ma Société, c=Fr";
  $justthese = array("ou");
  $sr=ldap_list($ds, $basedn, "ou=*", $justthese);
  $info = ldap_get_entries($ds, $sr);
  for ($i=0; $i<$info["count"]; $i++)
    echo $info[$i]["ou"][0] ;
?>
```

ldap_modify (PHP 3, PHP 4 >= 4.0b1)

Modifie une entrée LDAP.

```
int ldap_modify (resource link_identifier, string dn, array entry)
```

ldap_modify() retourne TRUE en cas de succès, et FALSE sinon.

ldap_modify() sert à modifier les entrées existantes dans un dossier LDAP. La structure de l'entrée est la même que décrite dans **ldap_add()**.

ldap_mod_add (PHP 3>= 3.0.8, PHP 4 >= 4.0b1)

Ajoute un attribut

```
resource ldap_mod_add (int link_identifier, string dn, array entry)
```

ldap_mod_add() retourne TRUE en cas de succès, et FALSE sinon.

ldap_mod_add() ajoute les attributs *entry* à l'entrée *dn*. La modification s'applique au niveau local (par opposition au niveau objet). Les ajouts au niveau de l'objet sont pris en charge par **ldap_add()**.

ldap_mod_del (PHP 3>= 3.0.8, PHP 4 >= 4.0b1)

Efface un attribut

```
int ldap_mod_del (resource link_identifier, string dn, array entry)
```

ldap_mod_del() retourne TRUE en cas de succès, et FALSE sinon.

ldap_mod_del() efface les attributs *entry* à l'entrée *dn*. La modification s'applique au niveau local (par opposition au niveau objet). Les ajouts au niveau de l'objet sont pris en charge par **ldap_delete()**.

ldap_mod_replace (PHP 3>= 3.0.8, PHP 4 >= 4.0b1)

Remplace un attribut

```
int ldap_mod_replace (resource link_identifier, string dn, array entry)
```

ldap_mod_replace() retourne TRUE en cas de succès, et FALSE sinon.

ldap_mod_replace() remplace les attributs *entry* à l'entrée *dn*. La modification s'applique au niveau local (par opposition au niveau objet). Les ajouts au niveau de l'objet sont pris en charge par **ldap_modify()** .

ldap_next_attribute (PHP 3, PHP 4 >= 4.0b1)

Lit l'attribut suivant.

```
string ldap_next_attribute (resource link_identifier, resource result_entry_identifier,
int ber_identifier)
```

ldap_next_attribute() retourne l'attribut suivant en cas de succès, et sinon, une erreur.

ldap_next_attribute() sert à lire tous les attributs d'une entrée. Le pointeur interne est géré par *ber_identifier*. Il est passé par référence à la fonction. Le premier appel à **ldap_next_attribute()** est fait avec le *result_entry_identifier* retourné par **ldap_first_attribute()**.

Voir aussi **ldap_get_attributes()**.

ldap_next_entry (PHP 3, PHP 4 >= 4.0b1)

Lit l'attribut suivant.

```
int ldap_next_entry (resource link_identifier, resource result_entry_identifier)
```

ldap_next_entry() retourne l'identifiant de l'entrée suivante, dans le résultat qui a été initialisé par **ldap_first_entry()**. S'il n'y a plus d'entrée, retourne FALSE.

ldap_next_entry() sert à retrouver toutes les entrées qui sont stockées dans un résultat. Les appels successifs à **ldap_next_entry()** retourneront les entrées une à une. Le premier appel à **ldap_next_entry()** est fait après un appel à **ldap_first_entry()** avec *result_entry_identifier* retourné par **ldap_first_entry()**.

Voir aussi **ldap_get_entries()**.

ldap_read (PHP 3, PHP 4 >= 4.0b1)

Lit une entrée.

```
resource ldap_read (resource link_identifier, string base_dn, string filter [, array
attributes [, int attrsonly [, int sizelimit [, int timelimit [, int deref]]]]])
```

ldap_read() retourne un identifiant de résultat en cas de succès, et FALSE sinon.

ldap_read() effectue une recherche avec le filter *filter* dans le dossier base_dn et avec l'option LDAP_SCOPE_BASE (recherche limitée au dossier, ou récursive). Cela revient à lire une entrée dans un dossier.

Les filtres vides ne sont pas autorisés. Si vous souhaitez lire toutes les informations d'un dossier, utiliser le filtre suivant : "objectClass=*". Si vous savez quel est le type des entrées dans le dossier que vous fouillez, vous pouvez aussi adapter ce filter de la façon suivante "objectClass=inetOrgPerson".

ldap_read() dispose de 5 arguments optionnels. Reportez-vous à **ldap_search()**.

> **Note :** Ces paramètres optionnels ont été ajoutés à partir de la version 4.0.2: *attrsonly*, *sizelimit*, *timelimit*, *deref*.

ldap_rename (PHP 4 >= 4.0.5)

Modifie le nom d'une entrée

```
boolean ldap_rename (int link_identifier, string dn, string newrdn, string newparent,
boolean deleteoldrdn)
```

ldap_rename() renomme (ou déplace) l'entrée *dn*. Le nouveau RDN est spécifié par *newrdn* et le nouveau père est spécifié par *newparent*. Si le paramètre *deleteoldrdn* est TRUE, l'ancienne valeur sera supprimée et la nouvelle valeur sera retenue. **ldap_rename()** retourne TRUE en cas de succès, et FALSE sinon.

ldap_rename() ne fonctionne qu'avec LDAPv3. Pour les anciennes versions, utilisez plutôt la fonction **ldap_set_option()**.

ldap_rename() n'est disponible qu'avec les serveurs OpenLDAP 2.x.x ou Netscape Directory SDK x.x, et a été ajoutée en PHP 4.0.5.

ldap_search (PHP 3, PHP 4 >= 4.0b1)

Recherche dans tout l'arbre LDAP.

```
resource ldap_search (resource link_identifier, string base_dn, string filter [, array
attributes [, int attrsonly [, int sizelimit [, int timelimit [, int deref]]]]])
```

ldap_search() retourne un identifiant de résultat en cas de succès, et FALSE sinon.

ldap_search() effectue une recherche avec le filtre *filter* dans le dossier *base_dn*, et avec l'option de récursivité LDAP_SCOPE_SUBTREE. Cela revient à rechercher dans toute la base sous le dossier *base_dn*.

Le quatrième paramètre est optionnel, et peut être ajouté pour restreindre les attributs et les valeurs retournées. Il est beaucoup plus efficace que la méthode qui consiste à lire tous les attributs et leur valeurs associées. L'utilisation de ce quatrième paramètre est encouragé.

Le quatrième paramètre est un tableau de chaînes, qui contient les attributs désirés, array("mail","sn","cn"). Notez que le nd *base_dn* est toujours retourné, quelques que soient les attributs demandés.

Notez que certains serveurs sont configurés pour limiter le nombre de résultats. Si cela arrive, le serveur indiquera qu'il n'a transféré qu'une partie du résultat. Cela arrivera aussi si le sixième paramètre *sizelimit* a été utilisé pour limiter le nombre de lignes lues.

Le cinquième paramètre *attrsonly* doit être mis à 1 si seuls les types d'attributs sont demandés. S'il est mis à 0, les types des attributs et leur valeur seront lue (comportement par défaut).

Le sixième paramètre *sizelimit* permet de limiter le nombre de lignes lues. 0 indique qu'il n'y a pas de limite. NOTE : ce paramètre ne peut PAS annuler la configuration du serveur. Il peut seulement la réduire.

Le septième paramètre *timelimit* fixe la durée de la recherche en seconde. 0 indique qu'il n'y a pas de limite. NOTE : ce paramètre ne peut PAS annuler la configuration du serveur. Il peut seulement la réduire.

Le huitième paramètre *deref* indique le comportement à suivre avec les alias durant une recherche. Sa valeur peut être l'une de celles-ci :

- LDAP_DEREF_NEVER - (par défaut) les alias ne sont pas déréférencés.
- LDAP_DEREF_SEARCHING - aliases sont déréférencés durant la recherche mais pas lors de leur localisation.
- LDAP_DEREF_FINDING - aliases sont déréférencés durant leur localisation mais pas lors de la recherche.
- LDAP_DEREF_ALWAYS - les alias sont toujours déréférencés.

Ces paramètres optionnels ont été ajoutés à partir de la version 4.0.2: *attrsonly*, *sizelimit*, *timelimit*, *deref*.

La chaîne de filtre peut être simple ou complexe. Elle utilise les opérateurs booléens au même format que celui décrit dans les documentations LDAP. (Allez voir celle de Netscape Directory SDK (http://developer.netscape.com/tech/directory/) pour plus d'informations sur les filtres).

L'exemple suivant récupère toutes les unités organisationnelles, le nom, prénom et email, dans la société "Ma Société" où le nom et prénom contiennent la sous-chaîne $person. Cet exemple utilise un filtre booléen pour indiquer au serveur qu'il doit rechercher des informations dans plusieurs attributs.

Exemple 1. Recherche LDAP

```
<?php
// $ds est un identifiant valide de connexion à un serveur LDAP
// $person est tout ou une partir d'un nom
  $dn = "o=Ma Société, c=Fr";
  $filter="(|(sn=$person*)(givenname=$person*))";
  $justthese = array( "ou", "sn", "givenname", "mail");
  $sr=ldap_search($ds, $dn, $filter, $justthese);
  $info = ldap_get_entries($ds, $sr);
  print $info["count"]." Entrées retournées.<p>";
?>
```

ldap_set_option (PHP 4 >= 4.0.4)

Modifie une option LDAP

```
boolean ldap_set_option (resource link_identifier, int option, mixed newval)
```

ldap_set_option() remplace la valeur de l'option *option* par *newval*, et retourn TRUE en cas de succès, FALSE sinon.

Le paramètre *option* peut prendre l'une des valeurs suivantes : LDAP_OPT_DEREF, LDAP_OPT_SIZELIMIT, LDAP_OPT_TIMELIMIT, LDAP_OPT_PROTOCOL_VERSION, LDAP_OPT_ERROR_NUMBER, LDAP_OPT_REFERRALS, LDAP_OPT_RESTART, LDAP_OPT_HOST_NAME, LDAP_OPT_ERROR_STRING, LDAP_OPT_MATCHED_DN, LDAP_OPT_SERVER_CONTROLS, LDAP_OPT_CLIENT_CONTROLS. Pour une brève description, reportez-vous au fichier draft-ietf-ldapext-ldap-c-api-xx.txt (http://www.openldap.org/devel/cvsweb.cgi/~checkout~/doc/drafts/draft-ietf-ldapext-ldap-c-api-xx.txt).

Les options LDAP_OPT_DEREF, LDAP_OPT_SIZELIMIT, LDAP_OPT_TIMELIMIT, LDAP_OPT_PROTOCOL_VERSION et LDAP_OPT_ERROR_NUMBER ont une valeur entière, LDAP_OPT_REFERRALS et LDAP_OPT_RESTART sont des booléens, et LDAP_OPT_HOST_NAME,

LDAP_OPT_ERROR_STRING et LDAP_OPT_MATCHED_DN sont des chaînes de caractères. Le premier exemple illustre leur utilisation. LDAP_OPT_SERVER_CONTROLS et LDAP_OPT_CLIENT_CONTROLS requiert une liste de contrôles, ce qui signifie que la valeur peut être un tableau de contrôles. Un contrôle est constitué d'un *oid* identifiant le contrôle, d'une valeur *value* optionnelle, et d'un flag optionnel de *criticalité*. En PHP un contrôle est un tableau ayant la clé *oid* et une valeur sous forme de chaîne, ainsi que deux éléments optionnels. Ces éléments ont pour clé *value*, sous forme de chaîne, et une clé *iscritical* contenant un booléen. Par défaut, *iscritical* vaut `FALSE` Voyez aussi l'exemple ci-dessous.

Cette fonction n'est disponible que lorsque vous utilisez OpenLDAP 2.x.x OU Netscape Directory SDK x.x. Elle a été ajoutée en PHP 4.0.4.

Exemple 1. Modification de la version du protocole

```
<?php
// $ds est un identifiant valide d'annuaire
  if (ldap_set_option($ds, LDAP_OPT_PROTOCOL_VERSION, 3))
    echo "Utilisons LDAPv3";
  else
    echo "Impossible de changer le protocole en version 3";
?>
```

Exemple 2. Modifier les contrôles du serveur LDAP

```
<?php
// $ds est un lien valide LDAP
// control n'est pas une valeur
  $ctrl1 = array("oid" => "1.2.752.58.10.1", "iscritical" => TRUE);
// iscritical vaut par défaut FALSE
  $ctrl2 = array("oid" => "1.2.752.58.1.10", "value" => "magic");
// modification des deux contrôles
  if (!ldap_set_option($ds, LDAP_OPT_SERVER_CONTROLS, array($ctrl1, $ctrl2)))
    echo "Impossible de se connecter au serveur";
?>
```

Voir aussi **ldap_get_option()**.

ldap_unbind (PHP 3, PHP 4 >= 4.0b1)

Termine la liaison avec un serveur LDAP.

```
int ldap_unbind (resource link_identifier)
```

ldap_unbind() retourne `TRUE` en cas de succès, et `FALSE` sinon.

ldap_unbind() termine la liaison avec le serveur LDAP.

XLIV. Email

mail() envoie du courrier éléctronique.

mail (PHP 3, PHP 4 >= 4.0b1)

Envoi de mail

```
boolean mail (string to, string subject, string message, string [additional_headers],
string [additional_parameters])
```

mail() poste automatiquement le message *message* à destination de *to*. Les destinataires multiples doivent être séparés par des virgules. Les emails avec pièces jointes ou contenus particuliers (comme les emails en HTML, par exemple), peuvent être réalisés avec cette fonction. Il faut respecter l'encodage MIME. Pour plus de détails, voyez http://www.zend.com/zend/spotlight/sendmimeemailpart1.php et la RFC 1896 (Visit http://www.rfc-editor.org/).

mail() retourne TRUE si le mail est envoyé, et FALSE sinon.

Exemple 1. Envoi de courrier électronique (mail)

```
<?php
  mail("rasmus@lerdorf.on.ca", "Mon Sujet", "Ligne 1\nLigne 2\nLigne 3");
?>
```

Le quatrième argument passé sera inséré à la fin de l'en-tête. Typiquement, cela permet d'insérer des en-têtes supplémentaires. Les en-têtes multiples doivent être séparés par des virgules.

> **Note :** Sous Windows 32bits, vous devez utiliser `\r\n` pour séparer les en-têtes. Notez aussi que les en-têtes cc: et bcc: sont sensibles à la casse et doivent être écrits `Cc:` et `Bcc:` sous Win32.

Si le cinquième argument *additional_parameters* est fourni, PHP l'utilisera dans son appel du programme d'envoi de courrier électronique. Ceci est pratique pour passer une valeur correcte à l'en-tête `Return-Path`, avec sendmail.

> **Note :** Le cinquième paramètre a été ajouté en PHP 4.0.5.

Exemple 2. Envoi de eMail avec des en-têtes supplémentaires.

```
<?php
  mail("nobody@aol.com", "Le sujet", $message,
       "From: webmaster@$SERVER_NAME\nReply-To: webmaster@$SERVER_NAME\nX-
Mailer: PHP/" . phpversion());
?>
```

Avec le cinquième paramètre, vous pouvez ajouter d'autres paramètres de ligne de commande qui seront utilisés par le programme d'envoi de courrier. Dans l'exemple ci-dessous, l'en-tête `Return-Path` est correctement paramétré. Normalement, sendmail ajoute automatiquement l'en-tête `X-Authentication-Warning` (paramètre -f), car l'utilisateur "serveur web" n'est probablement pas un de ses utilisateurs de confiance ("trusted users"). Pour supprimer cette alerte, ajoutez l'utilisateur du serveur web dans la configuration de sendmail.

Exemple 3. Envoi de eMail avec des en-têtes supplémentaires et un paramètre de ligne de commande supplémentaire

```
<?php
  mail("nobody@aol.com", "the subject", $message, "From: webmaster@$SERVER_NAME", "-
fwebmaster@$SERVERNAME");
?>
```

Vous pouvez aussi utiliser des techniques simples de concaténations de chaînes pour construire des messages complexes :

Exemple 4. Envoi de mail complexe.

```
<?php
  /* destinataire */
    $recipient .= "Mary <mary@u.college.edu>", " ; //remarquez les virgules
    $recipient .= "Kelly <kelly@u.college.edu>", ";
    $recipient .= "ronabop@php.net";
  /* sujet */
    $subject = "Rappel des anniversaires du mois d'aoÛt";
  /* message */
    $message .= "Le mail suivant inclut une table au format ASCII\n";
    $message .= "Jour \t\tMois \t\tAn\n";
    $message .= "3 \t\tAou\t\t1970\n";
    $message .= "17\t\tAou\t\t1973\n";
  /* Vous pouvez ajouter une signature */
    $message .= "--\r\n";
  //Délimiteur de signature
    $message .= "Rappel d'anniversaire : copyleft par public domain";
  /* D'autres en-têtes : errors, From cc's, bcc's, etc */
    $headers .= "From: Rappel d'anniversaire <birthday@php.net>\n";
    $headers .= "X-Sender: <birthday@php.net>\n";
    $headers .= "X-Mailer: PHP\n"; // maileur
    $headers .= "X-Priority: 1\n"; //  Message urgent!
    $headers .= "Return-Path: <birthday@php.net>\n";  // Re-
chemin de retour pour les erreurs
    $headers .= "Content-Type: text/html; charset=iso-8859-1\n" // Type MIME
    $headers .= "Cc:birthdayarchive@php.net\n"; // Champs CC
    $headers .= "Bcc:birthdaycheck@php.net, birthdaygifts@php.net"; // Champs BCCs
  /* et hop, à la poste */
    mail($recipient, $subject, $message, $headers);
?>
```

Note : Assurez-vous qu'il n'y a aucune nouvelle ligne (ou d'autre espace ou caractère blanc) dans les paramètres *`to`* ou *`subject`*, car cela peut avoir des effets secondaires irrationnels.

ezmlm_hash (PHP 3>= 3.0.17, PHP 4 >= 4.0.2)

Calcule la valeur de hash demandée par EZMLM

```
int ezmlm_hash (string addr)
```

ezmlm_hash() calcule la valeur de hash, nécessaire lors de la gestion de liste de diffusions EZMLM dans une base de données MySQL.

Exemple 1. Calcul du hash et enregistrement d'un utilisateur

```
<?php
  $user = "kris@koehntopp.de";
  $hash = ezmlm_hash($user);
  $query = sprintf("INSERT INTO sample VALUES (%s, '%s')", $hash, $user);
  $db->query($query);
// utilisation de l'interface PHPLIB
?>
```

XLV. Mathématiques

Introduction

Ces fonctions ne sont capables de manipuler que des entiers "double", ou des "long". Si vous avez besoin de manipuler des nombres plus grands, reportez-vous aux fonctions mathématiques sur des nombres de grande taille.

Constantes mathématiques

Les valeurs suivantes sont définies comme des constantes en PHP :

Tableau 1. Constantes mathématiques

Constante	Valeur	Description
M_PI	3.14159265358979323846	Pi
M_E	2.7182818284590452354	e
M_LOG2E	1.4426950408889634074	log_2 e
M_LOG10E	0.43429448190325182765	log_10 e
M_LN2	0.69314718055994530942	log_e 2
M_LN10	2.30258509299404568402	log_e 10
M_PI_2	1.57079632679489661923	pi/2
M_PI_4	0.78539816339744830962	pi/4
M_1_PI	0.31830988618379067154	1/pi
M_2_PI	0.63661977236758134308	2/pi
M_SQRTPI	1.77245385090551602729	sqrt(pi) [4.0.2]
M_2_SQRTPI	1.12837916709551257390	2/sqrt(pi)
M_SQRT2	1.41421356237309504880	sqrt(2)
M_SQRT3	1.73205080756887729352	sqrt(3) [4.0.2]
M_SQRT1_2	0.70710678118654752440	1/sqrt(2)
M_LNPI	1.14472988584940017414	log_e(pi) [4.0.2]
M_EULER	0.57721566490153286061	Euler constant [4.0.2]

Abs (PHP 3, PHP 4 >= 4.0b1)

Valeur absolue

```
mixed abs (mixed number)
```

abs() retourne la valeur absolue du nombre *number*. Si le nombre est un nombre à virgule flottante (float), le type retourné est aussi un nombre à virgule flottante (float), sinon, c'est un entier (integer).

Acos (PHP 3, PHP 4 >= 4.0b1)

arc cosinus

```
float acos (float arg)
```

acos() retourne l'arc cosinus de *arg* (*arg* en radians).

Voir aussi **asin()** et **atan()**.

Asin (PHP 3, PHP 4 >= 4.0b1)

arc sinus

```
float asin (float arg)
```

asin() retourne l'arc sinus de *arg* (*arg* en radians).

Voir aussi **acos()** et **atan()**.

Atan (PHP 3, PHP 4 >= 4.0b1)

arc tangent

```
float atan (float arg)
```

atan() retourne l'arc tangent de *arg* (*arg* en radians).

Voir aussi **acos()** et **atan()**.

Atan2 (PHP 3>= 3.0.5, PHP 4 >= 4.0b1)

arc tangent de deux variables

```
float atan2 (float y, float x)
```

atan2() retourne l'arc tangent de deux variables *x* et *y*. La formule est : " arc tangent (y / x) ", et les signes des arguments sont utilisés pour déterminer le quadrant du résultat.

atan2() retourne un résultat en radians, entre -PI et PI (inclus).

Voir aussi **acos()** et **atan()**.

base_convert (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Convertit un nombre entre des bases arbitraires.

```
string base_convert (string number, int frombase, int tobase)
```

base_convert() retourne une chaîne contenant l'argument *number* représenté dans la base *tobase*. La base de représentation de *number* est donnée par *frombase*. *frombase* et *tobase* doivent être compris entre 2 et 36 inclus. Les chiffres supérieurs à 10 des bases supérieures à 10 seront représentés par les lettres de A à Z, avec A = 10 et Z = 36.

Exemple 1. base_convert()

```
<?php
  $binary = base_convert($hexadecimal, 16, 2);
?>
```

BinDec (PHP 3, PHP 4 >= 4.0b1)

Convertit de binaire en décimal

```
int bindec (string binary_string)
```

bindec() retourne la conversion d'un nombre binaire représenté par la chaîne *binary_string* en décimal.

bindec() convertit un nombre binaire en décimal. Le plus grand nombre convertible a 31 bits à 1, soit 2147483647 en décimal.

Voir aussi **decbin()**.

Ceil (PHP 3, PHP 4 >= 4.0b1)

Arrondit au nombre supérieur

```
float ceil (float number)
```

ceil() retourne l'entier supérieur du nombre *number*. Utiliser **ceil()** sur un entier ne sert à rien. La valeur retournée est un nombre à virgule flottante (float), car ces nombres peuvent être plus grands que les entiers.

```
<?php
  $x = ceil(4.25);
//  ce qui donne $x=5
?>
```

NOTE: **ceil()** sous PHP/FI 2 retournait un nombre à virgule flottante. Utilisez: `$new = (double)ceil($number);` pour retrouver le comportement traditionnel.

Voir aussi **floor()** et **round()**.

Cos (PHP 3, PHP 4 >= 4.0b1)

cosinus

```
float cos (float arg)
```

cos() retourne le cosinus de *arg* (*arg* en radians).

Voir aussi **sin()** et **tan()**.

DecBin (PHP 3, PHP 4 >= 4.0b1)

Convertit de décimal en binaire

```
string decbin (int number)
```

decbin() retourne une chaîne contenant la représentation binaire de l'entier donné en argument. Le plus grand nombre pouvant être converti est 2147483647 en décimal, ce qui donne une série de 31 uns (1).

Voir aussi **bindec()**.

DecHex (PHP 3, PHP 4 >= 4.0b1)

Convertit de décimal en hexadécimal

```
string dechex (int number)
```

dechex() retourne une chaîne contenant la représentation hexadécimale du nombre *number*. Le nombre le plus grand qui puisse être converti est 2147483647 en décimal, ce qui donnera "7fffffff".

Voir aussi **hexdec()**.

DecOct (PHP 3, PHP 4 >= 4.0b1)

Convertit de décimal en octal

```
string decoct (int number)
```

decoct() retourne une chaîne contenant la représentation octale du nombre donné *number*. Le nombre le plus grand qui puisse être converti est 2147483647 en décimal, ce qui donnera "17777777777".

Voir aussi **octdec()**.

deg2rad (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Convertit un nombre de degrés en radians

```
double deg2rad (double number)
```

deg2rad() convertit *number* de degrés en radians.

Voir aussi **rad2deg()**.

Exp (PHP 3, PHP 4 >= 4.0b1)

exponentielle

```
float exp (float arg)
```

exp() retourne l'exponentielle de *arg*, c'est-à-dire e élevé à la puissance *arg*.

Voir aussi **pow()**.

Floor (PHP 3, PHP 4 >= 4.0b1)

Arrondi à l'entier inférieur

```
floats floor (float number)
```

floor() retourne l'entier inférieur du nombre *number*. La valeur retournée est un nombre à virgule flottante, (float) car ces nombres peuvent être plus grands que les entiers.

NOTE: **floor()** sous PHP/FI retournait un float. Utilisez: `$new = (double)floor($number);` pour retrouver le comportement traditionnel.

Voir aussi **ceil()** et **round()**.

getrandmax (PHP 3, PHP 4 >= 4.0b1)

Plus grande valeur aléatoire possible.

```
int getrandmax (void)
```

getrandmax() retourne la plus grande valeur aléatoire possible retournée par **rand()**.

Voir aussi **rand()**, **srand() mt_rand()**, **mt_srand()** et **mt_getrandmax()**.

hexdec (PHP 3, PHP 4 >= 4.0b1)

Convertit de hexadécimal en décimal

```
int hexdec (string hex_string)
```

hexdec() retourne une chaîne contenant la représentation décimale du nombre *hex_string*. Le nombre le plus grand qui puisse être converti est 7fffffff en décimal, ce qui donne "2147483647".

hexdec() remplace tous les caractères non-héxadecimal par des `0`. Et si les zéros de gauche sont ignorés, ceux de droite prennent le propre valeur.

Exemple 1. Exemple avec hexdec()

```
<?php
  var_dump(hexdec("Hop comme ceci"));
  var_dump(hexdec("0000c000e0cec0"));
  var_dump(hexdec("c000e0cec0"));
// les deux affichent "int(14732992)"
  var_dump(hexdec("aussi"));
  var_dump(hexdec("a0000"));
// les deux affichent "int(655360)"
```

```
?>
```

Voir aussi **dechex()**.

lcg_value (PHP 4 >= 4.0b4)

Générateur de congruence combinée linéaire

```
double lcg_value (void)
```

lcg_value() retourne un nombre pseudo-aléatoire, compris entre 0 et 1. **lcg_value()** combine deux générateurs de congruence, de périodes respectives `2^31 - 85` et `2^31 - 249`. La période de cette fonction est le produit de ces deux nombres premiers (soit `(2^31 - 85)*(2^31 - 249)`).

Log (PHP 3, PHP 4 >= 4.0b1)

Logarithme naturel (népérien)

```
float log (float arg)
```

log() retourne le logarithme naturel (ou népérien) de *arg*.

Log10 (PHP 3, PHP 4 >= 4.0b1)

logarithme en base 10.

```
float log10 (float arg)
```

log10() retourne le logarithme en base 10 de *arg*.

max (PHP 3, PHP 4 >= 4.0b1)

La plus grande valeur.

```
mixed max (mixed arg1, mixed arg2, mixed argn)
```

max() retourne la plus grande valeur numérique parmi les valeurs passées en paramètre.

Si le premier paramètre est un tableau, **max()** retourne la plus grande valeur de ce tableau. Si le premier paramètre est un entier, une chaîne ou un double, **max()** requiert au moins deux paramètres, et retournera alors le plus grand d'entre eux. Le nombre d'arguments est alors illimité.

Si au moins une valeur est un entier double, elles seront toutes traitées comme des doubles, et un double sera retourné. Si aucune valeur n'est de type double, elles seront traitées comme des entiers, et un entier sera retourné.

min (PHP 3, PHP 4 >= 4.0b1)

La plus petite valeur.

```
mixed min (mixed arg1, mixed arg2, mixed argn)
```

min() retourne la plus petite valeur numérique parmi les valeurs passées en paramètres.

Si le premier paramètre est un tableau, **min()** retourne la plus petite valeur de ce tableau. Si le premier paramètre est un entier, une chaîne ou un double, **min()** requiert au moins deux paramètres, et retournera alors le plus petit d'entre eux. Le nombre d'arguments est alors illimité.

Si au moins une valeur est un entier double, elles seront toutes traitées comme des doubles, et un double sera retourné. Si aucune valeur n'est de type double, elles seront traitées comme des entiers, et un entier sera retourné.

mt_rand (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Génère une meilleure valeur aléatoire.

```
int mt_rand ([int min [, int max]])
```

De nombreux générateurs de nombres aléatoires provenant de vieilles bibliothèques libcs ont des comportement douteux et sont très lents. Par défaut, PHP utilise le générateur de nombres aléatoires de libc avec la fonction **rand()**. **mt_rand()** est une fonction de remplacement, pour cette dernière. Elle utilise un générateur de nombres aléatoire de caractéristique connue, le " Mersenne Twister ", qui va produire des nombres utilisables en cryptographie, et qui est 4 fois plus rapide que la fonction standard libc. La "Homepage of the Mersenne Twister " est http://www.math.keio.ac.jp/~matumoto/emt.html. Une version optimisée des sources de MT est disponible à http://www.scp.syr.edu/~marc/hawk/twister.html.

Appelée sans les arguments optionnels *min* et *max*, **mt_rand()** retourne un nombre pseudo-aléatoire, entre 0 et RAND_MAX. Pour obtenir un nombre entre 5 et 15 inclus, il faut utiliser `mt_rand(5,15)`.

N'oubliez pas d'initialiser le générateur de nombres aléatoires avec **mt_srand()**.

> **Note :** Dans les versions antérieures à la 3.0.7, la signification du paramètre *max* était "longueur". Pour avoir le même résultat, il faut utiliser `mt_rand (5, 11)` pour obtenir un nombre aléatoire entre 5 et 15.

Voir aussi **mt_srand()**, **mt_getrandmax()**, **srand()**, **rand()** et **getrandmax()**.

mt_srand (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Initialise une meilleure valeur aléatoire

```
void mt_srand (int seed)
```

mt_srand() initialise une meilleure valeur aléatoire avec *seed*.

```
<?php
// initialise avec les microsecondes depuis la dernière seconde entière
  mt_srand((double)microtime()*1000000);
  $randval = mt_rand();
?>
```

Voir aussi **mt_rand()**, **mt_getrandmax()**, **srand()**, **rand()** et **getrandmax()**.

mt_getrandmax (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

La plus grand valeur aléatoire possible.

```
int mt_getrandmax (void)
```

mt_getrandmax() retourne la plus grand valeur aléatoire possible que peut retourner **mt_rand()**.

Voir aussi **mt_rand()**, **mt_srand() rand()**, **srand()** et **getrandmax()**.

number_format (PHP 3, PHP 4 >= 4.0b1)

Formate un nombre pour l'affichage.

```
string number_format (float number [, int decimals [, string dec_point [, string
thousands_sep]]])
```

number_format() retourne une chaîne représentant *number* formaté. **number_format()** accepte un, deux ou 4 paramètres (mais pas trois).

Si le seul paramètre *number* est donné, il sera formaté sans partie décimale, mais avec une virgule entre chaque millier.

Si les deux paramètres *number* et *decimals* sont fournis, *number* sera formaté avec *decimals* décimales, un point (".") comme séparateur décimal et une virgule entre chaque millier.

Avec quatre paramètres, *number* sera formaté avec *decimals* décimales, *dec_point* comme séparateur décimal, et *thousands_sep* comme séparateur de milliers.

> **Note :** Seul le premier caractère du paramètre *thousands_sep* est utilisé. Par exemple, si vous utilisez `foo` comme séparateur de milliers, sur le nombre `1000`, **number_format()** retournera `1f000`.

En notation française, on utilise généralement deux chiffres après la virgule, une virgule comme séparateur décimal, et un espace comme séparateur de milliers. Cela donne :

```
<?php
  $nombre = 1234.56;
// Notation anglaise (par défaut)
  $english_format_number = number_format($nombre);
// 1,234.56
// Notation française
  $nombre_format_francais = number_format($nombre, 2, ',', ' ');
// 1 234,56
?>
```

Voir aussi **sprintf()**, **printf()** et **sscanf()**.

OctDec (PHP 3, PHP 4 >= 4.0b1)

Convertit d'octal en décimal.

```
int octdec (string octal_string)
```

octdec() retourne une chaîne contenant la représentation décimale du nombre *octal_tring*. Le nombre le plus grand qui puisse être converti est 17777777777 en décimal, ce qui donnera "2147483647".

Voir aussi **decoct()**.

pi (PHP 3, PHP 4 >= 4.0b1)

Retourne la valeur de pi

```
double pi (void)
```

pi() retourne la valeur de pi.

```
<?php
  echo pi();
//  3.1415926535898
?>
```

pow (PHP 3, PHP 4 >= 4.0b1)

Puissance

```
number pow (number base, number exp)
```

pow() retourne *base* élevé à la puissance *exp*. Si possible, **pow()** retourne un integer.

Si le calcul ne peut être fait, une alerte sera affichée et **pow()** retournera FALSE.

Exemple 1. Quelques exemples avec pow()

```
<?php
  var_dump( pow(2,8) );
// int(256)
  echo pow(-1,20);
// 1
  echo pow(0, 0);
// 1
echo pow(-1, 5.5);
  // erreur
?>
```

> **Avertissement**
>
> En PHP 4.0.6 plus ancien, **pow()** retournait toujours un nombre à virgule flottante (float), et n'affichait pas d'alerte. Si le calcul est impossible (racine d'un nombre négatif, par exemple), **pow()** retournait NAN.

Voir aussi **exp()**.

rad2deg (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Convertit de radians en degrés

```
double rad2deg (double number)
```

rad2deg() convertit *number* (supposé en radians) en degrés.

Voir aussi **deg2rad()**.

rand (PHP 3, PHP 4 >= 4.0b1)

Génère une valeur aléatoire.

```
int rand ([int min [, int max]])
```

Appelée sans les options *min* et *max*, **rand()** retourne un nombre pseudo-aléatoire entre 0 et RAND_MAX. Si vous voulez un nombre aléatoire entre 5 et 15 (inclus), par exemple, utilisez `rand (5, 15)`.

N'oubliez pas d'initialiser le générateur de nombres aléatoires avec **srand()**.

> **Note :** Dans les versions antérieures à la 3.0.7 la signification du paramètre *max* était longueur. Pour avoir le même résultat, il faut utiliser `mt_rand (5, 11)` pour obtenir un nombre aléatoire entre 5 et 15.

Voir aussi **srand()**, **getrandmax()**, **mt_rand()**, **mt_srand()** et **mt_getrandmax()**.

round (PHP 3, PHP 4 >= 4.0b1)

Arrondi.

```
double round (double val [, int precision])
```

round() retourne la valeur arrondie de *val* à la précision *precision* (nombre de chiffres après la virgule).

```
<?php
  $foo = round( 3.4 );   // $foo == 3.0
  $foo = round( 3.5 );   // $foo == 4.0
  $foo = round( 3.6 );   // $foo == 4.0
?>
```

> **Note :** Le paramètre *precision* est disponible uniquement en PHP 4.

Voir aussi **ceil()** et **floor()**.

Sin (PHP 3, PHP 4 >= 4.0b1)

Sinus

```
float sin (float arg)
```

sin() retourne le sinus de *arg* (*arg* in radians).

Voir aussi **cos()** et **tan()**.

Sqrt (PHP 3, PHP 4 >= 4.0b1)

Racine carrée.

```
float sqrt (float arg)
```

sqrt() retourne la racine carrée de *arg*.

srand (PHP 3, PHP 4 >= 4.0b1)

Initialise le générateur de nombres aléatoires

```
void srand (int seed)
```

srand() initialise le générateur de nombres aléatoires avec *seed*.

```
<?php
// initialise avec les microsecondes depuis la dernière seconde entière
  srand((double)microtime()*1000000);
  $randval = rand();
?>
```

Voir aussi **rand()**, **getrandmax()**, **mt_rand()**, **mt_srand()** et **mt_getrandmax()**.

Tan (PHP 3, PHP 4 >= 4.0b1)

Tangente

```
float tan (float arg)
```

tan() retourne la tangente de *arg* (*arg* en radians).

Voir aussi **sin()** et **cos()**.

XLVI. Chaînes de caractères multi-octets

Avertissement

Ce module est *EXPERIMENTAL*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

Introduction

Avertissement

Ce module est expérimental. Les noms des fonctions sont sujets à des changaments probables. Actuellement, les conversions ne supportent que le Japonais.

De nombreuses langues dont les signes ne peuvent pas être exprimés sur un seul octet. Des codes multi-octets sont utilisés pour pallier à cette insuffisance. `mbstring` est développé pour supporter les caractères japonais. Cependant, de nombreuses fonctions `mbstring` peuvent supporter d'autres jeux de caractères.

Les jeux de caractères multi-octets représentent les caractères sur plusieurs octets consécutifs (d'où leur nom). Certains systèmes d'encodages ont des caractères d'échappement dédiés, pour démarrer/finir une séquence de caractères multi-octets. De ce fait, certains caractères peuvent être détruit lorsqu'une chaîne est coupée en plusieurs morceaux, ou bien conduire à des résultats erronnés lorsque le nombre de caractère est compté. Il faut utiliser des fonctions qui supportent ces encodages. Les fonctions `mbstring` supportent les jeux de caractères multi-octets, ainsi que les conversions.

Cas des caractères japonais

La plupart des caractères japonais demandent plus d'un octet pour être représentés. De plus, plusieurs jeux de caractères japonais existent : il y a notamment EUC-JP, Shift_JIS et ISO-2022-JP. Unicode devient de plus en plus populaire, et UTF-8 aussi. Pour développer des applications Web en environnement japonais, il faut savoir que les encodages ci-dessus dépendent de l'application qu'on en fait : entrée/sortie HTTP, bases de données ou courrier électronique.

- La taille nécessaire à un caractère peut aller jusqu'à 4 octets.
- Un caractère multi-octets occupe généralement deux octets, à comparer avec les caractères simple-octet traditionnellement utilisé. Les caractères les plus gros sont appelés "zen-kaku" (i.e. grande largeur) et les plus petits sont appelés "han-kaku" (i.e. demi-largeur). Les caractères "zen-kaku" sont généralement de taille constante.
- Certains encodage de caractères définissent des séquences de début/fin pour les sections multi-octets.
- Les bases de données allouent des tailles de stockages différentes de celles utilisées par PHP, même si le même encodage de caractère est utilisé (par exemple, PostGreSQL).
- Le courrier électronique utilise généralement ISO-2022-JP.
- Les sites web en "i-mode" utilisent Shift_JIS.

Jeux de caractères supportés

Les jeux de caractères suivants sont supportés par cette extension PHP : `UCS-4`, `UCS-4BE`, `UCS-4LE`, `UCS-2`, `UCS-2BE`, `UCS-2LE`, `UTF-32`, `UTF-32BE`, `UTF-32LE`, `UCS-2LE`, `UTF-16`, `UTF-16BE`, `UTF-16LE`, `UTF-8`, `UTF-7`, `ASCII`, `EUC-JP`, `SJIS`, `eucJP-win`, `SJIS-win`, `ISO-2022-JP(JIS)`, `ISO-8859-1`, `ISO-8859-2`, `ISO-8859-3`, `ISO-8859-4`, `ISO-8859-5`, `ISO-8859-6`, `ISO-8859-7`, `ISO-8859-8`, `ISO-8859-9`, `ISO-8859-10`, `ISO-8859-13`, `ISO-8859-14`, `ISO-8859-15`.

Configuration du fichier php.ini

- `mbstring.internal_encoding` définit le jeu de caractères interne par défaut.
- `mbstring.http_input` définit le jeu de caractères d'entrée HTTP par défaut.
- `mbstring.http_output` defines définit le jeu de caractères d'affichage HTTP par défaut.
- `mbstring.detect_order` définit l'ordre de détection des jeux de caractères (lors de la lecture sur une source externe inconnue).
- `mbstring.substitute_character` définit le caractère de substitution pour les codes invalides.

Exemple 1. Exemple de configuration du `php.ini`

```
;; Set default internal encoding
mbstring.internal_encoding    = UTF-8  ; Par défaut, to UTF-8
;; Set default HTTP input character code
mbstring.http_input = auto     ; Par défaut, mode automatique
; or
; mbstring.http_input = SJIS     ; Par défaut, HTTP fournit du code SJIS
; mbstring.http_input = eucjp-win, sjis-win, UTF-8 ; Ordre spécifique
;; Set default HTTP output character code
mbstring.http_output = UTF-8   ; Par défaut, HTTP affiche du code UTF-8
;; Set default character code detection order
mbstring.detect_order = auto   ; Par défaut, mode automatique
; or
; mbstring.detect_order = eucjp-win, sjis-win, UTF-8 ; Ordre spécifique
;; Set default substitute character
mbstring.substitute_character = 12307 ; Spécifie un code de caractère
; or
; mbstring.substitute_character = none  ; Caractère NULL
; mbstring.substitute_character = long  ; Long
```

mb_internal_encoding (PHP 4 >= 4.0.6)

Lit/modifie l'encodage interne

```
string mb_internal_encoding ([string encoding])
```

mb_internal_encoding() modifie l'encodage interne courant en le remplacant par *encoding*. Si ce paramètre est omis, l'encodage interne courant est retourné.

encoding sert lors des conversions des chaînes en provenance et en direction du web, ainsi que lors de la création de chaînes avec le module mbstring.

encoding: Nom d'encodage.

Valeur retournée : si *encoding* est fourni, **mb_internal_encoding()** retourne TRUE en cas de succès, et FALSE sinon. Si *encoding* est omis, **mb_internal_encoding()** retourne le nom de l'encodage courant.

Exemple 1. Exemple avec mb_internal_encoding()

```
<?php
/* Utilise l'encodage interne UTF-8 */
mb_internal_encoding("UTF-8");
/* Affiche l'encodage interne courant */
echo mb_internal_encoding();
?>
```

Voir aussi **mb_http_input()**, **mb_http_output()** et **mb_detect_order()**

mb_http_input (PHP 4 >= 4.0.6)

Détecte le type d'encodage d'un caractère HTTP

```
string mb_http_input ([string type])
```

mb_http_input() retourne le type d'encodage utilisé par une requête HTTP.

Le paramètre *type* spécifie le type d'entrée HTTP. Il peut prendre l'une des valeurs suivantes : "G" pour GET, "P" pour POST, "C" pour COOKIE. Si *type* est omis, il prend la valeur du dernier type utilisé.

Valeur retournée : nom de l'encodage utilisé. Si **mb_http_input()** ne peut traiter ce type d'encodage, elle retourne FALSE.

Voir aussi **mb_internal_encoding()**, **mb_http_output()** et **mb_detect_order()**

mb_http_output (PHP 4 >= 4.0.6)

Lit/modifie l'encodage d'affichage

```
string mb_http_output ([string encoding])
```

Si *encoding* est fourni, **mb_http_output()** utilisera dorénavant l'encodage *encoding* pour les affichages HTTP : les caractères qui seront envoyés aux clients web seront convertis dans le jeu de caractères *encoding*. **mb_http_output()** retourne TRUE en cas de succès, et FALSE en cas d'échec.

Si *encoding* est omis, **mb_http_output()** retourne l'encodage d'affichage courant.

Voir aussi **mb_internal_encoding()**, **mb_http_input()** et **mb_detect_order()**

mb_detect_order (PHP 4 >= 4.0.6)

Lit/modifie l’ordre de détection des encodages

```
array mb_detect_order ([mixed encoding-list])
```

mb_detect_order() remplace l’ordre de détection des encodages courant par *encoding-list*. **mb_detect_order()** retourne TRUE en cas de succès, et FALSE en cas d’erreur failure.

encoding-list est un tableau, ou une liste d’encodages séparés par une virgule. La valeur "auto" est automatiquement remplacé par "ASCII, JIS, UTF-8, EUC-JP, SJIS".

SI *encoding-list* est omis, **mb_detect_order()** retourne l’ordre de détection courant des encodages.

Ce paramétrage affected les fonctions **mb_detect_encoding()** et **mb_send_mail()**.

Exemple 1. Exemple avec mb_detect_order()

```
<?php
/* Remplace l’ordre de détection par une liste énumérée */
mb_detect_order("eucjp-win,sjis-win,UTF-8");
/* Remplace l’ordre de détection par un tableau */
$ary[] = "ASCII";
$ary[] = "JIS";
$ary[] = "EUC-JP";
mb_detect_order($ary);
/* Affiche l’ordre de détection courant */
echo implode(", ", mb_detect_order());
?>
```

Voir aussi **mb_internal_encoding()**, **mb_http_input()**, **mb_http_output()** et **mb_send_mail()**

mb_substitute_character (PHP 4 >= 4.0.6)

Lit/modifie les caractères de substitution

```
mixed mb_substitute_character ([mixed substrchar])
```

mb_substitute_character() spécifie le caractère de substitution des caractères invalides, ou des encodages invalides. Les caractères invalides peuvent être remplacés par NULL (pas d’affichage, ils sont supprimés), une chaîne ou un code héxadécimal.

Ce paramétrage affecte **mb_detect_encoding()** et **mb_send_mail()**.

substchar spécifie une valeur Unicode sous la forme d’un entier, ou bien une chaîne sous ces formes :

- "none" : pas d’affichage
- "long" : affiche la valeur hexadécimale (Par exemple : U+3000,JIS+7E7E)

SI *substchar* est fourni, **mb_substitute_character()** retourne TRUE en cas de succès, et FALSE en cas d’erreur. Si *substchar* est omis, **mb_substitute_character()** retourne une valeur Unicode, ou bien "none"/"long".

Exemple 1. Exemple avec mb_substitute_character()

```
<?php
/* Configure le caractère de substitution avec U+3013 (GETA MARK) */
mb_substitute_character(0x3013);
/* Configure le caractère de substitution avec un format héxadécimal */
mb_substitute_character("long");
/* Affiche la configuration courante */
echo mb_substitute_character();
?>
```

mb_output_handler (PHP 4 >= 4.0.6)

Fonction de traitement des affichages web

```
string mb_output_handler (string contents, int status)
```

mb_output_handler() est la fonction à fournir à **ob_start()**. **mb_output_handler()** convertit les caractères envoyés au client Web, dans l'encodage paramétré avec **mb_http_output()**.

contents : Le contenu à traiter

status : L'état du contenu

mb_output_handler() retourne la chaîne convertie.

Exemple 1. Exemple avec mb_output_handler()

```
<?php
mb_http_output("UTF-8");
ob_start("mb_output_handler");
?>
```

> **Note :** Si vous souhaitez envoyer des données binaires telles que des images issues d'un script PHP, vous devez spécifier l'encodage spécial `"pass"`, avec la fonction **mb_http_output()**.

Voir aussi **ob_start()**.

mb_preferred_mime_name (PHP 4 >= 4.0.6)

Détecte l'encodage MIME

```
string mb_preferred_mime_name (string encoding)
```

mb_preferred_mime_name() retourne le type d'encodage MIME utilisé dans le mail *encoding*. Le nom de l'encodage est retourné sous forme de chaîne.

Exemple 1. Exemple avec mb_preferred_mime_string()

```
<?php
$outputenc = "sjis-win";
mb_http_output($outputenc);
ob_start("mb_output_handler");
Header("Content-Type: text/html; charset=" . mb_preferred_mime_name($outputenc));
?>
```

mb_strlen (PHP 4 >= 4.0.6)

Retourne la taille d'une chaîne

```
string mb_strlen (string str [, string encoding])
```

mb_strlen() retourne le nombre de caractères dans la chaîne *`str`*, avec l'encodage *`encoding`*. Un caractère multi-octets est alors compté pour 1.

Voir aussi **mb_internal_encoding()**, **strlen()**.

mb_strpos (PHP 4 >= 4.0.6)

Repère la première occurence d'un caractère dans une chaîne

```
string mb_strpos (string haystack, string needle [, int offset [, string encoding]])
```

mb_strpos() retourne la position numérique de la première occurence du caractère *`needle`* dans la chaîne *`haystack`*. Si *`needle`* est introuvable, **mb_strpos()** retourne `FALSE`.

mb_strpos() effectue une recherche de type **strpos()**, en tenant compte des caractères multi-octets. La position de *`needle`* est comptée à partir du début de la chaîne *`haystack`* : les positions commencent à 0.

Si *`encoding`* est omis, l'encodage interne par défaut est utilisé. **mb_strrpos()** accepte des chaînes comme argument *`needle`*, alors que **strrpos()** n'accepte que des caractères.

`offset` est l'offset de début de recherche. S'il est omis, il sera utilisé à 0 (début de la chaîne).

`encoding` est un nom d'encodage de caractères. S'il n'est pas spécifié, l'encodage interne est utilisé.

Voir aussi **mb_strpos()**, **mb_internal_encoding()** et **strpos()**

mb_strrpos (PHP 4 >= 4.0.6)

Repère la dernière occurence d'un caractère dans une chaîne

```
string mb_strrpos (string haystack, string needle [, string encoding])
```

mb_strrpos() retourne la position numérique de la première occurence du caractère *`needle`* dans la chaîne *`haystack`*. Si *`needle`* est introuvable, **mb_strrpos()** retourne `FALSE`.

mb_strrpos() effectue une recherche de type **strpos()**, en tenant compte des caractères multi-octets. La position de *`needle`* est comptée à partir du début de la chaîne *`haystack`* : les positions commencent à 0.

Si *`encoding`* est omis, l'encodage interne par défaut est utilisé. **mb_strrpos()** accepte des chaînes comme argument *`needle`*, alors que **strrpos()** n'accepte que des caractères.

`encoding` est un nom d'encodage de caractères. S'il n'est pas spécifié, l'encodage interne est utilisé.

Voir aussi **mb_strpos()**, **mb_internal_encoding()** et **strrpos()**.

mb_substr (PHP 4 >= 4.0.6)

Lit une sous-chaîne

```
string mb_substr (string str, int start [, int length [, string encoding]])
```

mb_substr() retourne la portion de la chaîne *str* qui commence au caractère *start* et a la longueur de *length* caractères.

mb_substr() effectue une recherche de type **strpos()**, en tenant compte des caractères multi-octets. La position de *needle* est comptée à partir du début de la chaîne *haystack* : les positions commencent à 0.

encoding est un nom d'encodage de caractères. S'il n'est pas spécifié, l'encodage interne est utilisé.

Voir aussi **mb_strcut()** et **mb_internal_encoding()**.

mb_strcut (PHP 4 >= 4.0.6)

Coupe une partie de chaîne

```
string mb_strcut (string str, int start [, int length [, string encoding]])
```

mb_strcut() retourne la portion de la chaîne *str* qui commence au caractère *start* et a la longueur de *length* caractères.

mb_strcut() effectue une recherche de type **strpos()**, en tenant compte des caractères multi-octets. La position de *needle* est comptée à partir du début de la chaîne *haystack* : les positions commencent à 0.

mb_strcut() soustrait la partie de la chaîne *str* qui compte *length* caractères.

encoding est un nom d'encodage de caractères. S'il n'est pas spécifié, l'encodage interne est utilisé.

Voir aussi **mb_substr()** et **mb_internal_encoding()**.

mb_strwidth (PHP 4 >= 4.0.6)

Retourne la largeur d'une chaîne

```
int mb_strwidth (string str [, string encoding])
```

mb_strwidth() retourne la largeur de la chaîne *str*.

Les chaînes à encodage multi-octet sont généralement deux fois plus grandes que les chaînes à simple-octet.

```
Largeur de caractères :
U+0000 - U+0019   0
U+0020 - U+1FFF   1
U+2000 - U+FF60   2
U+FF61 - U+FF9F   1
U+FFA0 -          2
```

encoding est un nom d'encodage de caractères. S'il n'est pas spécifié, l'encodage interne est utilisé.

Voir aussi **mb_strimwidth()** et **mb_internal_encoding()**.

mb_strimwidth (PHP 4 >= 4.0.6)

Tronque une chaîne

```
string mb_strimwidth (string str, int start, int width, string trimmarker [, string
encoding])
```

mb_strimwidth() tronque la chaîne *str* à la longueur *width*. Elle retourne la chaîne tronquée.

SI *trimmarker* est fourni, *trimmarker* est ajoutée à la fin de la chaîne retournée.

start est l'offet de départ, en nombre de caractères depuis le début de la chaîne (cela commence à 0).

encoding est un nom d'encodage de caractères. S'il n'est pas spécifié, l'encodage interne est utilisé.

Exemple 1. Exemple avec mb_strimwidth()

```
<?php
$str = mb_strimwidth($str, 0, 40, "..>");
?>
```

Voir aussi **mb_strwidth()** et **mb_internal_encoding()**.

mb_convert_encoding (PHP 4 >= 4.0.6)

Conversion d'encodage

```
string mb_convert_encoding (string str, string to-encoding [, mixed from-encoding])
```

mb_convert_encoding() convertit la chaîne *str* depuis l'encodage *from-encoding* vers l'encodage *to-encoding*.

str à convertir.

from-encoding est l'encodage de la chaîne *str* à l'origine. Il sera détecté parmi plusieurs encodage fournis sous forme d'un tableau, ou d'une liste d'encodages séparés par des virgules.

Exemple 1. Exemple avec mb_convert_encoding()

```
<?php
/* Convertit l'encodage interne vers SJIS */
$str = mb_convert_encoding($str, "SJIS");
/* Convertit EUC-JP en UTF-7 */
$str = mb_convert_encoding($str, "UTF-7", "EUC-JP");
/* Détecte automatiquement un encodage entre JIS, eucjp-win ou sjis-win,
   Puis convertit en UCS-2LE */
$str = mb_convert_encoding($str, "UCS-2LE", "JIS, eucjp-win, sjis-win");
/* "auto" signifie "ASCII,JIS,UTF-8,EUC-JP,SJIS" */
$str = mb_convert_encoding($str, "EUC-JP", "auto");
?>
```

Voir aussi **mb_detect_order()**.

mb_detect_encoding (PHP 4 >= 4.0.6)

Détecte un encodage

```
string mb_detect_encoding (string str [, mixed encoding-list])
```

mb_detect_encoding() détecte l'encodage utilisé par la chaîne *str*. **mb_detect_encoding()** retourne le nom de l'encodage détecté.

encoding-list est une liste d'encodage, sous forme de tableau, ou bien de chaîne, les valeurs étant séparés par des virgules.

Si *encoding_list* est omis, l'ordre spécifié par **mb_detect_order()** est utilisé.

Exemple 1. Exemple avec mb_detect_encoding()

```
<?php
/* Détecte l'encodage avec les valeurs par défaut */
echo mb_detect_encoding($str);
/* "auto" signifie "ASCII,JIS,UTF-8,EUC-JP,SJIS" */
echo mb_detect_encoding($str, "auto");
/* Spécifie une liste d'encodages possibles avec une liste à virgules */
echo mb_detect_encoding($str, "JIS, eucjp-win, sjis-win");
/* Spécifie une liste d'encodages possibles avec un tableau  */
$ary[] = "ASCII";
$ary[] = "JIS";
$ary[] = "EUC-JP";
echo mb_detect_encoding($str, $ary);
?>
```

Voir aussi **mb_detect_order()**.

mb_convert_kana (PHP 4 >= 4.0.6)

Convertit entre les différents "kana"

```
string mb_convert_kana (string str, string option [, mixed encoding])
```

mb_convert_kana() effectue une conversion "han-kaku" - "zen-kaku" sur la chaîne *str*. Elle retourne la chaîne convertie. Cette fonction n'est utile que pour le japonais.

option est l'option de conversion. La valeur par défaut est "KV".

encoding est un nom d'encodage de caractères. S'il n'est pas spécifié, l'encodage interne est utilisé.

```
Options de conversions possibles :
"r" :  Convertit l'alphabet "zen-kaku" en "han-kaku"
"R" :  Convertit l'alphabet "han-kaku" en "zen-kaku"
"n" :  Convertit les nombres "zen-kaku" en "han-kaku"
"N" :  Convertit les nombres "han-kaku" en "zen-kaku"
"a" :  Convertit les nombres et alphabets "zen-kaku" en "han-kaku"
"A" :  Convertit les nombres et alphabets "zen-kaku" en "han-kaku"
(Les caractères inclus dans les options "a", "A" sont
U+0021 - U+007E en excluant U+0022, U+0027, U+005C, U+007E)
"s" :  Convertit les espaces "zen-kaku" en "han-kaku" (U+3000 -> U+0020)
"S" :  Convertit les espaces "han-kaku" en "zen-kaku" (U+0020 -> U+3000)
"k" :  Convertit "zen-kaku kata-kana" en "han-kaku kata-kana"
"K" :  Convertit "han-kaku kata-kana" en "zen-kaku kata-kana"
"h" :  Convertit "zen-kaku hira-gana" en "han-kaku kata-kana"
```

```
       "H" :  Convertit "han-kaku kata-kana" en "zen-kaku hira-gana"
       "c" :  Convertit "zen-kaku kata-kana" en "zen-kaku hira-gana"
       "C" :  Convertit "zen-kaku hira-gana" en "zen-kaku kata-kana"
       "V" :  Supprime les notations vocales, et les convertit en caractères. A utili-
ser avec "K","H"
```

Exemple 1. Exemple avec mb_convert_kana()

```
<?php
/* Convertit tous les "kana" en "zen-kaku" "kata-kana" */
$str = mb_convert_kana($str, "KVC");
/* Convertit "han-kaku" "kata-kana" en "zen-kaku" "kata-kana"
   et "zen-kaku" alpha-numeric en "han-kaku" */
$str = mb_convert_kana($str, "KVa");
?>
```

mb_encode_mimeheader (PHP 4 >= 4.0.6)

Encode une chaîne pour une en-tête MIME

```
string mb_encode_mimeheader (string str [, string charset [, string transfer-encoding [,
string linefeed]]])
```

mb_encode_mimeheader() convertit la chaîne *str* en en-tête MIME, et retourne la chaîne encodée.

charset est le nom de l'encodage. Par défaut, c'est `ISO-2022-JP`.

transfer-encoding est l'encodage de transfert. Il peut être `"B"` (Base64) ou `"Q"` (Quoted-Printable). Par défaut, c'est `"B"`.

linefeed est le marqueur de fin de ligne. Par défaut, c'est `"\r\n"` (CRLF).

Exemple 1. Exemple avec mb_convert_kana()

```
<?php
$name = ""; // kanji
$mbox = "kru";
$doma = "gtinn.mon";
$addr = mb_encode_mimeheader($name, "UTF-7", "Q") . " <" . $mbox . "@" . $doma . ">";
echo $addr;
?>
```

Voir aussi **mb_decode_mimeheader()**.

mb_decode_mimeheader (PHP 4 >= 4.0.6)

Décode une en-tête MIME

```
string mb_decode_mimeheader (string str)
```

mb_decode_mimeheader() décode l'en-tête MIME *str*, obtenue dans un courrier électronique.

mb_decode_mimeheader() retourne la chaîne décodée, encodée au format interne.

Voir aussi **mb_encode_mimeheader()**.

mb_convert_variables (PHP 4 >= 4.0.6)

Convertit l'encodage de variables

```
string mb_convert_variables (string to-encoding, mixed from-encoding, mixed vars)
```

mb_convert_variables() convertit l'encodage des variables *vars* depuis l'encodage *from-encoding* vers l'encodage *to-encoding*, puis retourne le nom de l'encodage détecté, en cas de succès, ou FALSE en cas d'échec.

from-encoding est une liste d'encodages possibles pour les variables *vars*, fourni sous forme d'un tableau ou d'une liste d'encodage, séparés par des virgules. Si *from-coding* est omis, les encodages fournit dans **mb_detect_order()** sont utilisés.

vars est une référence sur une variables à convertir. Les chaînes, tableaux et objets sont aussi supportés.

Exemple 1. Exemple avec mb_convert_variables()

```
<?php
/* Convertit les variables $post1, $post2 en encodage interne */
$interenc = mb_internal_encoding();
$inputenc = mb_convert_variables($interenc, "ASCII,UTF-8,SJIS-win", $post1, $post2);
?>
```

mb_encode_numericentity (PHP 4 >= 4.0.6)

Encode des entités HTML

```
string mb_encode_numericentity (string str, array convmap [, string encoding])
```

mb_encode_numericentity() convertit la chaîne *str* depuis encodage interne en les codes numériques HTML, puis retourne cette chaîne.

array est un tableau qui spécifie les codes à convertir.

encoding est un nom d'encodage de caractères. S'il n'est pas spécifié, l'encodage interne est utilisé.

Exemple 1. Exemple de paramètre *convmap*

```
<?php
$convmap = array (
 int start_code1, int end_code1, int offset1, int mask1,
 int start_code2, int end_code2, int offset2, int mask2,
 ........
 int start_codeN, int end_codeN, int offsetN, int maskN );
// Spécifie les valeurs Unicode de début (start_codeN) et fin (end_codeN)
// Ajoutez offsetN à la valeur, et faites un ET bit-à-bit avec maskN, puis
// il convertit la valeur obtenu en entite numérique
?>
```

Exemple 2. Exemple avec mb_encode_numericentity()

```
<?php
/* Convertit du ISO-8859-1 en entités HTML */
$convmap = array(0x80, 0xff, 0, 0xff);
$str = mb_encode_numericentity($str, $convmap, "ISO-8859-1");
/* Convertit du code SJIS-win (uniquement le bloc 95-104) en entités numérique */
$convmap = array(
       0xe000, 0xe03e, 0x1040, 0xffff,
       0xe03f, 0xe0bb, 0x1041, 0xffff,
       0xe0bc, 0xe0fa, 0x1084, 0xffff,
       0xe0fb, 0xe177, 0x1085, 0xffff,
       0xe178, 0xe1b6, 0x10c8, 0xffff,
       0xe1b7, 0xe233, 0x10c9, 0xffff,
       0xe234, 0xe272, 0x110c, 0xffff,
       0xe273, 0xe2ef, 0x110d, 0xffff,
       0xe2f0, 0xe32e, 0x1150, 0xffff,
       0xe32f, 0xe3ab, 0x1151, 0xffff );
$str = mb_encode_numericentity($str, $convmap, "sjis-win");
?>
```

Voir aussi **mb_decode_numericentity()**.

mb_decode_numericentity (PHP 4 >= 4.0.6)

Décode les entités HTML en caractères

```
string mb_decode_numericentity (string str, array convmap [, string encoding])
```

mb_decode_numericentity() la chaîne d'entités HTML *`str`* en chaîne, et retourne cette chaîne.

`array` est un tableau qui spécifie les codes à convertir.

`encoding` est un nom d'encodage de caractères. S'il n'est pas spécifié, l'encodage interne est utilisé.

Exemple 1. Exemple avec le paramètre *`convmap`*

```
$convmap = array (
   int start_code1, int end_code1, int offset1, int mask1,
   int start_code2, int end_code2, int offset2, int mask2,
   ........
   int start_codeN, int end_codeN, int offsetN, int maskN );
// Spécifie les valeurs Unicode de début (start_codeN) et fin (end_codeN)
// Ajoutez offsetN à la valeur, et faites un ET bit-à-bit avec maskN, puis
// il convertit la valeur obtenu en entite numérique
?>
```

Voir aussi **mb_encode_numericentity()**.

mb_send_mail (PHP 4 >= 4.0.6)

Envoie un mail encodé ISO-2022-JP (mail japonais)

```
boolean mb_send_mail (string to, string subject, string message [, string
additional_headers [, string additional_parameter]])
```

mb_send_mail() envoie un courrier électronique. Les en-têtes et le corps du mesage sont convertis et encodés en ISO-2022-JP. **mb_send_mail()** est une version adaptée de **mail()**.

to est l'adresse de destination du mail. Les adresses multiples peuvent être spécifiées en les séparant par des virgules.

subject est le sujet du mail.

message est le message du mail.

La chaîne *additional_headers* est insérée à la fin de l'en-tête mail. Elle sert à ajouter d'autres en-têtes email. N'oubliez pas de les séparer par des nouvelles lignes (\n).

mb_send_mail() retourne TRUE en cas de succès, et FALSE en cas d'erreur.

Voir aussi **mail()**.

XLVII. MCAL

MCAL signifie Modular Calendar Access Library (librairie calendaire modulaire).

Libmcal est une librairie C de calendriers. Elle est écrite pour être très modulaire, et dispose de nombreux modules. MCAL est l'équivalent de IMAP pour les calendriers.

Avec mcal, un calendrier peut être ouvert comme une boîte aux lettres. Les calendriers peuvent être des fichiers locaux, ou bien être sur des serveurs ICAP distants, ou encore tout autre format supporté par la librairie.

Les événements peuvent être lus, sélectionnés et enregistrés. Il y a aussi la possibilité d'ajouter des alarmes, et de placer des événéments récurents.

Avec libmcal, les serveurs centralisés peuvent être accédés et utilisés, et remplacent avantageusement tout développement spécifique de base de données.

Pour faire fonctionner cette librairie, vous devez compiler PHP avec l'option `--with-mcal`. Il vous faudra alors avoir installé la librairie mcal. Téléchargez la dernière version à http://mcal.chek.com/ et compilez-la, puis installez-la.

Les constantes suivantes sont définies avec l'extension mcal. Pour les jours de la semaine :

- MCAL_SUNDAY (Dimanche)
- MCAL_MONDAY (Lundi)
- MCAL_TUESDAY (Mardi)
- MCAL_WEDNESDAY (Mercredi)
- MCAL_THURSDAY (Jeudi)
- MCAL_FRIDAY (Vendredi)
- MCAL_SATURDAY (Samedi)

Pour les récurrences :

- MCAL_RECUR_NONE (Aucune)
- MCAL_RECUR_DAILY (Quotidienne)
- MCAL_RECUR_WEEKLY (Hebdomadaire)
- MCAL_RECUR_MONTHLY_MDAY (Mensuelle, date fixe)
- MCAL_RECUR_MONTHLY_WDAY (Mensuelle, jour fixe)
- MCAL_RECUR_YEARLY (Annuelle)

Pour les mois :

- MCAL_JANUARY (Janvier)
- MCAL_FEBRUARY (Février)
- MCAL_MARCH (Mars)
- MCAL_APRIL (Avril)
- MCAL_MAY (Mai)
- MCAL_JUNE (Juin)
- MCAL_JULY (Juillet)
- MCAL_AUGUST (AoÛt)
- MCAL_SEPTEMBER (Septembre)
- MCAL_OCTOBER (Octobre)
- MCAL_NOVEMBER (Novembre)

- MCAL_DECEMBER (Décembre)

La plupart des fonctions utilisent une structure d'événement interne, qui est unique pour chaque connexion. Cela évite d'avoir à passer des objets de grande taille entre les fonctions. Il y a des accesseurs bien pratiques pour créer, initialiser et lire des objets événements.

mcal_open (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Ouvre une connexion MCAL.

```
resource mcal_open (string calendar, string username, string password, int options)
```

mcal_open() retourne un flot MCAL en cas de succès, et FALSE en cas d'erreur.

mcal_open() ouvre une connexion MCAL au serveur *calendar*. Si *options* est spécifié, passe aussi *options* à la boîte aux lettres (???). La structure interne du flot MCAL est initialisée à la connexion.

mcal_popen (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Ouvre une connexion persistante MCAL.

```
resource mcal_popen (string calendar, string username, string password, int options)
```

mcal_popen() retourne un flot MCAL en cas de succès, et FALSE sinon.

mcal_popen() ouvre une connexion MCAL au serveur de calendrier *calendar*. Si les options *options* sont spécifiées, elles sont aussi passé à cette boîte au lettre. La structure interne du flot est aussi initialisée.

mcal_reopen (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Réouvre une connexion MCAL

```
resource mcal_reopen (string calendar, int options)
```

mcal_reopen() réouvre une connexion MCAL.

mcal_reopen() réouvre une connexion MCAL avec le serveur *calendar*. Si les options *options* sont spécifiées, elles sont aussi passé à cette boîte aux lettres.

mcal_close (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Ferme une connexion MCAL.

```
int mcal_close (resource mcal_stream, int flags)
```

mcal_close() ferme la connexion *mcal_stream*.

mcal_create_calendar (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Crée un nouveau calendrier

```
string mcal_create_calendar (resource mcal_stream, string calendar)
```

mcal_create_calendar() crée un nouveau calendrier nommé *calendar*.

mcal_rename_calendar (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Renomme un calendrier

```
string mcal_rename_calendar (resource mcal_stream, string old_name, string new_name)
```

mcal_rename_calendar() renomme le calendrier *old_name* en *new_name*.

mcal_delete_calendar (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Efface un calendrier

```
string mcal_delete_calendar (resource mcal_stream, string calendar)
```

mcal_delete_calendar() efface le calendrier *calendar*.

mcal_fetch_event (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Recherche un événement dans le calendrier.

```
object mcal_fetch_event (resource mcal_stream, int event_id [, int options])
```

mcal_fetch_event() recherche un évenement dans le calendrier spécifié par *id*.
mcal_fetch_event() retourne un objet événement dont les attributs sont :

- int id - ID de l'événement.
- int public - TRUE si l'événement est public, FALSE si il est privé.
- string category - Catégorie de l'événement.
- string title - Titre de l'événement.
- string description - Description de l'événement.
- int alarm - Nombre de minutes avant d'envoyer une alerte pour cet événement.
- object start - Objet contenant une date et une heure.
- object end - Objet contenant une date et une heure.
- int recur_type - type de récurrence
- int recur_interval - intervalle de récurrence
- datetime recur_endate - date de fin de récurrence
- int recur_data - données de récurrence

Tous les objets de date et heure sont construits comme suit :

- int year - année
- int month - mois
- int mday - jour du mois
- int hour - heure
- int min - minutes
- int sec - secondes
- int alarm - nombre de minutes avant de déclencher l'alarme

Les valeurs possibles de recur_type sont :

- 0 - Indique que l'événement ne se répète jamais
- 1 - Indique que l'événement se répète tous les jours
- 2 - Indique que l'événement se répète toutes les semaines
- 3 - Indique que l'événement se répète tous les mois, à la même date (le 10 du mois)
- 4 - Indique que l'événement se répète tous les mois, un certain jours (i.e., le troisième samedi du mois)
- 5 - Indique que l'événement se répète tous les ans

mcal_list_events (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Retourne une liste d'événement entre deux dates.

```
array mcal_list_events (resource mcal_stream, objectbegin_date [, object end_date])
```

mcal_list_events() retourne un tableau d'identifiants d'événéments, compris entre deux dates.
mcal_list_events() prend une date de début et une date de fin. Un tableau d'identifiants est retourné.

mcal_append_event (PHP 4 >= 4.0RC1)

Enregistre un nouvel événement dans un calendrier MCAL.

```
int mcal_append_event (resource mcal_stream)
```

mcal_append_event() enregistre l'événement global dans le calendrier MCAL *mcal_stream*.
mcal_append_event() retourne l'uid de l'enregistement ainsi inséré.

mcal_store_event (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Modifie un événement dans un calendrier MCAL.

```
int mcal_store_event (resource mcal_stream)
```

mcal_store_event() enregistre l'événement global dans le calendrier MCAL *mcal_stream*.
mcal_store_event() retourne l'identifiant de l'événement modifié en cas de succès, et FALSE en cas d'erreur.

mcal_delete_event (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Efface un événement dans un calendrier MCAL.

```
int mcal_delete_event (resource mcal_stream [, int event_id])
```

mcal_delete_event() efface l'événement d'identifiant *uid*.
Retourne TRUE.

mcal_snooze (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Eteint l'alarme d'un événement.

```
int mcal_snooze (int id)
```

mcal_snooze() éteint l'alarme de l'événement identifié par l'UID *uid*.

mcal_snooze() retourne TRUE.

mcal_list_alarms (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Retourne une liste d'événements qui ont une alarme prévue à une date.

```
array mcal_list_alarms (resource mcal_stream [, int begin_year [, int begin_month [, int
begin_day [, int end_year [, int end_month [, int end_day]]]]]])
```

mcal_list_events() retourne un tableau d'identifiants, qui ont une alarme de prévue à la date *alarm_date*. Si seul le flot MCAL est donné, la date de début et de fin de la structure globale sera utilisée.

mcal_list_events() prend une date, et retourne un tableau d'identifiants.

mcal_event_init (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Initialise la structure globale d'un flot.

```
int mcal_event_init (resource mcal_stream)
```

mcal_event_init() initialise la structure globale d'un flot. Cela remet tous les éléments de la structure à 0, ou à leur valeur par défaut.

mcal_event_init() retourne TRUE.

mcal_event_set_category (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Fixe la catégorie de la structure globale.

```
int mcal_event_set_category (resource mcal_stream, string category)
```

mcal_event_set_category() fixe la catégorie de la structure globale à la valeur de *category*.

mcal_event_set_category() retourne TRUE.

mcal_event_set_title (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Fixe le titre de la structure globale.

```
int mcal_event_set_title (resource mcal_stream, string title)
```

mcal_event_set_title() fixe le titre de la structure globale à la valeur de *title*.

mcal_event_set_title() retourne TRUE.

mcal_event_set_description (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Fixe la description de la structure globale.

```
int mcal_event_set_description  (resource mcal_stream, string description)
```

mcal_event_set_description() fixe la catégorie de la structure globale à la valeur de *description*.

mcal_event_set_description() retourne TRUE.

mcal_event_set_start (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Fixe les dates de début et de fin de la structure globale.

```
int mcal_event_set_start (resource mcal_stream, int year, int month, int [day], int [hour],
int [min], int [sec])
```

mcal_event_set_start() fixe la date de début de la structure globale.

mcal_event_set_start() retourne TRUE.

mcal_event_set_end (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Fixe la date de fin de la structure globale.

```
int mcal_event_set_end (resource mcal_stream, int year, int month, int [day], int [hour],
int [min], int [sec])
```

mcal_event_set_end() fixe la date de fin de la structure globale.

mcal_event_set_end() retourne TRUE.

mcal_event_set_alarm (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Fixe l'alarme de la structure globale.

```
int mcal_event_set_alarm (resource mcal_stream, int alarm)
```

mcal_event_set_alarm() fixe l'alarme de la structure globale,à un nombre de minutes avant déclenchement.

mcal_event_set_alarm() retourne TRUE.

mcal_event_set_class (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Fixe la classe de la structure globale.

```
int mcal_event_set_class (resource mcal_stream, int class)
```

mcal_event_set_class() fixe la classe de la structure globale. La classe vaut 0 pour si elle est publique, et 1 si elle est privée.

mcal_event_set_class() retourne TRUE.

mcal_is_leap_year (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Vérifie que l'année est bissextile.

```
int mcal_is_leap_year (int year)
```

mcal_is_leap_year() retoune 1 si l'année *year* est bissextile, et 0 sinon.

mcal_days_in_month (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Retourne le nombre de jour d'un mois.

```
int mcal_days_in_month (int month, int leap_year)
```

mcal_days_in_month() retourne le nombre de jour du mois *month*, et prend en compte le fait que l'année est bissextile avec le paramètre *leap_year*.

mcal_date_valid (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Valide une date.

```
int mcal_date_valid (int year, int month, int day)
```

mcal_date_valid() retourne TRUE si la date (constituée par l'année *year*, le mois *month* et la date *day*) est valide, et FALSE sinon.

mcal_time_valid (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Valide une heure.

```
int mcal_time_valid (int hour, int minutes, int seconds)
```

mcal_time_valid() retourne TRUE si l'heure (constituée par l'heure *hour*, les minutes *minutes* et les secondes *seconds*) est une heure valide, et FALSE sinon.

mcal_day_of_week (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Le jour de la semaine.

```
int mcal_day_of_week (int year, int month, int day)
```

mcal_day_of_week() retourne le jour de la semaine, pour la date constituée par l'année *year*, le mois *month* et la date *day*.

mcal_day_of_year (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Le jour de l'année.

```
int mcal_day_of_year (int year, int month, int day)
```

mcal_day_of_year() retourne le numéro de jour dans l'année pour la date constituée par l'année *year*, le mois *month* et la date *day*.

mcal_date_compare (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Compare deux dates.

```
int mcal_date_compare (int a_year, int a_month, int a_day, int b_year, int b_month, int b_day)
```

mcal_date_compare() compares les deux dates données, et retourne <0, 0, > 0 si a<b, a==b, a>b respectivement.

mcal_next_recurrence (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Retourne la prochaine occurrence d'un événement.

```
int mcal_next_recurrence (resource mcal_stream, int weekstart, array next)
```

mcal_next_recurrence() retourne un objet contenant la prochaine date de l'événement, ou la date de l'événement suivant la date. **mcal_next_recurrence()** retourne un objet date vide si l'événement n'a pas de réoccurrence, ou si quelquechose est invalide. Utilisez *weekstart* pour déterminer le premier jour.

mcal_event_set_recur_none (PHP 3>= 3.0.15, PHP 4 >= 4.0RC1)

Supprime la récurrence de la structure globale.

```
int mcal_event_set_recur_none  (resource mcal_stream)
```

mcal_event_set_recur_none() supprime la récurrence de la structure globale (event->recur_type est mis à MCAL_RECUR_NONE).

mcal_event_set_recur_daily (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Fixe la récurrence quotidienne.

```
int mcal_event_set_recur_daily  (resource mcal_stream, int year, int month, int day, int interval)
```

mcal_event_set_recur_daily() fixe la récurrence quotidienne de la structure globale, jusqu'à la date passée en paramètre. (la date de début est celle de la structure).

mcal_event_set_recur_weekly (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Fixe la récurrence hebdomadaire.

```
int mcal_event_set_recur_weekly  (resource mcal_stream, int year, int month, int day, int
interval, int weekdays)
```

mcal_event_set_recur_weekly() fixe la récurrence hebdomadaire de la structure globale, jusqu'à la date passée en paramètre. (la date de début est celle de la structure).

mcal_event_set_recur_monthly_mday (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Fixe la récurrence.

```
int mcal_event_set_recur_monthly_mday  (resource mcal_stream, int year, int month, int
day, int interval)
```

mcal_event_set_recur_monthly_mday() fixe la récurrence de la structure globale, jusqu'à la date passée en paramètre. (la date de début est celle de la structure).

mcal_event_set_recur_monthly_wday (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Fixe la récurrence mensuelle.

```
int mcal_event_set_recur_monthly_wday  (resource mcal_stream, int year, int month, int
day, int interval)
```

mcal_event_set_recur_monthly_wday() fixe la récurrence mensuelle de la structure globale, jusqu'à la date passée en paramètre. (la date de début est celle de la structure).

mcal_event_set_recur_yearly (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Fixe la récurrence annuelle.

```
int mcal_event_set_recur_yearly  (resource mcal_stream, int year, int month, int day, int
interval)
```

mcal_event_set_recur_yearly() fixe la récurrence annuelle de la structure globale, jusqu'à la date passée en paramètre. (la date de début est celle de la structure).

mcal_fetch_current_stream_event (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Retourne un objet contenant la structure de date pour le flot courant.

```
object mcal_fetch_current_stream_event  (resource mcal_stream)
```

mcal_fetch_current_stream_event() retourne la structure de la date du flot courant sous la forme d'un objet, qui contient :

- int id - ID de l'événement.

- int public - TRUE si l'événement est public, FALSE si il est privé.
- string category - Catégorie de l'événement.
- string title - Titre de l'événement.
- string description - Description de l'événement.
- int alarm - Nombre de minutes avant d'envoyer une alerte pour cet événement.
- object start - Objet contenant une date et une heure.
- object end - Objet contenant une date et une heure.
- int recur_type - type de récurrence
- int recur_interval - intervalle de récurrence
- datetime recur_endate - date de fin de récurrence
- int recur_data - données de récurrence

Tous les objets de date et heure sont construits comme suit :

- int year - année
- int month - mois
- int mday - jour du mois
- int hour - heure
- int min - minutes
- int sec - secondes
- int alarm - nombre de minutes avant de déclencher l'alarme

Les valeurs possibles de recur_type sont :

- 0 - Indique que l'événement ne se répète jamais
- 1 - Indique que l'événement se répète tous les jours
- 2 - Indique que l'événement se répète toutes les semaines
- 3 - Indique que l'événement se répète tous les mois, à la même date (le 10 du mois)
- 4 - Indique que l'événement se répète tous les mois, un certain jours (i.e., le troisième samedi du mois)
- 5 - Indique que l'événement se répète tous les ans

mcal_event_add_attribute (PHP 3>= 3.0.15, PHP 4 >= 4.0RC1)

Ajoute un attribut et une valeur à la structure globale

```
void mcal_event_add_attribute  (resource mcal_stream, string attribute, string value)
```

mcal_event_add_attribute() ajoute l'attribut *attribute* à la structure globale, avec la valeur *value*.

mcal_expunge (unknown)

Supprime tous les événements marqués pour l'effacement

```
int mcal_expunge  (resource mcal_stream)
```

mcal_expunge() supprime tous les événements marqués pour l'effacement.

XLVIII. Cryptage

Ces fonctions utilisent mcrypt (ftp://mcrypt.hellug.gr/pub/mcrypt/libmcrypt).

Ces fonctions permettent d'accéder à la librairie mcrypt, qui dispose d'une grande variété d'algorithmes de cryptage, tels que DES, TripleDES, Blowfish (par défaut), 3-WAY, SAFER-SK64, SAFER-SK128, TWOFISH, TEA, RC2 et GOST en modes CBC, OFB, CFB et ECB. De plus, elle accepte aussi RC6 et IDEA qui sont considérés comme "non libre".

Si vous compilez PHP avec la librairie libmcrypt 2.4.x, les algorithmes suivants sont supportés : CAST, LOKI97, RIJNDAEL, SAFERPLUS, SERPENT ainsi que les chiffrements suivants : ENIGMA (cryptage), PANAMA, RC4 et WAKE. Avec libmcrypt 2.4.x un autre mode de chiffrement est disponible : nOFB.

Pour l'utiliser, téléchargez la librairie libmcrypt-x.x.tar.gz grâce par ici (ftp://mcrypt.hellug.gr/pub/mcrypt/libmcrypt) et suivez les instructions d'installations incluses. Vous aurez aussi besoin de compiler PHP avec le paramètre `--with-mcrypt` pour activer cette extension.

Mcrypt permet de crypter et de décrypter, en utilisant les méthodes mentionnées ci-dessus. Les 4 commandes importantes **mcrypt_cfb()**, **mcrypt_cbc()**, **mcrypt_ecb()** et **mcrypt_ofb()**) peuvent toutes opérer en mode MCRYPT_ENCRYPT et MCRYPT_DECRYPT.

Exemple 1. Crypte une valeur avec un TripleDES, en mode ECB.

```
<?php
$key = "Cette cle est ultra secrete";
$input = "Rencontrons nous dans notre place secrete a 9 h 00.";
$encrypted_data = mcrypt_ecb(MCRYPT_TripleDES, $key, $input, MCRYPT_ENCRYPT);
?>
```

Cet exemple va retourner les données cryptées dans la variable $encrypted_data.

Si vous avez compilé PHP avec libmcrypt 2.4.x, ces fonctions sont toujours disponibles, mais il est vivement conseillé d'utiliser les nouvelles fonctions avancées.

Exemple 2. Encryption d'une valeur avec TripleDES sous 2.4.x en mode ECB

```
<?php
$key = "Ceci est une vraie cle secrete";
$input = "Rendez vous à 9 heures, dans notre planque.";
$td = mcrypt_module_open (MCRYPT_TripleDES, "", MCRYPT_MODE_ECB, "");
$iv = mcrypt_create_iv (mcrypt_enc_get_iv_size ($td), MCRYPT_RAND);
mcrypt_generic_init ($td, $key, $iv);
$encrypted_data = mcrypt_generic ($td, $input);
mcrypt_generic_end ($td);
?>
```

Cet exemple va retourner les données cryptées dans la variable `$encrypted_data`.

Mcrypt peut opérer en 4 modes de cryptage (CBC, OFB, CFB, et ECB). Nous allons présenter la technique d'utilisation de ces modes. Pour plus de références et de détails, reportez vous au livre suivant : Applied Cryptography par Schneier (ISBN 0-471-11709-9).

- ECB (electronic codebook) ECB (electronic codebook) est prévu pour des données aléatoires, telles que des clés. Etant donné que les données sont peu nombreuses et aléatoires, les inconvénients de l'ECB ont ici un effet négatif favorable.
- CBC (cipher block chaining) est spécialement pratique avec les fichiers dont la sécurité ECB n'est pas suffisante.
- CFB (cipher feedback) est la meilleure méthode pour crypter des flots d'octets, quand les octets doivent être encryptés un par un.
- OFB (output feedback) est comparable à CFB, mais peut être utilisé lorsque des erreurs ne doivent pas être propagées.
- nOFB (output feedback, in nbit) est comparable à OFB, mais plus sÛr, car il opère avec la taille de bloc de l'algorithme.
- STREAM est un mode supplémentaire, pour permettre l'utilisation d' algorithmes tels que WAKE ou RC4.

PHP ne supporte par encore le cryptage des flots d'octets. Pour l'instant, PHP n'accepte que le cryptage de chaîne.

Pour obtenir la liste complète des modes de chiffrement, reportez vous aux derniers #define, dans le fichier `mcrypt.h`. En règle générale, vous pouvez accéder à une méthode de chiffrement avec l'option MCRYPT_nomDuChiffrement.

Voici une liste non exhaustive des modes de chiffrements de l'extension mcrypt. Si un chiffrement n'est pas dans cette liste, mais disponible dans la librairie, vous pouvez supposer que cette documentation est hors d'âge.

- MCRYPT_3DES
- MCRYPT_ARCFOUR_IV (libmcrypt 2.4.x seulement)
- MCRYPT_ARCFOUR (libmcrypt 2.4.x seulement)
- MCRYPT_BLOWFISH
- MCRYPT_CAST_128
- MCRYPT_CAST_256
- MCRYPT_CRYPT
- MCRYPT_DES
- MCRYPT_DES_COMPAT (libmcrypt 2.2.x seulement)
- MCRYPT_ENIGMA (libmcrypt 2.4.x seulement, alias de MCRYPT_CRYPT)
- MCRYPT_GOST
- MCRYPT_IDEA (payant)
- MCRYPT_LOKI97 (libmcrypt 2.4.x seulement)
- MCRYPT_MARS (libmcrypt 2.4.x seulement, payant)
- MCRYPT_PANAMA (libmcrypt 2.4.x seulement)
- MCRYPT_RIJNDAEL_128 (libmcrypt 2.4.x seulement)
- MCRYPT_RIJNDAEL_192 (libmcrypt 2.4.x seulement)
- MCRYPT_RIJNDAEL_256 (libmcrypt 2.4.x seulement)
- MCRYPT_RC2
- MCRYPT_RC4 (libmcrypt 2.2.x seulement)
- MCRYPT_RC6 (libmcrypt 2.4.x seulement)
- MCRYPT_RC6_128 (libmcrypt 2.2.x seulement)
- MCRYPT_RC6_192 (libmcrypt 2.2.x seulement)
- MCRYPT_RC6_256 (libmcrypt 2.2.x seulement)
- MCRYPT_SAFER64
- MCRYPT_SAFER128
- MCRYPT_SAFERPLUS (libmcrypt 2.4.x seulement)
- MCRYPT_SERPENT (libmcrypt 2.4.x seulement)
- MCRYPT_SERPENT_128 (libmcrypt 2.2.x seulement)
- MCRYPT_SERPENT_192 (libmcrypt 2.2.x seulement)
- MCRYPT_SERPENT_256 (libmcrypt 2.2.x seulement)
- MCRYPT_SKIPJACK (libmcrypt 2.4.x seulement)
- MCRYPT_TEAN (libmcrypt 2.2.x seulement)
- MCRYPT_THREEWAY
- MCRYPT_TRIPLEDES (libmcrypt 2.4.x seulement)
- MCRYPT_TWOFISH (Pour les anciennes versions de mcrypt 2.x versions, ou mcrypt 2.4.x)
- MCRYPT_TWOFISH128 (TWOFISHxxx sont disponibles avec les nouvelles versions de 2.x, mais pas dans les

versions 2.4.x)

- MCRYPT_TWOFISH192
- MCRYPT_TWOFISH256
- MCRYPT_WAKE (libmcrypt 2.4.x seulement)
- MCRYPT_XTEA (libmcrypt 2.4.x seulement)

Vous devez (mode CFB et OFB) ou pouvez (mode CBC) fournir un vecteur d’initialisation (IV) pour ces modes de chiffrement. IV doit être unique, et avoir la même valeur au chiffrement et au déchiffrement. Pour des données qui seront enregistrées après encryptage, vous pouvez prendre le résultat d’une fonction telle que MD5, appliquée sur le nom du fichier. Sinon, vous pouvez envoyer IV avec les données chiffrées, (reportez vous au chapitre 9.3 de Applied Cryptography by Schneier (ISBN 0-471-11709-9) pour plus de détails sur le sujet).

mcrypt_get_cipher_name (PHP 3>= 3.0.8, PHP 4 >= 4.0b1)

Lit le nom du chiffrement utilisé.

```
string mcrypt_get_cipher_name (int cipher)
```

```
string mcrypt_get_cipher_name (string cipher)
```

mcrypt_get_cipher_name() retourne le nom du chiffrement utilisé.

mcrypt_get_cipher_name() prend le numéro de chiffrement (avec libmcrypt 2.2.x) ou prend la nom du chiffrement (avec libmcrypt 2.4.x) comme paramètre, et retourne le nom du chiffrement, ou `FALSE`, si ce chiffrement n'existe pas.

Exemple 1. Exemple avec mcrypt_get_cipher_name

```
<?php
$cipher = MCRYPT_TripleDES;
print mcrypt_get_cipher_name($cipher);
?>
```

L'exemple ci dessus va donner : `TripleDES`

mcrypt_get_block_size (PHP 3>= 3.0.8, PHP 4 >= 4.0b1)

Retourne la taille de bloc d'un chiffrement.

```
int mcrypt_get_block_size (int cipher)
int mcrypt_get_block_size (string cipher, string module)
```

mcrypt_get_block_size() sert à lire la taille de bloc du chiffrement *`cipher`*.

mcrypt_get_block_size() prend comme argument le chiffrement *`cipher`* et retourne une taille en octets.

Voir aussi : **mcrypt_get_key_size()**.

mcrypt_get_key_size (PHP 3>= 3.0.8, PHP 4 >= 4.0b1)

Retourne la taille de la clé d'un chiffrement.

```
int mcrypt_get_key_size (int cipher)
int mcrypt_get_key_size (string cipher, string module)
```

mcrypt_get_key_size() sert à lire la taille de clé du chiffrement *`cipher`*.

mcrypt_get_block_size() prend comme argument le chiffrement *`cipher`* et retourne une taille en octets.

Voir aussi: **mcrypt_get_block_size()**.

mcrypt_create_iv (PHP 3>= 3.0.8, PHP 4 >= 4.0b1)

Crée un vecteur d'initialisation à partir d'une source aléatoire.

```
string mcrypt_create_iv (int size, int source)
```

mcrypt_create_iv() sert à créer un IV (vecteur d'initialisation).

mcrypt_create_iv() prend deux arguments, *size* détermine la taille de IV, *source* spécifie la source de IV.

La source peut être MCRYPT_RAND (générateur de nombres aléatoires système), MCRYPT_DEV_RANDOM (lecture des données depuis le fichier /dev/random) et MCRYPT_DEV_URANDOM (lecture des données depuis le fichier /dev/urandom). Si vous utilisez MCRYPT_RAND, assurez vous de bien appeler **srand()** pour initialiser le générateur de nombres aléatoires.

Exemple 1. Exemple avec mcrypt_create_iv

```
<?php
$cipher = MCRYPT_TripleDES;
$block_size = mcrypt_get_block_size($cipher);
$iv = mcrypt_create_iv($block_size, MCRYPT_DEV_RANDOM);
?>
```

mcrypt_cbc (PHP 3>= 3.0.8, PHP 4 >= 4.0b1)

Encrypte/décrypte des données en mode CBC

```
string mcrypt_cbc (int cipher, string key, string data, int mode [, string iv])
```

```
string mcrypt_cbc (string cipher, string key, string data, int mode [, string iv])
```

La première syntaxe utilise libmcrypt 2.2.x, et la seconde utilise libmcrypt 2.4.x.

mcrypt_cbc() encrypte ou décrypte (suivant le *mode* sélectionné) les données *data* avec le chiffrement *cipher* et la clé *key* en mode CBC et retourne la chaîne résultant.

Cipher est une des constantes MCRYPT_ciphername

Key est la clé fournie à l'algorithme. Elle doit être tenue secrète.

Data sont les données à traiter.

Mode vaut MCRYPT_ENCRYPT ou MCRYPT_DECRYPT.

IV est le vecteur d'initialisation (optionnel).

Voir aussi: **mcrypt_cfb()**, **mcrypt_ecb()**, et **mcrypt_ofb()**.

mcrypt_cfb (PHP 3>= 3.0.8, PHP 4 >= 4.0b1)

Encrypte/décrypte des données en mode CFB

```
string mcrypt_cfb (int cipher, string key, string data, int mode, string iv)
```

```
string mcrypt_cfb (string cipher, string key, string data, int mode [, string iv])
```

La première syntaxe utilise libmcrypt 2.2.x, et la seconde utilise libmcrypt 2.4.x.

mcrypt_cfb() encrypte ou décrypte (suivant le *mode* sélectionné) les données *data* avec le chiffrement *cipher* et la clé *key* en mode CFB et retourne la chaîne résultant.

Cipher est une des constantes MCRYPT_ciphername

Key est la clé fournie à l'algorithme. Elle doit être tenue secrète.

Data sont les données à traiter.

Mode vaut MCRYPT_ENCRYPT ou MCRYPT_DECRYPT.

IV est le vecteur d'initialisation (optionnel).

Voir aussi: **mcrypt_cbc()**, **mcrypt_ecb()**, et **mcrypt_ofb()**.

mcrypt_ecb (PHP 3>= 3.0.8, PHP 4 >= 4.0b1)

Encrypte/décrypte des données en mode ECB

```
string mcrypt_ecb (int cipher, string key, string data, int mode)

string mcrypt_ecb (string cipher, string key, string data, int mode [, string iv])
```

La première syntaxe utilise libmcrypt 2.2.x, et la seconde utilise libmcrypt 2.4.x.

mcrypt_ecb() encrypte ou décrypte (suivant le *mode* sélectionné) les données *data* avec le chiffrement *cipher* et la clé *key* en mode CFB et retourne la chaîne résultant.

Cipher est une des constantes MCRYPT_ciphername

Key est la clé fournie à l'algorithme. Elle doit être tenue secrète.

Data sont les données à traiter.

Mode vaut MCRYPT_ENCRYPT ou MCRYPT_DECRYPT.

IV est le vecteur d'initialisation (optionnel).

Voir aussi: **mcrypt_cbc()**, **mcrypt_cfb()**, et **mcrypt_ofb()**.

mcrypt_ofb (PHP 3>= 3.0.8, PHP 4 >= 4.0b1)

Encrypte/décrypte des données en mode OFB

```
string mcrypt_ofb (int cipher, string key, string data, int mode, string iv)

string mcrypt_ofb (string cipher, string key, string data, int mode [, string iv])
```

La première syntaxe utilise libmcrypt 2.2.x, et la seconde utilise libmcrypt 2.4.x.

mcrypt_ofb() encrypte ou décrypte (suivant le *mode* sélectionné) les données *data* avec le chiffrement *cipher* et la clé *key* en mode OFB et retourne la chaîne résultant.

Cipher est une des constantes MCRYPT_ciphername

Key est la clé fournie à l'algorithme. Elle doit être tenue secrète.

Data sont les données à traiter.

Mode vaut MCRYPT_ENCRYPT ou MCRYPT_DECRYPT.

IV est le vecteur d'initialisation (optionnel).

Voir aussi: **mcrypt_cbc()**, **mcrypt_cfb()**, et **mcrypt_ecb()**.

mcrypt_list_algorithms (PHP 4 >= 4.0.2)

Liste tous les algorithmes de chiffrement supportés

```
array mcrypt_list_algorithms (string [lib_dir])
```

mcrypt_list_algorithms() sert à lister tous les algorithmes de chiffrement de *lib_dir*. **mcrypt_list_algorithms()** prend un argument optionnel, qui spécifie le dossier qui contient tous les algorithmes. Si il est omis, la valeur de mcrypt.algorithms_dir dans php.ini est utilisée.

Exemple 1. Exemple avec mcrypt_list_algorithms()

```
<?php
$algorithms = mcrypt_list_algorithms ("/usr/local/lib/libmcrypt");
foreach ($algorithms as $cipher) {
echo $cipher."/n";
}
?>
```

L'exemple ci dessus va affiche tous les algorithmes supportés dans le dossier "/usr/local/lib/libmcrypt".

mcrypt_list_modes (PHP 4 >= 4.0.2)

Liste tous les modes de chiffrement supportés

```
array mcrypt_list_modes (string [lib_dir])
```

mcrypt_list_algorithms() sert à lister tous les modes de chiffrement de *lib_dir*. **mcrypt_list_algorithms()** prend un argument optionnel, qui spécifie le dossier qui contient tous les algorithmes. Si il est omis, la valeur de mcrypt.algorithms_dir dans php.ini est utilisée.

Exemple 1. Exemple avec mcrypt_list_modes()

```
<?php
$modes = mcrypt_list_modes ();
foreach ($modes as $mode) {
echo $mode."<br>";";
}
?>
```

L'exemple ci dessus va affiche tous les modes supportés dans le dossier "/usr/local/lib/libmcrypt".

mcrypt_get_iv_size (PHP 4 >= 4.0.2)

Retourne la taille du VI utilisé par un couple chiffrement/mode

```
int mcrypt_get_iv_size (string cipher, string mode)
int mcrypt_get_iv_size (resource td)
```

La première syntaxe utilise libmcrypt 2.2.x, et la seconde utilise libmcrypt 2.4.x.

mcrypt_get_iv_size() retourne la taille du Vecteur d'initialisation (VI). En cas d'erreur, la fonction retourne `FALSE`. Si le VI est ignoré dans le couple chiffrement/mode demandé, zéro est retourné.

`Cipher` est une constante MCRYPT_ciphername qui indique le nom de l'algorithme sous forme de chaîne.

`Mode` est une constante MCRYPT_MODE_modename qui peut valoir : "ecb", "cbc", "cfb", "ofb", "nofb" ou "stream".

mcrypt_encrypt (PHP 4 >= 4.0.2)

Encrypte un texte

```
string mcrypt_encrypt (string cipher, string key, string data, string mode, string [iv])
```

mcrypt_encrypt() encrypte les données, et retourne les données cryptées.

`Cipher` est une constante MCRYPT_ciphername qui indique le nom de l'algorithme sous forme de chaîne.

`Key` est la clé utilisée pour encrypter les données. Si elle est plus petite que nécessaire, elle sera complétée avec des '\0'.

`Data` sont les données qui doivent être encryptées. Si la taille des données n'est pas de la forme n * taille_de_bloc, elles seront complétées avec des '\0'. La valeur retournée peut être plus grande que la valeur d'origine.

`Mode` est une constante MCRYPT_MODE_modename qui peut valoir : "ecb", "cbc", "cfb", "ofb", "nofb" ou "stream".

`IV` (Vecteur d'initialisation) est utilisé pour les modes CBC, CFB, OFB, et dans certains algorithmes de mode STREAM. Si vous le fournissez par le VI, alors qu'il est nécessaire, la fonction affichera une alerte, et utilise un VI composé de caractères '\0'.

Exemple 1. Exemple avec mcrypt_encrypt()

```
<?php
$iv = mcrypt_create_iv (mcrypt_get_iv_size (MCRYPT_RIJNDAEL_256, MCRYPT_MODE_ECB), MCRYPT_RAND);
$key = "Ceci est une clé secrète";
$text = "Rencontrons nous à 11 heures, derrière le monument";
echo strlen ($text)."\n";
$crypttext = mcrypt_encrypt (MCRYPT_RIJNDAEL_256, $key, $text, MCRYPT_MODE_ECB, $iv);
echo strlen ($crypttext)."\n";
?>
```

L'exemple ci dessus affichera : `42 64`

mcrypt_decrypt (PHP 4 >= 4.0.2)

Décrypte un texte

```
string mcrypt_decrypt (string cipher, string key, string data, string mode, string [iv])
```

`Cipher` est une constante MCRYPT_ciphername qui indique le nom de l'algorithme sous forme de chaîne.

`Key` est la clé utilisée pour encrypter les données. Si elle est plus petite que nécessaire, elle sera complétée avec des '\0'.

`Data` sont les données qui doivent être encryptées. Si la taille des données n'est pas de la forme n * taille_de_bloc, elles seront complétées avec des '\0'. La valeur retournée peut être plus grande que la valeur d'origine.

`Mode` est une constante MCRYPT_MODE_modename qui peut valoir : "ecb", "cbc", "cfb", "ofb", "nofb" ou "stream".

`IV` (Vecteur d'initialisation) est utilisé pour les modes CBC, CFB, OFB, et dans certains algorithmes de mode STREAM. Si vous le fournissez par le VI, alors qu'il est nécessaire, la fonction affichera une alerte, et utilise un VI composé de caractères '\0'.

mcrypt_module_open (PHP 4 >= 4.0.2)

Ouvre le module de l'algorithme et le mode à utiliser

```
resource mcrypt_module_open (string algorithm, string algorithm_directory, string mode,
string mode_directory)
```

mcrypt_module_open() ouvre le module de l'algorithme et du mode à utiliser. Le nom de l'algorithme est specifié par le paramètre *algorithm* (par exemple : "twofish"), ou bien une des constantes MCRYPT_ciphername. La librairie est refermée en appelant **mcrypt_module_close()**, mais il n'est pas nécessaire d'appeler cette fonction si **mcrypt_generic_end()** est utilisé. Normalement, **mcrypt_module_open()** retourne un pointeur d'encryption, ou bien FALSE en cas d'erreur.

algorithm_directory et *mode_directory* servent à repérer les modules d'encryption. Si vous fournissez un nom de dossier, il sera utilisé. Si vous passez une chaîne vide (""), la valeur utilisé par *mcrypt.algorithms_dir* ou *mcrypt.modes_dir* sera celle indiquée dans les directives de configuration. Lorsque ces paramètres ne sont pas fournis les valeurs par défaut, compilées avec la librairie sont utilisées. (généralement /usr/local/lib/libmcrypt).

Exemple 1. Exemple avec mcrypt_module_open()

```
<?php
$td = mcrypt_module_open (MCRYPT_DES, "", MCRYPT_MODE_ECB, "/usr/lib/mcrypt-modes");
?>
```

L'exemple ci dessus va essayer d'ouvrir le module de chiffrement par DES, dans le dossier par défaut, et le mode EBC dans le dossier /usr/lib/mcrypt-modes.

mcrypt_generic_init (PHP 4 >= 4.0.2)

Initialise tous les buffers nécessaires

```
int mcrypt_generic_init (resource td, string key, string iv)
```

La taille maximale de la clé doit être cette retournée par **mcrypt_enc_get_key_size()** et toutes les valeurs inférieures seront aussi valides. Le vecteur d'initialisation (VI) doit avoir la taille d'un bloc, mais vous devez lire sa taille en appelant **mcrypt_enc_get_iv_size()**. IV est ignoré en mode ECB. IV DOIT exister en modes CFB, CBC, STREAM, nOFB et OFB. Il doit être aléatoire et unique (mais pas secret). Le même VI doit être utilisé pour le cryptage et le décryptage. Si vous ne voulez pas l'utiliser, remplissez le de zéros, mais ce n'est pas recommandé. La fonction retourne (-1) en cas d'erreur.

Vous devez appeler **mcrypt_generic_init()** avant chaque appel à **mcrypt_generic()** ou **mdecrypt_generic()**.

mcrypt_generic (PHP 4 >= 4.0.2)

Encrypte

```
string mcrypt_generic (resource td, string data)
```

mcrypt_generic() crypte des données. Les données sont complétées par des "\0" pour obtenir une taille de n fois la taille d'un bloc. Elle retourne les données encryptées. Notez que la longueur de la chaîne retournée peut être plus longue que celle passée en argument, à cause du complément.

mdecrypt_generic (PHP 4 >= 4.0.2)

Décrypte

```
string mdecrypt_generic (resource td, string data)
```

mdecrypt_generic(). Notez que la longueur de la chaîne décryptée peut être plus longue que la chaîne originale, car elle peut avoir été complétée par des "\0".

Exemple 1. Exemple avec mdecrypt_generic()

```
<?php
$iv_size = mcrypt_enc_get_iv_size ($td));
$iv = @mcrypt_create_iv ($iv_size, MCRYPT_RAND);
if (@mcrypt_generic_init ($td, $key, $iv) != -1)
{
    $c_t = mcrypt_generic ($td, $plain_text);
    @mcrypt_generic_init ($td, $key, $iv);
    $p_t = mdecrypt_generic ($td, $c_t);
}
if (strncmp ($p_t, $plain_text, strlen($plain_text)) == 0)
    echo "ok";
else
    echo "erreur";
?>
```

L'exemple ci dessus montre comment vérifier que les données avant cryptage sont bien les mêmes que celles après cryptage/décryptage.

mcrypt_generic_end (PHP 4 >= 4.0.2)

Termine un cryptage

```
bool mcrypt_generic_end (resource td)
```

mcrypt_generic_end() termine le cryptage désigné par le pointeur *td*. En fait, elle supprime tous les buffers, et ferme les modules utilisés. Elle retourne FALSE en cas d'erreur, et TRUE sinon.

mcrypt_enc_self_test (PHP 4 >= 4.0.2)

Test un module ouvert

```
int mcrypt_enc_self_test (resource td)
```

mcrypt_enc_self_test() effectue un test du module ouvert et désigné par *td*. Si le test est concluant, elle retourne 0, sinon, 1.

mcrypt_enc_is_block_algorithm_mode (PHP 4 >= 4.0.2)

Teste le cryptage par bloc d'un mode

```
int mcrypt_enc_is_block_algorithm_mode (resource td)
```

mcrypt_enc_is_block_algorithm_mode() retourne 1 si ce mode utilise des algorithmes par blocs, et 0 sinon. (i.e. 0 pour stream, et 1 pour cbc, cfb, ofb).

mcrypt_enc_is_block_algorithm (PHP 4 >= 4.0.2)

Test le cryptage par bloc d'un algorithme

```
int mcrypt_enc_is_block_algorithm (resource td)
```

mcrypt_enc_is_block_algorithm() retourne 1 si l'algorithme utilisé est un algorithme par bloc, et 0 si c'est un algorithme par flot.

mcrypt_enc_is_block_mode (PHP 4 >= 4.0.2)

Teste si le mode retourne les données par bloc

```
int mcrypt_enc_is_block_mode (resource td)
```

mcrypt_enc_is_block_mode() retourne 1 si le mode retourne des blocs d'octets, ou bien 0 si il retourne des octets (par flot). (i.e. 1 pour cbc et ecb, et 0 pour cfb et stream).

mcrypt_enc_get_block_size (PHP 4 >= 4.0.2)

Retourne la taille de bloc d'un algorithme

```
int mcrypt_enc_get_block_size (resource td)
```

mcrypt_enc_get_block_size() retourne la taille de blocs d'un algorithme en octets.

mcrypt_enc_get_key_size (PHP 4 >= 4.0.2)

Retourne la taille maximale de la clé pour un mode

```
int mcrypt_enc_get_key_size (resource td)
```

mcrypt_enc_get_key_size() retourne la taille maximale de clé acceptée par le mode désigné par *td*, en octets.

mcrypt_enc_get_supported_key_sizes (PHP 4 >= 4.0.2)

Retourne un tableau contenant les tailles de clés acceptées par un algorithme

```
array mcrypt_enc_get_supported_key_sizes (resource td)
```

mcrypt_enc_get_supported_key_sizes() retourne un tableau contenant les tailles des clés supportées par l'algorithme désigné par *td*. Si il retourne un tableau vide, c'est que toutes les clés entre 1 et **mcrypt_enc_get_key_size()** sont acceptées par l'algorithme.

mcrypt_enc_get_iv_size (PHP 4 >= 4.0.2)

Retourne la taille du VI d'un algorithme

```
int mcrypt_enc_get_iv_size (resource td)
```

mcrypt_enc_get_iv_size() retourne la taille du VI de l'algorithme désigné par *td*, en octets. Si la valeur retournée est 0, c'est que l'algorithme ne demande pas de VI. Un VI est demandé en mode cbc, cfb et ofb, et parfois en mode stream.

mcrypt_enc_get_algorithms_name (PHP 4 >= 4.0.2)

Retourne le nom de l'algorithme

```
string mcrypt_enc_get_algorithms_name (resource td)
```

mcrypt_enc_get_algorithms_name() retourne le nom de l'algorithme désigné par *td*.

mcrypt_enc_get_modes_name (PHP 4 >= 4.0.2)

Retourne le nom du mode

```
string mcrypt_enc_get_modes_name (resource td)
```

mcrypt_enc_get_modes_name() retourne le nom du mode désigné par *td*.

mcrypt_module_self_test (PHP 4 >= 4.0.2)

Teste un mode

```
bool mcrypt_module_self_test (string algorithm [, string lib_dir])
```

mcrypt_module_self_test() effectue un test sur l'algorithme spécifié. Le paramètre optionnel *lib_dir* contient le chemin jusqu'au module de l'algorithme sur le système.

Retourne TRUE si le test fonctionne, et FALSE sinon.

mcrypt_module_is_block_algorithm_mode (PHP 4 >= 4.0.2)

Indique si un mode fonctionne par bloc

```
bool mcrypt_module_is_block_algorithm_mode (string mode [, string lib_dir])
```

mcrypt_module_is_block_algorithm_mode() retourne TRUE si le mode doit être utilisé avec un algorithme par bloc, sinon retourne 0 (i.e. 0 pour stream, et 1 pour cbc, cfb, ofb). Le paramètre optionnel *lib_dir* contient le chemin jusqu'au module de l'algorithme sur le système.

mcrypt_module_is_block_algorithm (PHP 4 >= 4.0.2)

Indique si un algorithme fonctionne par bloc

```
bool mcrypt_module_is_block_algorithm (string algorithm [, string lib_dir])
```

mcrypt_module_is_block_algorithm() retourne TRUE si *algorithm* est un algorithme par bloc, sinon retourne 0. Le paramètre optionnel *lib_dir* contient le chemin jusqu'au module de l'algorithme sur le système.

mcrypt_module_is_block_mode (PHP 4 >= 4.0.2)

Indique si un mode travaille par blocs

```
bool mcrypt_module_is_block_mode (string mode [, string lib_dir])
```

mcrypt_module_is_block_mode() retourne TRUE si ce mode fournit des blocs d'octets, ou bien un flot d'octets. (i.e. 1 pour cbc et ecb, et 0 pour cfb et stream). Le paramètre optionnel *lib_dir* contient le chemin jusqu'au module de l'algorithme sur le système.

mcrypt_module_get_algo_block_size (PHP 4 >= 4.0.2)

Retourne la taille de bloc d'un algorithme

```
int mcrypt_module_get_algo_block_size (string algorithm [, string lib_dir])
```

mcrypt_module_get_algo_block_size() retourne la taille de bloc d'un algorithme, en octets. Le paramètre optionnel *lib_dir* contient le chemin jusqu'au module de l'algorithme sur le système.

mcrypt_module_get_algo_key_size (PHP 4 >= 4.0.2)

Retourne la taille maximale de clé

```
int mcrypt_module_get_algo_key_size (string algorithm [, string lib_dir])
```

mcrypt_module_get_algo_key_size() retourne la taille maximale de la clé supporté par l'algorithme *algorithm*. Le paramètre optionnel *lib_dir* contient le chemin jusqu'au module de l'algorithme sur le système.

mcrypt_module_get_algo_supported_key_sizes (unknown)

Indique les tailles de clé supportées par un algorithme

```
array mcrypt_module_enc_get_algo_supported_key_sizes (string algorithm [, string lib_dir])
```

mcrypt_module_enc_get_algo_supported_key_sizes() retourne un tableau avec les tailles des clés supportées par l'algorithme *algorithm*. Si le tableau retourné est vide, c'est que toutes les tailles de clé entre 1 et **mcrypt_module_get_algo_key_size()** sont supportées. Le paramètre optionnel *lib_dir* peut contenir le dossier du module sur le système.

XLIX. Hash

Ces fonctions ont été prévues pour fonctionner avec mhash (http://mhash.sourceforge.net/).

Cet ensemble de fonctions représente une interface avec la librairie mhash. mhash accepte un grand nombre d'algorithmes différents, tels que MD5, SHA1, GOST, bien d'autres.

Pour l'utiliser, téléchargez les distributions de mhash depuis le site web ici (http://mhash.sourceforge.net/) et suivez les instructions d'installation incluses. Vous aurez besoin de recompiler PHP avec l'option `--with-mhash` pour activer cette extension.

mhash sert à calculer des sommes de vérification, des signatures de message, etc...

Exemple 1. Calcule un hash de type SHA1 et l'affiche au format hexadécimal

```
<?php
  $input = "Rencontrons-nous à 9h00 dans notre repaire secret.";
  $hash = mhash(MHASH_SHA1, $input);
  print "Le hash est ".bin2hex($hash)."\n";
?>
```

Cela va produire quelque chose du type (Note du Traducteur : c'est le hash de la version anglaise) `Le hash est d3b85d710d8f6e4e5efd4d5e67d041f9cecedafe` Pour avoir une liste complète des hash supportés, reportez-vous à la documentation de mhash. En règle générale, vous pouvez utiliser un algorithme de hash avec le type : MHASH_NOMDEHASH. Par exemple pour utiliser HAVAL vous devez spécifier la constante PHP MHASH_HAVAL.

Voici une liste de hash qui sont actuellement supportés par mhash. Si un hash n'est pas dans la liste, mais qu'il est disponible avec mhash, c'est que ce document a pris de l'âge.

- MHASH_MD5
- MHASH_SHA1
- MHASH_HAVAL
- MHASH_RIPEMD160
- MHASH_RIPEMD128
- MHASH_SNEFRU
- MHASH_TIGER
- MHASH_GOST
- MHASH_CRC32
- MHASH_CRC32B

mhash_get_hash_name (PHP 3>= 3.0.9, PHP 4 >= 4.0b1)

Retourne le nom du hash.

```
string mhash_get_hash_name (int hash)
```

mhash_get_hash_name() sert à connaître le nom d'un hash.

mhash_get_hash_name() prend un numero d'identifiant de hash, et retourne son nom, ou bien FALSE si le hash n'existe pas, ou si une erreur est survenue.

Exemple 1. exemple mhash_get_hash_name()

```
<?php
  $hash = MHASH_MD5;
  print mhash_get_hash_name($hash);
?>
```

L'exemple ci-dessus va afficher : `MD5`

mhash_get_block_size (PHP 3>= 3.0.9, PHP 4 >= 4.0b1)

Retourne la taille de bloc du hash.

```
int mhash_get_block_size (int hash)
```

mhash_get_block_size() sert à connaître la taille de bloc du hash spécifié *hash*.

mhash_get_block_size() prend un seul argument : le *hash* et retourne la taille en octets, ou bien FALSE si le *hash* n'existe pas.

mhash_count (PHP 3>= 3.0.9, PHP 4 >= 4.0b1)

retourne l'identifiant maximal de hash.

```
int mhash_count (void)
```

mhash_count() retourne l'identifiant de hash maximal. Les hash sont numérotés de 0 jusqu'à cet identifiant.

Exemple 1. Parcourir la liste des hash

```
<?php
  $nr = mhash_count();
  for($i = 0; $i <= $nr; $i++) {
    echo sprintf("The blocksize of %s is %d\n",
            mhash_get_hash_name($i),
            mhash_get_block_size($i));
  }
?>
```

mhash (PHP 3>= 3.0.9, PHP 4 >= 4.0b1)

Calcule un hash.

```
string mhash (int hash, string data)
```

mhash() applique la fonction de hash *hash* aux données *data* et retourne le résultat.

mhash_keygen_s2k (PHP 4 >= 4.0.4)

Génère une clé

```
string mhash_keygen_s2k (int hash, string password, string salt, int bytes)
```

mhash_keygen_s2k() génère une clé de *bytes* octets de long, à partir d'un mot de passe. Cette fonction utilise l'algorithme Salted S2K, spécifié dans OpenPGP (RFC 2440). Cet algorithme va utiliser l'algorithme de hashage *hash* pour créer la clé. Le paramètre *salt* doit être différent et suffisamment aléatoire pour chaque clé que vous génèrez, afin de créer des clés différentes. Ce grain de sel réservira lorsque vous vérifierez les clés : c'est alors une bonne idée que de l'ajouter à la fin de la clé générée. *salt* doit avoir la longueur de 8 octets, et sera complété par des 0 si vous ne fournissez par suffisamment de données. N'oubliez pas que les mots de passe fournis par les utilisateurs ne sont pas conseillé pour faire des clés cryptographique, étant donné que les utilisateurs normaux retiennent des mots de passe qu'ils peuvent saisir au clavier. Ces mots de passe utilisent uniquement 6 à 7 des 8 bits d'un caractère (voir moins). Il est vivement recommandé d'appliquer une fonction de transformation (comme celle-ci), à un mot de passe utilisateur.

L. Microsoft SQL Server

mssql_close (PHP 3, PHP 4 >= 4.0b1)

Ferme une connexion MS SQL Server.

```
bool mssql_close ([resource link_identifier])
```

mssql_close() retourne TRUE en cas de succès, ou FALSE sinon.

mssql_close() ferme la connexion à la base MS SQL Server, qui était associé à l'identifiant *link_identifier*. Si ce dernier n'est pas précisé, la dernière connexion ouverte sera fermée.

Notez qu'il n'est pas nécessaire de fermer les connexions non persistantes aux bases de données, car elles seront fermées automatiquement à la fin du script.

mssql_close() ne peut pas fermer les liens persistants, générés par **mssql_pconnect()**.

Voir aussi **mssql_connect()** et **mssql_pconnect()**.

mssql_connect (PHP 3, PHP 4 >= 4.0b1)

Ouvre une connexion à un serveur MS SQL server.

```
resource mssql_connect ([string servername [, string username [, string password]]])
```

mssql_connect()r etourne un identifiant positif de lien en cas de succès, et FALSE sinon.

mssql_connect() établit une connexion à un serveur MS SQL. Le nom du serveur *servername* doit être valide, comme défini dans les fichiers d'interface.

Si un deuxième appel est fait à **mssql_connect()** avec les mêmes arguments, un nouveau lien ne sera pas retourné, mais le lien déjà ouvert sera retourné.

Le lien avec le serveur sera fermé dès la fin du script, ce qui fait qu'on n'est pas obligé de fermer explicitement la connexion à la fin du script avec **mssql_close()**.

Voir aussi **mssql_pconnect()** et **mssql_close()**.

mssql_data_seek (PHP 3, PHP 4 >= 4.0b1)

Déplace le pointeur interne de ligne.

```
int mssql_data_seek (resource result_identifier, int row_number)
```

mssql_data_seek() retourne TRUE en cas de succès, FALSE en cas d'échec.

mssql_data_seek() déplace le pointeur interne de ligne, dans le résultat *result_identifier*, jusqu'à la ligne *row_number*. Le prochain appel à **mssql_fetch_row()** retournera cette ligne.

Voir aussi **mssql_data_seek()**.

mssql_fetch_array (PHP 3, PHP 4 >= 4.0b1)

Lit une ligne dans un tableau.

```
int mssql_fetch_array (resource result)
```

mssql_fetch_array() retourne un tableau qui contient les valeurs de la ligne lues, en cas de succès, et FALSE en cas d'échec.

mssql_fetch_array() est une version améliorée de **mssql_fetch_row()**. En plus de stocker les données dans un tableau à index numérique, elle les stocke aussi dans un tableau associatif, en utilisant les noms de colonnes comme clé.

Une chose importante à noter est que **mssql_fetch_array()** n'est PAS significativement plus lente que **mssql_fetch_row()**, tandis qu'elle apporte un confort apréciable.

Pour plus de détails, voyez **mssql_fetch_row()**.

mssql_fetch_field (PHP 3, PHP 4 >= 4.0b1)

Lit les inforamtions sur le champs.

```
object mssql_fetch_field (resource result [, int field_offset])
```

mssql_fetch_field() retourne un objet contenant les informations sur un champs.

mssql_fetch_field() sert à lire des informations spécifiques à un champs, dans un résultat de requête. Si l'offset du champs *field_offset* n'est pas précisé, le prochain champs sera analysé.

Les propriétés de l'objet sont :

- name - nom de la colonne. Si la colonne est le résultat d'une fonction, le nom de la colonne sera computed#N, oú #N est un numéro de série.
- column_source - le nom de la table d'oú la colonne est originaire.
- max_length - taille maximale de la colonne
- numeric - 1 si la colonne est numérique

Voir aussi **mssql_field_seek()**.

mssql_fetch_object (PHP 3, PHP 4 >= 4.0b1)

Retourne une ligne sous la forme d'un objet.

```
int mssql_fetch_object (resource result)
```

mssql_fetch_object() retourne un objet dont les propriétés contiennent les valeurs de la ligne, ou FALSE si il n'y a plus de ligne.

mssql_fetch_object() est similaire à **mssql_fetch_array()**, avec un différence : un objet est retourné, au lieu d'un tableau. Indirectement, cela signifie que vous ne pouvez accéder aux données que par leur nom de champs, et pas par leur offset (les nombres sont illégaux comme nom de propriété).

En terme de vitesse, cette fonction est identique à **mssql_fetch_array()**, quasiment aussi rapide que **mssql_fetch_row()** (la différence est non significative).

Voir aussi **mssql_fetch_array()** et **mssql_fetch_row()**.

mssql_fetch_row (PHP 3, PHP 4 >= 4.0b1)

Lit une ligne comme un tableau.

```
array mssql_fetch_row (resource result)
```

mssql_fetch_row() retourne un tableau qui contient les valeurs de la ligne à lire, ou bien FALSE si il n'y a plus de lignes à lire.

mssql_fetch_row() lit une ligne dans le résultat *result* et place les valeurs dans un tableau. Chaque valeur est enregistré dans un élément du tableau, et les indices commencent à 0.

Les appels suivants à **mssql_fetch_row()** retourneront la ligne suivante, ou bien FALSE s'il ne reste plus de lignes.

Voir aussi **mssql_fetch_array()**, **mssql_fetch_object()**, **mssql_data_seek()** et **mssql_result()**.

mssql_field_length (PHP 3>= 3.0.3, PHP 4 >= 4.0b4)

Lit la longueur d'un champs.

```
int mssql_field_length (resource result [, int offset])
```

mssql_field_name (PHP 3>= 3.0.3, PHP 4 >= 4.0b4)

Lit le nom d'un champs.

```
int mssql_field_name (resource result [, int offset])
```

mssql_field_seek (PHP 3, PHP 4 >= 4.0b1)

Fixe l'offset du pointeur de champs.

```
int mssql_field_seek (resource result, int field_offset)
```

mssql_field_seek() modifie la valeur du pointeur de champs. Le prochain appel à **mssql_fetch_field()** qui ne précisera pas de numéro de champs, le champs fixé par **mssql_field_seek()** sera retournée.

Voir aussi **mssql_fetch_field()**.

mssql_field_type (PHP 3>= 3.0.3, PHP 4 >= 4.0b4)

Lit le nom d'un champs.

```
string mssql_field_type (resource result [, int offset])
```

mssql_free_result (PHP 3, PHP 4 >= 4.0b1)

Libère la mémoire.

```
int mssql_free_result (resource result)
```

mssql_free_result() n'a besoin d'être appelé que si on craint d'utiliser trop de mémoire durant une opération. Toutes les ressources liées à un résultat seront libérés par **mssql_free_result()**.

mssql_get_last_message (PHP 3, PHP 4 >= 4.0b1)

Retourne le dernier message d'erreur du serveur (min_message_severity?).

```
string mssql_get_last_message  (void)
```

mssql_min_error_severity (PHP 3, PHP 4 >= 4.0b1)

Fixe le niveau de sévérité des erreurs.

```
void mssql_min_error_severity  (int severity)
```

mssql_min_message_severity (PHP 3, PHP 4 >= 4.0b1)

Fixe le niveau de sévérité des messages d'erreurs.

```
void mssql_min_message_severity  (int severity)
```

mssql_num_fields (PHP 3, PHP 4 >= 4.0b1)

Retourne le nombre de champs dans un résultat.

```
int mssql_num_fields (resource result)
```

mssql_num_fields() retourne le nombre de champs dans un résultat.
Voir aussi **mssql_query()**, **mssql_fetch_field()** et **mssql_num_rows()**.

mssql_num_rows (PHP 3, PHP 4 >= 4.0b1)

Retourne le nombre de lignes dans un résultat.

```
int mssql_num_rows (resource result)
```

mssql_num_rows() retourne le nombre de lignes dans un résultat.
Voir aussi **mssql_query()** et **mssql_fetch_row()**.

mssql_pconnect (PHP 3, PHP 4 >= 4.0b1)

Ouvre une connexion persistante à un serveur MS SQL.

```
resource mssql_pconnect ([string servername [, string username [, string password]]])
```

mssql_pconnect() retourne un identifiant positif de lien MS SQL en cas de succès, et FALSE en cas d'erreur.
mssql_pconnect() se comporte comme **mssql_connect()** mais avec deux différences :

Premièrement, lors de la connexion, la fonction va commencer par rechercher un lien persistant déjà ouvert avec le même hôte, le même nom d'utilisateur, *username* et le même mot de passe *password*. Si un tel lien est trouvé, cet identifiant sera retourné, au lieux d'en ouvrir une autre connexion.

Deuxièmement, la connexion au serveur SQL ne sera pas fermée à la fin du script, mais restera ouverte, pour d'autres utilisations ultérieures (**mssql_close()** ne fermera pas un lien établi avec **mssql_pconnect()**).

C'est pourquoi ce type de lien est dit 'persistant'.

mssql_query (PHP 3, PHP 4 >= 4.0b1)

Envoie une requête SQL.

```
resource mssql_query (string query [, resource link_identifier])
```

mssql_query() retourne un identifiant positif de résultat en cas de succès, ou FALSE sinon.

mssql_query() envoie la requête au serveur courant, associé à l'identifiant *link_identifier* (ou la base par défaut, si il est omis). Si aucun lien n'est ouvert, **mssql_query()** essaiera d'en ouvrir une, en appelant **mssql_connect()**.

Voir aussi **mssql_select_db()** et **mssql_connect()**.

mssql_result (PHP 3, PHP 4 >= 4.0b1)

Get result data.

```
int mssql_result (mssql_query result, int i, mixed field)
```

mssql_result() retourne la valeur de la colonne, à la ligne donnée, dans le résultat MS SQL, ou FALSE en cas d'erreur.

mssql_result() retourne le contenu d'une des cellules d'un résultat MS SQL. Le nom du champs peut être son nom litéral ou son offet, ou encore, le nom de la table + "." + le nom du champs, ou encore la même chose avec le nom de la base de données. Si la colonne a été aliasée, utilisez le nom de l'alias plutôt que celui de la colonne.

Lorsque vous travaillez sur des résultats de grande taille, il vaut mieux utiliser les fonctions qui récupèrent toute une ligne (voir ci après). Comme ces fonctions lisent toutes les valeurs en une passe, elles sont EXTREMEMENT PLUS RAPIDES que **mssql_result()**. De plus, pensez que l'utilisation de l'offset numérique est beaucoup plus rapide que l'utilisation du nom de la colonne.

Alternatives recommandées : **mssql_fetch_row()**, **mssql_fetch_array()** et **mssql_fetch_object()**.

mssql_select_db (PHP 3, PHP 4 >= 4.0b1)

Selectionne la base de données MS SQL.

```
int mssql_select_db (string database_name [, resource link_identifier])
```

mssql_select_db() retourne TRUE en cas de succès, et FALSE en cas d'erreur.

mssql_select_db() sélectionne la base de données active. Si aucun identifiant de connexion n'est fourni, la fonction utilisera la dernière connexion ouverte. Si aucune connexion n'a été ouverte, la fonction essaiera d'en ouvrir une avec **mssql_connect()**, et de l'utiliser.

Tous les appels à **mssql_query()** seront faits dans cette base.

Voir aussi **mssql_connect()**, **mssql_pconnect()** et **mssql_query()**.

LI. Ming pour Flash

> **Avertissement**
>
> Ce module est *EXPERIMENTAL*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

Introduction

Ming est une librairie open-source (LGPL) qui vous permet de créer des animations au format Flash. Ming supporte toutes les fonctionnalités de Flash 4 : les formes (shapes), les gradients, les images bitmaps (JPEG et PNG), les morphing (transformations d'une forme en une autre), les textes, actions, sprites (mini animations), le streaming MP3 et les transformations de couleurs. Le seul ajout futur est celui des événements sons.

Ming n'est pas un acronyme.

Notez que toutes les distances spécifiées (longueurs, distances, tailles...) sont en "twips", c'est-à-dire 20 unités par pixels. C'est plus ou moins arbitraire, car le lecteur Flash fait une mise à l'échelle avec les valeurs qui lui sont fournis dans la balise embed, ou la frame courante si la balise embed n'est pas utilisée.

Ming propose de nombreux avantages par rapport à l'extension swf. Vous pouvez utiliser Ming sur tous les OS où vous pouvez compiler le code, tandis que swf est limité à Windows. Ming vous évite la déconcertante complexité du format SWF, en transformant les éléments des animations en objets PHP. Enfin, Ming est toujours en cours de développement et surveillé par son auteur : si vous souhaitez une nouvelle fonctionnalité, dites le lui : ming@opaque.net (mailto:ming@opaque.net).

Ming et tous les objets cités ont été ajouté en PHP 4.0.5.

Installation

Pour utiliser Ming avec PHP, vous devez d'abord installer la librairy Ming. Le code source et les instructions d'installation sont disponible sur la page d'accueil de Ming : http://www.opaque.net/ming/, avec des exemples un tutorial et l'actualité Ming.

Téléchargez l'archive Ming. Décompressez la et allez dans le dossier Ming. Faites "make", puis "make install".

Cela va compiler le fichier `libming.so` et l'installer dans `/usr/lib/`, et copier `ming.h` into `/usr/include/`. Editez la ligne `PREFIX=` dans le fichier `Makefile` pour indiquer votre dossier d'installation.

Compilation CGI avec PHP (Unix)

mkdir <phpdir>/ext/ming
cp php_ext/* <phpdir>/ext/ming
cd <phpdir>
./buildconf
./configure --with-ming <other config options>
Compilez et installez PHP comme d'habitude.
Redémarrez votre serveur web si nécessaire.

Compilation en module avec PHP (Unix)

téléchargez `php_ming.so.gz`. Décompressez le et copiez le dans votre dossier de modules PHP (vous pouvez trouver votre dossier de module PHP en utilisant la commande **php-config --extension-dir**). Ensuite, ajoutez la ligne

`extension=php_ming.so` dans votre fichier `php.ini`, ou bien ajoutez `dl('php_ming.so');` en haut de tous vos scripts Ming.

Comment utiliser Ming

Ming introduit 13 objet en PHP. Pour les utilisez, vous devez être familez avec les objets.

- **swfmovie()**.
- **swfshape()**.
- **swfdisplayitem()**.
- **swfgradient()**.
- **swfbitmap()**.
- **swffill()**.
- **swfmorph()**.
- **swftext()**.
- **swffont()**.
- **swftextfield()**.
- **swfsprite()**.
- **swfbutton()**.
- **swfaction()**.

SWFMovie (PHP 4 >= 4.0.5)

Crée un objet 'animation'.

```
new swfmovie (void)
```

swfmovie() Crée un objet 'animation', représentant une animation Flash version 4.

SWFMovie a les méthodes suivantes : **swfmovie->output()**,**swfmovie->save()**, **swfmovie->add()**, **swfmovie->remove()**, **swfmovie->nextframe()**, **swfmovie->setbackground()**, **swfmovie->setrate()**, **swfmovie->setdimension()**, **swfmovie->setframes()** et **swfmovie->streammp3()**.

Des exemples d'utilisation dans : **swfdisplayitem->rotateto()**, **swfshape->setline()**, **swfshape->addfill()**... En fait, tous les exemples utilisent cet objet.

SWFMovie->output (unknown)

Envoie votre animation au navigateur.

```
void swfmovie->output (void)
```

swfmovie->output() envoie votre animation au navigateur. En PHP, faîte le précéder de la fonction **header()**.

```
<?php
header('Content-type: application/x-shockwave-flash');
?>
```

Cela indique au navigateur que l'animation qui arrive est en Flash.

Voir aussi **swfmovie->save()**.

Des exemples d'utilisation dans : **swfmovie->streammp3()**, **swfdisplayitem->rotateto()**, **swfaction()**... En fait, tous les exemples utilisent cette méthode.

SWFMovie->save (unknown)

Sauve dans un fichier.

```
void swfmovie->save (string filename)
```

swfmovie->save() sauve votre animation dans le fichier *filename*.

Voir aussi **swfmovie->output()**.

SWFMovie->add (unknown)

Ajoute un objet dans une animation.

```
void swfmovie->add (ressource instance)
```

swfmovie->add() ajoute l'objet *instance* dans l'animation courante. *instance* peut être de n'importe quel type : forme (shape), texte (text), police (font), etc... Ils doivent être ajouté à une animation pour être utilisé.

Pour les objets affichables (formes, textes, boutons, sprites), **swfmovie->add()** retourne un objet **swfdisplayitem()** de la liste d'affichage. Ainsi, vous pouvez ajouter la même forme plusieurs fois dans la même animation, et obtenir des ressources différentes pour chaque instance.

Voir aussi tous les autres objets et **swfmovie->remove()**

Des exemples d'utilisation dans : **swfdisplayitem->rotateto()** et **swfshape->addfill()**.

SWFMovie->remove (unknown)

Supprime un objet d'une animation.

```
void swfmovie->remove (ressource instance)
```

swfmovie->remove() supprime l'objet *instance* de la liste d'affichage, pour l'animation courante. L'objet ne sera plus disponible pour être affiché ou utilisé.

Voir aussi **swfmovie->add()**.

SWFMovie->setbackground (unknown)

Modifie la couleur de fond.

```
void swfmovie->setbackground (int red, int green, int blue)
```

swfmovie->setbackground() modifie la couleur de fond. Pourquoi est-ce que cette fonction n'accepte pas de canal alpha? (réflechissez quelques instants :-)). En fait, cela ne serait pas si stupide : vous pouvez laisser apercevoir le fond HTML à travers l'animation. Il y a un moyen de faire cela, mais cela ne fonctionne qu'avec IE 4. Recherchez sur le site de http://www.macromedia.com/ pour plus de détails.

SWFMovie->setrate (unknown)

Modifie la vitesse de l'animation.

```
void swfmovie->setrate (int rate)
```

swfmovie->setrate() fixe la vitesse de l'animation à *rate* images par secondes. L'animation ralentira d'elle-même si le lecteur Flash ne peut pas afficher suffisamment rapidement, à moins qu'il n'y ait du son en stream, auquel cas les images sont sacrifiées pour garder un son fluide.

SWFMovie->setdimension (unknown)

Modifie les dimensions de l'animation.

```
void swfmovie->setdimension (int width, int height)
```

swfmovie->setdimension() modifie les dimensions de l'animation : *width* est la largeur et *height* la hauteur.

SWFMovie->setframes (unknown)

Modifie le nombre total d'images dans l'animation.

```
void swfmovie->setframes (string numberofframes)
```

swfmovie->setframes() modifie le nombre total d'images dans l'animation, et le fixe à *`numberofframes`*.

SWFMovie->nextframe (unknown)

Passe à l'image suivante.

```
void swfmovie->nextframe (void)
```

swfmovie->setframes() passe à l'image suivante de l'animation.

SWFMovie->streammp3 (unknown)

Envoie un fichier MP3 en streaming.

```
void swfmovie->streammp3 (string mp3FileName)
```

swfmovie->streammp3() envoie le fichier MP3*`mp3FileName`* en stream audio. **swfmovie->streammp3()** n'est pas très robuste, et se prend facilement les pieds dans le tapis (elle peut éviter la balise initiale ID3, mais c'est bien tout). Tout comme swfshape->addjpegfill(), ce n'est pas une fonction stable. Il faudra sÛrement faire un objet séparé, pour gérer les types de son.

Notez que l'animation n'est pas suffisamment intelligente pour ajouter un nombre suffisant d'images, afin de correspondre à la durée totale du stream MP3. Il vous faudra ajouter des images jusqu'à durée de la musique multiplié par le nombre d'images par secondes.

Oui, vous pouvez utiliser Ming pour mettre un rock-'n-roll endiablé dans vos animation. Evitez d'en parler à l'RIAA ou la SACEM.

Exemple 1. Exemple avec swfmovie->streammp3()

```
<?php
  $m = new SWFMovie();
  $m->setRate(12.0);
  $m->streamMp3("distortobass.mp3");
// utilisez vos propres MP3
// assurez-vous d'avoir les droits

// 11.85 secondes avec 12.0 images par seconde = 142 frames
  $m->setFrames(142);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFDisplayItem (unknown)

Crée un nouvel objet d'affichage displayitem.

```
new swfdisplayitem (void)
```

swfdisplayitem() crée un nouvel objet d'affichage displayitem.

C’est là que toute l’animation prend vie. Une fois que vous avez défini une forme, un texte, un sprite ou un bouton, vous l’ajoutez à une animation, puis vous utilisez la ressource retournée pour déplacer, étirer, contracter, faire tourner ou incliner la forme.

SWFDisplayItem a les méthodes suivantes : **swfdisplayitem->move()**, **swfdisplayitem->moveto()**, **swfdisplayitem->scaleto()**, **swfdisplayitem->scale()**, **swfdisplayitem->rotate()**, **swfdisplayitem->rotateto()**, **swfdisplayitem->skewxto()**, **swfdisplayitem->skewx()**, **swfdisplayitem->skewyto() swfdisplayitem->skewyto()**, **swfdisplayitem->setdepth() swfdisplayitem->remove()**, **swfdisplayitem->setname() swfdisplayitem->setratio()**, **swfdisplayitem->addcolor()** et **swfdisplayitem->multcolor()**.

SWFDisplayItem->moveTo (unknown)

Déplace un objet en coordonnées globales.

```
void swfdisplayitem->moveto (int x, int y)
```

swfdisplayitem->moveto() déplace la forme courante jusqu’au point de coordonnées globales (*x*,*y*).

L’objet peut être **swfshape()**, **swfbutton()**, **swftext()** ou **swfsprite()**. Il doit avoir été ajouté à une animation avec la fonction **swfmovie->add()**.

Voir aussi **swfdisplayitem->move()**.

SWFDisplayItem->move (unknown)

Déplace un objet en coordonnées relatives.

```
void swfdisplayitem->move (int dx, int dy)
```

swfdisplayitem->move() déplace la forme courante de *dx* et *dy* unités, depuis sa position courante.

L’objet peut être **swfshape()**, **swfbutton()**, **swftext()** ou **swfsprite()**. Il doit avoir été ajouté à une animation avec la fonction **swfmovie->add()**.

Voir aussi **swfdisplayitem->moveto()**.

SWFDisplayItem->scaleTo (unknown)

Etire un objet en coordonnées globales.

```
void swfdisplayitem->scaleto (int x, int y)
```

swfdisplayitem->scaleto() étire un objet jusqu’au dimensions (*x*,*y*).

L’objet peut être **swfshape()**, **swfbutton()**, **swftext()** ou **swfsprite()**. Il doit avoir été ajouté à une animation avec la fonction **swfmovie->add()**.

Voir aussi **swfdisplayitem->scale()**.

SWFDisplayItem->scale (unknown)

Etire un objet relativement.

```
void swfdisplayitem->scale (int dx, int dy)
```

swfdisplayitem->scale() étire un objet de (*dx*,*dy*), à partir de sa taille courante.

L'objet peut être **swfshape()**, **swfbutton()**, **swftext()** ou **swfsprite()**. Il doit avoir été ajouté à une animation avec la fonction **swfmovie->add()**.

Voir aussi **swfdisplayitem->scaleto()**.

SWFDisplayItem->rotateTo (unknown)

Tourne un objet en angle absolu.

```
void swfdisplayitem->rotateto (double degrees)
```

swfdisplayitem->rotateto() tourne l'objet jusqu'à l'angle absolu*degrees*, en degrés.

L'objet peut être **swfshape()**, **swfbutton()**, **swftext()** ou **swfsprite()**. Il doit avoir été ajouté à une animation avec la fonction **swfmovie->add()**.

Cet exemple amène trois chaînes tournoyantes depuis le fond de l'écran. Plutôt sympa.

Exemple 1. Exemple avec swfdisplayitem->rotateto()

```
<?php
  $thetext =  "ming!";

  $f = new SWFFont("Bauhaus 93.fdb");

  $m = new SWFMovie();
  $m->setRate(24.0);
  $m->setDimension(2400, 1600);
  $m->setBackground(0xff, 0xff, 0xff);

  // Les fonctions avec un nombre d'arguments sont vraiment une bonne idées.
  // Sincèrement!

  function text($r, $g, $b, $a, $rot, $x, $y, $scale, $string)
  {
    global $f, $m;

    $t = new SWFText();
    $t->setFont($f);
    $t->setColor($r, $g, $b, $a);
    $t->setHeight(960);
    $t->moveTo(-($f->getWidth($string))/2, $f->getAscent()/2);
    $t->addString($string);

    // On peut ajouter des propriétés comme pour une variable PHP standard
    // tant que les noms ne sont pas déjà pris.
    // e.g., vous ne pouvez pas utilisez $i->scale, car c'est une fonction.

    $i = $m->add($t);
    $i->x = $x;
    $i->y = $y;
    $i->rot = $rot;
    $i->s = $scale;
    $i->rotateTo($rot);
    $i->scale($scale, $scale);

    // mais les modification sont locales à une fonction, donc il faut
    // retourner l'objet modifié. Pas pratique...

    return $i;
 }
```

```
  function step($i)
  {
    $oldrot = $i->rot;
    $i->rot = 19*$i->rot/20;
    $i->x = (19*$i->x + 1200)/20;
    $i->y = (19*$i->y + 800)/20;
    $i->s = (19*$i->s + 1.0)/20;

    $i->rotateTo($i->rot);
    $i->scaleTo($i->s, $i->s);
    $i->moveTo($i->x, $i->y);

    return $i;
  }

  // Alors? Ça valait la peine, non?

  $i1 = text(0xff, 0x33, 0x33, 0xff, 900, 1200, 800, 0.03, $thetext);
  $i2 = text(0x00, 0x33, 0xff, 0x7f, -560, 1200, 800, 0.04, $thetext);
  $i3 = text(0xff, 0xff, 0xff, 0x9f, 180, 1200, 800, 0.001, $thetext);

  for($i=1; $i<=100; ++$i)
  {
    $i1 = step($i1);
    $i2 = step($i2);
    $i3 = step($i3);

    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

Voir aussi **swfdisplayitem->rotate()**.

SWFDisplayItem->Rotate (unknown)

Fait tourner une forme relativement.

```
void swfdisplayitem->rotate (double ddegrees)
```

swfdisplayitem->rotate() fait tourner la forme de *ddegrees* degrés, en plus de sa rotation courante.

L'objet peut être **swfshape()**, **swfbutton()**, **swftext()** ou **swfsprite()**. Il doit avoir été ajouté à une animation avec la fonction **swfmovie->add()**.

Voir aussi **swfdisplayitem->rotateto()**.

SWFDisplayItem->skewXTo (unknown)

Incline suivant les X.

```
void swfdisplayitem->skewxto (double degrees)
```

swfdisplayitem->skewxto() modifie l'inclinaison (x-skew) à *degrees*. Si *degrees* vaut 1.0, l'angle sera de 45¡, en avant. S'il vaut plus, ce sera plus penché, et s'il vaut moins, ce sera plus droit.

L'objet peut être **swfshape()**, **swfbutton()**, **swftext()** ou **swfsprite()**. Il doit avoir été ajouté à une animation avec la fonction **swfmovie->add()**.

Voir aussi **swfdisplayitem->skewx()**, **swfdisplayitem->skewy()** and **swfdisplayitem->skewyto()**.

SWFDisplayItem->skewX (unknown)

Incline suivant les X relativement.

```
void swfdisplayitem->skewx (double ddegrees)
```

swfdisplayitem->skewx() ajoute *ddegrees* à l'inclinaison courante (x-skew).

L'objet peut être **swfshape()**, **swfbutton()**, **swftext()** ou **swfsprite()**. Il doit avoir été ajouté à une animation avec la fonction **swfmovie->add()**.

Voir aussi **swfdisplayitem->skewx()**, **swfdisplayitem->skewy()** and **swfdisplayitem->skewyto()**.

SWFDisplayItem->skewYTo (unknown)

Incline suivant les Y.

```
void swfdisplayitem->skewyto (double degrees)
```

swfdisplayitem->skewyto() modifie l'inclinaison (y-skew) à *degrees*. Si *degrees* vaut 1.0, l'angle sera de 45¡, en haut. S'il vaut plus, ce sera plus penché, et s'il vaut moins, ce sera plus droit.

L'objet peut être **swfshape()**, **swfbutton()**, **swftext()** ou **swfsprite()**. Il doit avoir été ajouté à une animation avec la fonction **swfmovie->add()**.

Voir aussi **swfdisplayitem->skewy()**, **swfdisplayitem->skewx()** and **swfdisplayitem->skewxto()**.

SWFDisplayItem->skewY (unknown)

Incline suivant les Y relativement.

```
void swfdisplayitem->skewy (double ddegrees)
```

swfdisplayitem->skewy() ajoute *ddegrees* à l'inclinaison courante (y-skew).

L'objet peut être **swfshape()**, **swfbutton()**, **swftext()** ou **swfsprite()**. Il doit avoir été ajouté à une animation avec la fonction **swfmovie->add()**.

Voir aussi **swfdisplayitem->skewyto()**, **swfdisplayitem->skewx()** and **swfdisplayitem->skewxto()**.

SWFDisplayItem->setDepth (unknown)

Modifie la place en profondeur (z-order)

```
void swfdisplayitem->setdepth (double depth)
```

swfdisplayitem->rotate() place l'objet à la profondeur *depth*. Par défaut, l'objet est placé au niveau où il a été ajouté dans l'animation. Les objets les plus anciens sont placés tout en bas, et les nouveaux sont superposés.

L'objet peut être **swfshape()**, **swfbutton()**, **swftext()** ou **swfsprite()**. Il doit avoir été ajouté à une animation avec la fonction **swfmovie->add()**.

SWFDisplayItem->remove (unknown)

Supprime un objet d'une animation

```
void swfdisplayitem->remove (void)
```

swfdisplayitem->remove() supprime cet objet de la liste d'affichage.

L'objet peut être **swfshape()**, **swfbutton()**, **swftext()** ou **swfsprite()**. Il doit avoir été ajouté à une animation avec la fonction **swfmovie->add()**.

Voir aussi **swfmovie->add()**.

SWFDisplayItem->setName (unknown)

Nomme un objet

```
void swfdisplayitem->setname (string name)
```

swfdisplayitem->setname() donne à l'objet courant le nom de *`name`*. Cela servira à repérer les acteurs d'un script d'action. Cela ne sert qu'avec les sprites.

L'objet peut être **swfshape()**, **swfbutton()**, **swftext()** ou **swfsprite()**. Il doit avoir été ajouté à une animation avec la fonction **swfmovie->add()**.

SWFDisplayItem->setRatio (unknown)

Modifie le ratio de l'objet.

```
void swfdisplayitem->setratio (double ratio)
```

swfdisplayitem->setratio() modifie le ratio de l'objet, et le fixe à *`ratio`*. Uniquement utile pour les morphings.

L'objet peut être **swfshape()**, **swfbutton()**, **swftext()** ou **swfsprite()**. Il doit avoir été ajouté à une animation avec la fonction **swfmovie->add()**.

Cet exemple simple effectue un morphing délicat de trois cercles concentriques.

Exemple 1. Exemple swfdisplayitem->setname()

```
<?php

  $p = new SWFMorph();

  $g = new SWFGradient();
  $g->addEntry(0.0, 0, 0, 0);
  $g->addEntry(0.16, 0xff, 0xff, 0xff);
  $g->addEntry(0.32, 0, 0, 0);
  $g->addEntry(0.48, 0xff, 0xff, 0xff);
  $g->addEntry(0.64, 0, 0, 0);
  $g->addEntry(0.80, 0xff, 0xff, 0xff);
  $g->addEntry(1.00, 0, 0, 0);

  $s = $p->getShape1();
  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
```

```
  $f->scaleTo(0.05);
  $s->setLeftFill($f);
  $s->movePenTo(-160, -120);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $g = new SWFGradient();
  $g->addEntry(0.0, 0, 0, 0);
  $g->addEntry(0.16, 0xff, 0, 0);
  $g->addEntry(0.32, 0, 0, 0);
  $g->addEntry(0.48, 0, 0xff, 0);
  $g->addEntry(0.64, 0, 0, 0);
  $g->addEntry(0.80, 0, 0, 0xff);
  $g->addEntry(1.00, 0, 0, 0);

  $s = $p->getShape2();
  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
  $f->scaleTo(0.05);
  $f->skewXTo(1.0);
  $s->setLeftFill($f);
  $s->movePenTo(-160, -120);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $m = new SWFMovie();
  $m->setDimension(320, 240);
  $i = $m->add($p);
  $i->moveTo(160, 120);

  for($n=0; $n<=1.001; $n+=0.01)
  {
    $i->setRatio($n);
    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFDisplayItem->addColor (unknown)

Ajoute une couleur à une transoformation.

```
void swfdisplayitem->addcolor ([int red [, int green [, int blue [, int a]]]])
```

swfdisplayitem->addcolor() ajoute une couleur à la transformations courante. La couleur est donnée sous la forme RGB.

L'objet peut être **swfshape()**, **swfbutton()**, **swftext()** ou **swfsprite()**. Il doit avoir été ajouté à une animation avec la fonction **swfmovie->add()**.

SWFDisplayItem->multColor (unknown)

Multiplie la couleur de transformation.

```
void swfdisplayitem->multcolor ([int red [, int green [, int blue [, int a]]]])
```

swfdisplayitem->multcolor() multiplie la couleur de transformation par les valeurs données.

L'objet peut être **swfshape()**, **swfbutton()**, **swftext()** ou **swfsprite()**. Il doit avoir été ajouté à une animation avec la fonction **swfmovie->add()**.

Cet exemple simple modifie l'atmosphère de votre image, et en fait une scène d'Halloween (utilisez un paysage ou une image claire pour un meilleur effet)

Exemple 1. Exemple avec swfdisplayitem->multcolor()

```
<?php

  $b = new SWFBitmap("backyard.jpg");
  // Utilisez une de vos images
  $s = new SWFShape();
  $s->setRightFill($s->addFill($b));
  $s->drawLine($b->getWidth(), 0);
  $s->drawLine(0, $b->getHeight());
  $s->drawLine(-$b->getWidth(), 0);
  $s->drawLine(0, -$b->getHeight());

  $m = new SWFMovie();
  $m->setDimension($b->getWidth(), $b->getHeight());

  $i = $m->add($s);

  for($n=0; $n<=20; ++$n)
  {
    $i->multColor(1.0-$n/10, 1.0, 1.0);
    $i->addColor(0xff*$n/20, 0, 0);
    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFShape (PHP 4 >= 4.0.5)

Crée une nouvelle forme.

```
new swfshape (void)
```

swfshape() crée une nouvelle forme.

SWFShape a les méthodes suivantes : **swfshape->setline()**, **swfshape->addfill()**, **swfshape->setleftfill()**, **swfshape->setrightfill()**, **swfshape->movepento()**, **swfshape->movepen()**, **swfshape->drawlineto()**, **swfshape->drawline()**, **swfshape->drawcurveto()** et **swfshape->drawcurve()**.

Ce exemple simple dessine un quadrant d'ellipse rouge.

Exemple 1. Exemple avec swfshape()

```
<?php
  $s = new SWFShape();
  $s->setLine(40, 0x7f, 0, 0);
  $s->setRightFill($s->addFill(0xff, 0, 0));
  $s->movePenTo(200, 200);
  $s->drawLineTo(6200, 200);
  $s->drawLineTo(6200, 4600);
  $s->drawCurveTo(200, 4600, 200, 200);

  $m = new SWFMovie();
  $m->setDimension(6400, 4800);
  $m->setRate(12.0);
  $m->add($s);
  $m->nextFrame();

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFShape->setLine (unknown)

Modifie le style de ligne de la forme.

```
void swfshape->setline (int width [, int red [, int green [, int blue [, int a]]]])
```

swfshape->setline() modifie le style de ligne de la forme. *width* est la largeur de la ligne. Si *width* vaut 0, le style est supprimé (et tous les autres arguments sont ignorés). Si *width* > 0, alors la couleur de la ligne devient (*red*, *green*, *blue*). Les couleurs sont représentées en RGB. Le dernier paramètre *a* est optionnel.

swfshape->setline() accepte 1, 4 ou 5 arguments (mais jamais 3 ou 2).

Vous devez déclarer un style avant de l'utiliser (voir exemple).

Cet exemple enfantin dessine une chaîne "!#%*@", dans des couleurs marrantes et un style rigolo.

Exemple 1. Exemple swfshape->setline()

```
<?php
  $s = new SWFShape();
  $f1 = $s->addFill(0xff, 0, 0);
  $f2 = $s->addFill(0xff, 0x7f, 0);
  $f3 = $s->addFill(0xff, 0xff, 0);
  $f4 = $s->addFill(0, 0xff, 0);
  $f5 = $s->addFill(0, 0, 0xff);

  // erreur : il faut déclarer tous les styles avant
  // de les utiliser.
  $s->setLine(40, 0x7f, 0, 0);
  $s->setLine(40, 0x7f, 0x3f, 0);
  $s->setLine(40, 0x7f, 0x7f, 0);
  $s->setLine(40, 0, 0x7f, 0);
  $s->setLine(40, 0, 0, 0x7f);

  $f = new SWFFont('Techno.fdb');

  $s->setRightFill($f1);
  $s->setLine(40, 0x7f, 0, 0);
  $s->drawGlyph($f, '!');
  $s->movePen($f->getWidth('!'), 0);
```

```
  $s->setRightFill($f2);
  $s->setLine(40, 0x7f, 0x3f, 0);
  $s->drawGlyph($f, '#');
  $s->movePen($f->getWidth('#'), 0);

  $s->setRightFill($f3);
  $s->setLine(40, 0x7f, 0x7f, 0);
  $s->drawGlyph($f, '%');
  $s->movePen($f->getWidth('%'), 0);

  $s->setRightFill($f4);
  $s->setLine(40, 0, 0x7f, 0);
  $s->drawGlyph($f, '*');
  $s->movePen($f->getWidth('*'), 0);

  $s->setRightFill($f5);
  $s->setLine(40, 0, 0, 0x7f);
  $s->drawGlyph($f, '@');

  $m = new SWFMovie();
  $m->setDimension(3000,2000);
  $m->setRate(12.0);
  $i = $m->add($s);

// note la chaine est ici!!!
  $i->moveTo(1500-$f->getWidth("!#%*@")/2, 1000+$f->getAscent()/2);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFShape->addFill (unknown)

Ajoute un remplissage plein à la forme.

```
void swfshape->addfill (int red, int green, int blue [, int a])


void swfshape->addfill (SWFbitmap bitmap [, int flags])


void swfshape->addfill (SWFGradient gradient [, int flags])
```

swfshape->addfill() ajoute un remplissage plein à la forme. **swfshape->addfill()** accepte trois différents types d'arguments.

red, *green*, *blue* est une couleur (format RGB). Le dernier paramètre *a* est optionnel.

L'argument *bitmap* est un objet **swfbitmap()**. Le paramètre *flags* peut être l'un des suivants : SWFFILL_CLIPPED_BITMAP ou SWFFILL_TILED_BITMAP. Par défault, c'est SWFFILL_TILED_BITMAP. Je crois.

L'argument *gradient* est un objet **swfgradient()**. L'argument *flags* peut alors prendre l'une des valeurs suivantes : SWFFILL_RADIAL_GRADIENT ou SWFFILL_LINEAR_GRADIENT. Par défaut, c'est SWFFILL_LINEAR_GRADIENT. Cette fois ci, j'en suis sÛr.

swfshape->addfill() retourne un objet **swffill()** à utiliser avec **swfshape->setleftfill()**, et **swfshape->setrightfill()** décrite un peu plus loin.

Voir aussi **swfshape->setleftfill()** et **swfshape->setrightfill()**.

Ceci est un exemple simple qui affiche un cadre sur une bitmap. Ah, il y a un petit bug dans le lecteur Flash : il ne semble pas faire grand cas de la transformation de la seconde forme en morphing. Suivant les specs, la bitmap devrait s'étirer avec la forme dans cet exemple...

Exemple 1. Exemple avec swfshape->addfill()

```
<?php

  $p = new SWFMorph();

  $b = new SWFBitmap("alphafill.jpg");
  // utilisez vos propres bitmaps!
  $width = $b->getWidth();
  $height = $b->getHeight();

  $s = $p->getShape1();
  $f = $s->addFill($b, SWFFILL_TILED_BITMAP);
  $f->moveTo(-$width/2, -$height/4);
  $f->scaleTo(1.0, 0.5);
  $s->setLeftFill($f);
  $s->movePenTo(-$width/2, -$height/4);
  $s->drawLine($width, 0);
  $s->drawLine(0, $height/2);
  $s->drawLine(-$width, 0);
  $s->drawLine(0, -$height/2);

  $s = $p->getShape2();
  $f = $s->addFill($b, SWFFILL_TILED_BITMAP);

  // ces déplacements n'ont aucun effet
  $f->moveTo(-$width/4, -$height/2);
  $f->scaleTo(0.5, 1.0);

  $s->setLeftFill($f);
  $s->movePenTo(-$width/4, -$height/2);
  $s->drawLine($width/2, 0);
  $s->drawLine(0, $height);
  $s->drawLine(-$width/2, 0);
  $s->drawLine(0, -$height);

  $m = new SWFMovie();
  $m->setDimension($width, $height);
  $i = $m->add($p);
  $i->moveTo($width/2, $height/2);

  for($n=0; $n<1.001; $n+=0.03)
  {
    $i->setRatio($n);
    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFShape->setLeftFill (unknown)

Modifie la couleur de rastérisation de gauche.

```
void swfshape->setleftfill (swfgradient fill)
```

```
void swfshape->setleftfill (int red, int green, int blue [, int a])
```

Tout ce sac de noeud fait qu'il y a deux couleurs de remplissage des lignes. Lorsque l'objet est rasterisé, il est pratique de savoir à l'avance quelle sont les remplissages, et le format SWF les demande.

swfshape->setleftfill() affecte à la couleur de rastérisation de gauche, c'est-à-dire l'intérieur d'un objet, si vous définissez les contours d'un objet dans le sens inverse des aiguilles d'une montre. L'objet de remplissage est un objet **swffill()**, retourné par la fonction **swfshape->addfill()** ci-dessus.

Cela semble être le contraire lorsque vous défnissez une forme dans un morphing. Si votre navigateur crashe, essayez de placer le remplissage sur l'autre côté.

Raccourci pour `swfshape->setleftfill($s->addfill($r, $g, $b [, $a]));`.

Voir aussi **swfshape->setrightfill()**.

SWFShape->setRightFill (unknown)

Modifie la couleur de rastériation de droite.

```
void swfshape->setrightfill (swfgradient fill)
```

```
void swfshape->setrightfill (int red, int green, int blue [, int a])
```

Voir aussi **swfshape->setleftfill()**.

Raccourci pour `swfshape->setrightfill($s->addfill($r, $g, $b [, $a]));`.

SWFShape->movePenTo (unknown)

Déplace le stylo.

```
void swfshape->movepento (int x, int y)
```

swfshape->setrightfill() déplace le stylo dans la forme jusqu'au coordonnées globales (*x*,*y*).

Voir aussi **swfshape->movepen()**, **swfshape->drawcurveto()**, **swfshape->drawlineto()** et **swfshape->drawline()**.

SWFShape->movePen (unknown)

Déplace le stylo relativement.

```
void swfshape->movepen (int dx, int dy)
```

swfshape->setrightfill() déplace le stylo dans la forme depuis les coordonnées (current x,current y) jusqu'au coordonnées (current x + *dx*, current y + *dy*), dans l'espace de coordonnées de la forme.

Voir aussi **swfshape->movepento()**, **swfshape->drawcurveto()**, **swfshape->drawlineto()** et **swfshape->drawline()**.

SWFShape->drawLineTo (unknown)

Dessine une ligne.

```
void swfshape->drawlineto (int x, int y)
```

swfshape->setrightfill() dessine une ligne (avec le style courant de ligne, modifié par **swfshape->setline()**) depuis le point courant jusqu'au point (*x*,*y*) dans l'espace de coordonnées de la forme.

Voir aussi **swfshape->movepento()**, **swfshape->drawcurveto()**, **swfshape->movepen()** et **swfshape->drawline()**.

SWFShape->drawLine (unknown)

Dessine une ligne relativement.

```
void swfshape->drawline (int dx, int dy)
```

swfshape->setrightfill() dessine une ligne (avec le style courant de ligne, modifié par **swfshape->setline()**) depuis le point courant, et sur le déplacement de (*dx*,*dy*).

Voir aussi **swfshape->movepento()**, **swfshape->drawcurveto()**, **swfshape->movepen()** et **swfshape->drawlineto()**.

SWFShape->drawCurveTo (unknown)

Dessine une courbe.

```
void swfshape->drawcurveto (int controlx, int controly, int anchorx, int anchory)
```

swfshape->drawcurveto() dessine une courbe quadratique (avec le style courant de ligne, modifié par **swfshape->setline()**) depuis le point courant jusqu'au point (*anchorx*,*anchory*) en utilisant (*controlx*,*controly*) comme point de contrôle. C'est-à-dire qu'il commence en allant vers le point de contrôle, puis se dirigie sur le point d'ancrrage.

Voir aussi **swfshape->drawlineto()**, **swfshape->drawline()**, **swfshape->movepento()** et **swfshape->movepen()**.

SWFShape->drawCurve (unknown)

Dessine une courbe relativement.

```
void swfshape->drawcurve (int controldx, int controldy, int anchordx, int anchordy)
```

swfshape->drawcurve() dessine une courbe quadratique (avec le style courant de ligne, modifié par **swfshape->setline()**) depuis le point courant jusqu'au point (*anchordx*,*anchordy*) relativement au point courant, et en utilisant le point de contrôle (*controldx*,*controldy*). C'est-à-dire qu'il commence en allant vers le point de contrôle, puis se dirigie sur le point d'ancrrage.

Voir aussi **swfshape->drawlineto()**, **swfshape->drawline()**, **swfshape->movepento()** et **swfshape->movepen()**.

SWFGradient (PHP 4 >= 4.0.5)

Crée un objet gradient

```
new swfgradient (void)
```

swfgradient() crée un nouvel objet gradient.

Une fois que vous avez ajouté les couleurs à votre gradient, vous pouvez l'utiliser dans des formes, avec la fonction **swfshape->addfill()**.

SWFGradient a la méthode suivante : **swfgradient->addentry()**.

Cet exemple simple affiche un gradient noir-blanc comme fond, et un gradient concentrique au centre.

Exemple 1. Exemple avec swfgradient()

```
<?php

  $m = new SWFMovie();
  $m->setDimension(320, 240);

  $s = new SWFShape();

  // gradient noir-blanc
  $g = new SWFGradient();
  $g->addEntry(0.0, 0, 0, 0);
  $g->addEntry(1.0, 0xff, 0xff, 0xff);

  $f = $s->addFill($g, SWFFILL_LINEAR_GRADIENT);
  $f->scaleTo(0.01);
  $f->moveTo(160, 120);
  $s->setRightFill($f);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $m->add($s);

  $s = new SWFShape();

  // gradient radial : rouge vers transparent
  $g = new SWFGradient();
  $g->addEntry(0.0, 0xff, 0, 0, 0xff);
  $g->addEntry(1.0, 0xff, 0, 0, 0);

  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
  $f->scaleTo(0.005);
  $f->moveTo(160, 120);
  $s->setRightFill($f);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $m->add($s);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFGradient->addEntry (unknown)

Ajoute une couleur à la liste du gradient.

```
void swfgradient->addentry (double ratio, int red, int green, int blue [, int a])
```

swfgradient->addentry() ajoute une couleur à la liste des couleurs du gradient. *ratio* est un nombre de 0 à 1, qui indique l'ordre d'apparition des couleurs. Vous devez ajouter les couleurs dans l'ordre croissant de ratio.

red, *green*, *blue* représente une couleur, au format RGB. Le dernier paramètre *a* est optionnel.

SWFBitmap (PHP 4 >= 4.0.5)

Crée un objet bitmap

```
new swfbitmap (string filename [, int alphafilename])
```

swfbitmap() crée un objet bitmap à partir d'un fichier JPEG ou DBL, nommé *filename*. *alphafilename* indique un fichier de masque à utiliser comme canal alpha sur une image JPEG.

> **Note :** Seule les JPEG baseline (frame 0) sont supportés. Les baseline optimisée ou les JPEG progressives ne sont pas supportées.

SWFBitmap a les méthodes suivantes : **swfbitmap->getwidth()** et **swfbitmap->getheight()**.

Il n'est pas possible d'importer directement des images PNG, il faut utiliser l'utilitaire de conversion `png2dbl` pour en faire un fichier .dbl ("define bits lossless"). La raison est que l'auteur ne souhaite pas de dépendance avec la librairie PNG. Le fichier d'autoconfiguration devrait régler ce problème, mais il n'est pas encore fait.

Exemple 1. Importation de fichiers PNG sous Ming

```
<?php
  $s = new SWFShape();
  $f = $s->addFill(new SWFBitmap("png.dbl"));
  $s->setRightFill($f);

  $s->drawLine(32, 0);
  $s->drawLine(0, 32);
  $s->drawLine(-32, 0);
  $s->drawLine(0, -32);

  $m = new SWFMovie();
  $m->setDimension(32, 32);
  $m->add($s);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

Et vous pouvez ajouter un masque alpha sur une image JPEG.

Exemple 2. Exemple avec swfbitmap()

```
<?php

  $s = new SWFShape();

  //les fichiers .msk sont générés par l'utilitaire "gif2mask"
  $f = $s->addFill(new SWFBitmap("alphafill.jpg", "alphafill.msk"));
  $s->setRightFill($f);

  $s->drawLine(640, 0);
  $s->drawLine(0, 480);
  $s->drawLine(-640, 0);
  $s->drawLine(0, -480);

  $c = new SWFShape();
  $c->setRightFill($c->addFill(0x99, 0x99, 0x99));
  $c->drawLine(40, 0);
  $c->drawLine(0, 40);
  $c->drawLine(-40, 0);
  $c->drawLine(0, -40);

  $m = new SWFMovie();
  $m->setDimension(640, 480);
  $m->setBackground(0xcc, 0xcc, 0xcc);

  // décide un fond à damier
  for($y=0; $y<480; $y+=40)
  {
    for($x=0; $x<640; $x+=80)
    {
      $i = $m->add($c);
      $i->moveTo($x, $y);
    }

    $y+=40;

    for($x=40; $x<640; $x+=80)
    {
      $i = $m->add($c);
      $i->moveTo($x, $y);
    }
  }

  $m->add($s);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFBitmap->getWidth (unknown)

Retourne la largeur d'une bitmap.

```
int swfbitmap->getwidth (void)
```

swfbitmap->getwidth() retourne la largeur d'une bitmap, en pixels.

Voir aussi **swfbitmap->getheight()**.

SWFBitmap->getHeight (unknown)

Retourne la hauteur d'une bitmap.

```
int swfbitmap->getheight (void)
```

swfbitmap->getheight() retourne la hauteur d'une bitmap, en pixels.

Voir aussi **swfbitmap->getwidth()**.

SWFFill (PHP 4 >= 4.0.5)

Crée un objet de remplissage

swffill() vous permet de transformer une image bitmap ou un gradient. Les objets **swffill()** sont créé par **swfshape->addfill()**.

SWFFill a les méthodes suivantes : **swffill->moveto()**,**swffill->scaleto()**, **swffill->rotateto()**, **swffill->skewxto()** et **swffill->skewyto()**.

SWFFill->moveTo (unknown)

Déplace l'origine de l'objet SWFFill

```
void swffill->moveto (int x, int y)
```

swffill->moveto() déplace l'origine de la forme jusqu'au point de coordonnées globales (*x*,*y*).

SWFFill->scaleTo (unknown)

Modifie l'échelle de la forme

```
void swffill->scaleto (int x, int y)
```

swffill->scaleto() modifie l'échelle de la forme de *x* dans le sens des abscisses et *y* dans le sens des ordonnées.

SWFFill->rotateTo (unknown)

Tourne la forme

```
void swffill->rotateto (double degrees)
```

swffill->rotateto() tourne la forme depuis sont orientation initiale jusqu'à un angle de *degrees* degrés.

SWFFill->skewXTo (unknown)

Incline (abscisses)

```
void swffill->skewxto (double x)
```

swffill->skewxto() incline la forme de *x* suivant l'axe des abscisses. Si *x* vaut 1.0, l'inclinaison sera de 45¡ degrés, en avant. Si *x* vaut plus, l'inclinaison sera plus forte, et sinon, la forme sera plus droite.

SWFFill->skewYTo (unknown)

Incline (ordonnées)

```
void swffill->skewyto (double y)
```

swffill->skewyto() incline la forme de *y* suivant l'axe des abscisses. Si *y* vaut 1.0, l'inclinaison sera de 45¡ degrés, en avant. Si *x* vaut plus, l'inclinaison sera plus forte, et sinon, la forme sera plus droite.

SWFMorph (PHP 4 >= 4.0.5)

Crée un morphing.

```
new swfmorph (void)
```

swfmorph() crée un morphing.

swfmorph() s'appelle aussi "shape tween". C'est cet objet qui permet toutes ces superbes animations qui mettent à genou votre ordinateur. Joie!

Les méthodes ici sont plutôt bizarres. Il serait tellement plus logique d'avoir seulement `new SWFMorph(shape1, shape2);`, mais, telles que sont les choses aujourd'hui, la deuxième forme a besoin de savoir qu'elle est l'aboutissement d'un morphing. (Tout cela, parceque Flash commence à dessiner aussitôt qu'il a les commandes de dessins. S'il conservait les descriptions de ses propres formes, et attendait leur totalité avant d'écrire, ceci et bien d'autres choses serait tellement plus simple).

SWFMorph a les méthodes suivantes : **swfmorph->getshape1()** et **swfmorph->getshape1()**.

Cet exemple simple effectue le morphing d'une gros carré rouge en un carré plus petit, bleu et bordé de noir.

Exemple 1. Exemple avec swfmorph()

```
<?php
  $p = new SWFMorph();

  $s = $p->getShape1();
  $s->setLine(0,0,0,0);

  /* Notez que cela se fait dans l'ordre inverse de l'ordre habituel
     (gauche au lieu de droite), mais je n'ai aucune idée de pourquoi... */

  $s->setLeftFill($s->addFill(0xff, 0, 0));
  $s->movePenTo(-1000,-1000);
  $s->drawLine(2000,0);
  $s->drawLine(0,2000);
  $s->drawLine(-2000,0);
  $s->drawLine(0,-2000);

  $s = $p->getShape2();
  $s->setLine(60,0,0,0);
  $s->setLeftFill($s->addFill(0, 0, 0xff));
  $s->movePenTo(0,-1000);
  $s->drawLine(1000,1000);
  $s->drawLine(-1000,1000);
  $s->drawLine(-1000,-1000);
  $s->drawLine(1000,-1000);
```

```
  $m = new SWFMovie();
  $m->setDimension(3000,2000);
  $m->setBackground(0xff, 0xff, 0xff);

  $i = $m->add($p);
  $i->moveTo(1500,1000);

  for($r=0.0; $r<=1.0; $r+=0.1)
  {
    $i->setRatio($r);
    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFMorph->getshape1 (unknown)

Sélectionne la forme de départ

```
mixed swfmorph->getshape1 (void)
```

swfmorph->getshape1() sélectionne la forme de début de morphing. **swfmorph->getshape1()** retourne un objet **swfshape()**.

SWFMorph->getshape2 (unknown)

Sélectionne la forme de fin

```
mixed swfmorph->getshape2 (void)
```

swfmorph->getshape2() sélectionne la forme de début de morphing. **swfmorph->getshape2()** retourne un objet **swfshape()**.

SWFText (PHP 4 >= 4.0.5)

Crée un nouvel objet texte.

```
new swftext (void)
```

swftext() crée un nouvel objet texte, prêt à être manipulé.

SWFText a les méthodes suivantes : **swftext->setfont()**, **swftext->setheight()**, **swftext->setspacing()**, **swftext->setcolor()**, **swftext->moveto()**, **swftext->addstring()** et **swftext->getwidth()**.

Cet exemple simple va afficher la phrase "PHP fait du Flash avec Ming" sur un fond blanc.

Exemple 1. Exemle avec swftext()

```
<?php
  $f = new SWFFont("Techno.fdb");
  $t = new SWFText();
  $t->setFont($f);
  $t->moveTo(200, 2400);
  $t->setColor(0xff, 0xff, 0);
  $t->setHeight(1200);
  $t->addString("PHP fait du Flash avec Ming!!");

  $m = new SWFMovie();
  $m->setDimension(5400, 3600);

  $m->add($t);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFText->setFont (unknown)

Sélectionne la police courante

```
void swftext->setfont (string font)
```

swftext->setfont() remplace la police courante par *`font`*.

SWFText->setHeight (unknown)

Modifie la hauteur de la police courante

```
void swftext->setheight (int height)
```

swftext->setheight() fixe la hauteur courante de la police courante à *`height`*. Par défaut, c'est 240.

SWFText->setSpacing (unknown)

Modifie l'espacement de police

```
void swftext->setspacing (double spacing)
```

swftext->setspacing() fixe l'espacement de police à *`spacingspacing`*. Par défaut, c'est 1.0. 0 signifie que toutes les lettres seront écrites au même point. Cela fonctionne pas terrible, car l'avance des lettres augmente, et l'espacement entre lettre n'est pas toujours le même. Il faudra que je l'explique plus clairement. Ou bien que je corrige les erreurs.

SWFText->setColor (unknown)

Modifie la couleur de la police

```
void swftext->setcolor (int red, int green, int blue [, int a])
```

swftext->setspacing() change la couleur de la police courante. Par défaut, c'est noir. La couleur est représentée avec la convention RGB.

SWFText->moveTo (unknown)

Déplace le stylo de texte

```
void swftext->moveto (int x, int y)
```

swftext->moveto() déplace le style (ou le curseur, si ça a un sens) jusqu'au coordonnées (*x*,*y*) dans l'espace de coordonnées du texte. Si *x* ou *y* vaut 0, la valeur de coordonnées de la dimension reste la même. C'est ennuyeux, et cela devrait être corrigé.

SWFText->addString (unknown)

Ajoute du texte

```
void swftext->addstring (string string)
```

swftext->addstring() ajoute le texte *string* au texte courant, et le dessine. Le stylo est situé sur la ligne de base du texte, c'est-à-dire que le texte sera écrit horizontalement.

SWFText->getWidth (unknown)

Calcule la longueur d'une chaîne

```
void swftext->addstring (string string)
```

swftext->addstring() retourne la taille de la chaîne *string*, une fois qu'elle est dessinée avec la police et l'espaceement courant.

SWFFont (PHP 4 >= 4.0.5)

Charge une police

```
new swffont (string filename)
```

si *filename* est le nom d'un fichier FDB (i.e., si le nom de fichier se termine par ".fdb"), charge la police.

FDB ("font definition block") est un petit utilitaire pour Flash DefineFont2 qui contient une description complète de la police. Vous pouvez créer des fichiers FDB à partir du "SWT Generator", qui est inclus avec les utilitaires makefdb - regardez dans le dossier utilitaire de Ming.

Les polices utilisateurs ne contiennent aucune information autre que le nom de la police. On suppose que la police sera elle-même accessible au lecteur. Les polices "_serif", "_sans", et "_typewriter" doivent être universellement disponibles. Par exemple :

```
<?php
$f = newSWFFont("_sans");
?>
```

vous donne la police standard "sans-serif", probablement identique à celle que vous obtenez avec le code `<font name="sans-serif">`.

swffont() retourne une ressource de police, à utiliser avec les méthodes **swftext->setfont()** et **swftextfield->setfont()**.

SWFFont a les méthodes suivantes : **swffont->getwidth()**.

swffont->getwidth (unknown)

Retourne la taille de la chaîne

```
int swffont->getwidth (string string)
```

swffont->getwidth() retourne la taille de la chaîne *`string`*, avec la police courante. Vous utiliserez plutôt la même méthode de l'objet **swftext()**, qui utilise les paramètres de l'objet.

SWFTextField (PHP 4 >= 4.0.5)

Crée un nouveau champs texte

```
new swftextfield ([int flags])
```

swftextfield() crée un nouveau champs texte. Les champs textes sont moins souples que les **swftext()**, car ils ne peuvent être tournés, mis à l'échelle ou incliné, mais ils peuvent être utilisés sous forme de champs de formulaire, et ils peuvent utiliser des polices navigateur.

Les flags optionnels modifient les comportements du champs. Ils peuvent prendre les valeurs suivantes :

- SWFTEXTFIELD_NOEDIT : indique que le champs ne doit pas être éditable.
- SWFTEXTFIELD_PASSWORD : indique que c'est un champs mot de passe
- SWFTEXTFIELD_DRAWBOX : dessine le contour du champs
- SWFTEXTFIELD_MULTILINE : autorise les lignes multiples
- SWFTEXTFIELD_WORDWRAP : autorise la mise en forme du texte
- SWFTEXTFIELD_NOSELECT : rend le champs non-sélectionnable

Les flags peuvent être combinés avec l'opérateur OR. Par exemple :

```
<?php
$t = newSWFTextField(SWFTEXTFIELD_PASSWORD | SWFTEXTFIELD_NOEDIT);
?>
```

crée un champs de mot de passe totalement inéditable (et inutile).

SWFTextField a les méthodes suivantes : **swftextfield->setfont()**, **swftextfield->setbounds()**, **swftextfield->align()**, **swftextfield->setheight()**, **swftextfield->setleftmargin()**, **swftextfield->setrightmargin()**, **swftextfield->setmargins()**, **swftextfield->setindentation()**, **swftextfield->setlinespacing()**, **swftextfield->setcolor()**, **swftextfield->setname()** et **swftextfield->addstring()**.

SWFTextField->setFont (unknown)

Modifie la police du champs

```
void swftextfield->setfont (string font)
```

swftextfield->setfont() remplace la police courante par la police *font* (police client?).

SWFTextField->setbounds (unknown)

Sélectionne la largeur et hauteur du champs

```
void swftextfield->setbounds (int width, int height)
```

swftextfield->setbounds() fixe la longueu du champs à *width* et sa hauteur à *height*. Si vous ne fixez pas les bords vous-mêmes, Ming tentera de les deviner lui-même (mais ne le laissez pas faire!!).

SWFTextField->align (unknown)

Modifie l'alignement du texte

```
void swftextfield->align (int alignement)
```

swftextfield->align() change l'alignement du texte par *alignement*. Les valeurs valides pour *alignement* sont : SWFTEXTFIELD_ALIGN_LEFT, SWFTEXTFIELD_ALIGN_RIGHT, SWFTEXTFIELD_ALIGN_CENTER et SWFTEXTFIELD_ALIGN_JUSTIFY.

SWFTextField->setHeight (unknown)

Modifie la hauteur de la police du champs texte.

```
void swftextfield->setheight (int height)
```

swftextfield->setheight() modifie la hauteur de la police du champs texte par *height*. Par défaut, c'est 240.

SWFTextField->setLeftMargin (unknown)

Modifie la marge de gauche.

```
void swftextfield->setleftmargin (int width)
```

swftextfield->setleftmargin() modifie la marge de gauche du champs texte à *width*. Par défaut, c'est 0.

SWFTextField->setrightMargin (unknown)

Modifie la marge de droite.

```
void swftextfield->setrightmargin (int width)
```

swftextfield->setrightmargin() modifie la marge de gauche du champs texte à *`width`*. Par défaut, c'est

SWFTextField->setMargins (unknown)

Modifie les marges du champs texte.

```
void swftextfield->setmargins (int left, int right)
```

swftextfield->setmargins() modifie les deux marges du champs texte : *`left`* sera la nouvelle largeur de la marge de gauche, et *`right`*, celle de gauche. Par défaut, elles sont toutes les deux à 0.

SWFTextField->setindentation (unknown)

Modifie l'indentation de la première ligne.

```
void swftextfield->setindentation (int width)
```

swftextfield->setindentation() modifie l'indentation de la première ligne du champs texte, en la fixant à *`width`*.

SWFTextField->setLineSpacing (unknown)

Modifie l'espacement de lignes.

```
void swftextfield->setlinespacing (int height)
```

swftextfield->setlinespacing() modifie l'espacement de lignes, en le fixant à *`height`*. Par défaut, c'est 40.

SWFTextField->setcolor (unknown)

Modifie la couleur du champs texte

```
void swftextfield->setcolor (int red, int green, int blue [, int a])
```

swftextfield->setcolor() modifie la couleur du champs texte, en la remplaçant par la couleur fournie. Par défaut, c'est noir opaque. Les couleurs sont représentées en convention RGB.

SWFTextField->setname (unknown)

Nomme le champs texte

```
void swftextfield->setname (string name)
```

swftextfield->setname() baptise le champs texte *name*. Cela servira pour les formulaires et les actions.

SWFTextField->addstring (unknown)

Ajoute au texte

```
void swftextfield->addstring (string string)
```

swftextfield->setname() concatène la chaîne *string* avec la chaîne courante.

SWFSprite (PHP 4 >= 4.0.5)

Crée un sprite

```
new swfsprite (void)
```

swfsprite() sont aussi connue sous le nom de "clip" : ils permettent la création d'objet animé dans une animation, avec un scénario propre. De ce fait, un sprite a les mêmes méthodes qu'une animation.

swfsprite() a les méthodes suivantes : **swfsprite->add()**, **swfsprite->remove()**, **swfsprite->nextframe()** et **swfsprite->setframes()**.

Ce exemple pratique fait tourner un superbe carré rouge.

Exemple 1. Exemple de swfsprite()

```
<?php
  $s = new SWFShape();
  $s->setRightFill($s->addFill(0xff, 0, 0));
  $s->movePenTo(-500,-500);
  $s->drawLineTo(500,-500);
  $s->drawLineTo(500,500);
  $s->drawLineTo(-500,500);
  $s->drawLineTo(-500,-500);

  $p = new SWFSprite();
  $i = $p->add($s);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();

  $m = new SWFMovie();
  $i = $m->add($p);
  $i->moveTo(1500,1000);
  $i->setName("blah");

  $m->setBackground(0xff, 0xff, 0xff);
  $m->setDimension(3000,2000);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFSprite->add (unknown)

Ajoute un objet à un sprite

```
mixed swfsprite->add (ressource object)
```

swfsprite->add() ajoute une **swfshape()**, un **swfbutton()**, un **swftext()**, une **swfaction()** ou une autre animation **swfsprite()**.

Pour les objets affichables (**swfshape()**, **swfbutton()**, **swftext()**, **swfaction()** or **swfsprite()**), cela retourne une ressource sur l'objet dans la liste d'affichage.

SWFSprite->remove (unknown)

Supprime un objet dans un sprite

```
void swfsprite->remove (ressource object)
```

swfsprite->remove() supprime une **swfshape()**, un **swfbutton()**, un **swftext()**, une **swfaction()** ou un **swfsprite()** du sprite courant.

SWFSprite->setframes (unknown)

Fixe le nombre maximum d'image dans le sprite.

```
void swfsprite->setframes (int numberofframes)
```

swfsprite->setframes() fixe le nombre total d'images de l'animation à *`numberofframes`*.

SWFSprite->nextframe (unknown)

Va à la prochaine image du sprite.

```
void swfsprite->nextframe (void)
```

swfsprite->setframes() se déplace à la prochaine image du sprite.

SWFbutton (PHP 4 >= 4.0.5)

Crée un nouveau bouton.

```
new swfbutton (void)
```

swfbutton() crée un nouveau bouton. Cliquez-le, passez la souris dessus, et appelez des actions. Facile!

SWFButton a les méthodes suivantes : **swfbutton->addshape()**, **swfbutton->setup()**, **swfbutton->setover()** **swfbutton->setdown()**, **swfbutton->sethit()** **swfbutton->setaction()** et **swfbutton->addaction()**.

Cet exemple simplissime vous montre comme faire un roll-over, un roll-on, un clic, un relaché de souris, et rien du tout (pas d'action).

Exemple 1. Exemple avec swfbutton()

```
<?php

  $f = new SWFFont("_serif");

  $p = new SWFSprite();

  function label($string)
  {
    global $f;

    $t = new SWFTextField();
    $t->setFont($f);
    $t->addString($string);
    $t->setHeight(200);
    $t->setBounds(3200,200);
    return $t;
  }
  function addLabel($string)
  {
    global $p;

    $i = $p->add(label($string));
    $p->nextFrame();
    $p->remove($i);
  }

  $p->add(new SWFAction("stop();"));
  addLabel("NO ACTION");
  addLabel("SWFBUTTON_MOUSEUP");
  addLabel("SWFBUTTON_MOUSEDOWN");
  addLabel("SWFBUTTON_MOUSEOVER");
  addLabel("SWFBUTTON_MOUSEOUT");
  addLabel("SWFBUTTON_MOUSEUPOUTSIDE");
  addLabel("SWFBUTTON_DRAGOVER");
  addLabel("SWFBUTTON_DRAGOUT");

  function rect($r, $g, $b)
  {
    $s = new SWFShape();
    $s->setRightFill($s->addFill($r, $g, $b));
    $s->drawLine(600,0);
    $s->drawLine(0,600);
    $s->drawLine(-600,0);
    $s->drawLine(0,-600);

    return $s;
  }

  $b = new SWFButton();
  $b->addShape(rect(0xff, 0, 0), SWFBUTTON_UP | SWFBUTTON_HIT);
  $b->addShape(rect(0, 0xff, 0), SWFBUTTON_OVER);
  $b->addShape(rect(0, 0, 0xff), SWFBUTTON_DOWN);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(1);"),
        SWFBUTTON_MOUSEUP);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(2);"),
SWFBUTTON_MOUSEDOWN);
```

```
  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(3);"),
SWFBUTTON_MOUSEOVER);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(4);"),
SWFBUTTON_MOUSEOUT);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(5);"),
SWFBUTTON_MOUSEUPOUTSIDE);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(6);"),
SWFBUTTON_DRAGOVER);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(7);"),
SWFBUTTON_DRAGOUT);

  $m = new SWFMovie();
  $m->setDimension(4000,3000);

  $i = $m->add($p);
  $i->setName("label");
  $i->moveTo(400,1900);

  $i = $m->add($b);
  $i->moveTo(400,900);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

Cet exemple simple illustre le déplacement d'un gros bouton rouge dans la fenêtre. Ce n'est pas du tirer-déposer, mais juste du tirer.

Exemple 2. Exemple avec swfbutton->addaction()

```
<?php

  $s = new SWFShape();
  $s->setRightFill($s->addFill(0xff, 0, 0));
  $s->drawLine(1000,0);
  $s->drawLine(0,1000);
  $s->drawLine(-1000,0);
  $s->drawLine(0,-1000);

  $b = new SWFButton();
  $b->addShape($s, SWFBUTTON_HIT | SWFBUTTON_UP | SWFBUTTON_DOWN | SWFBUTTON_OVER);

  $b->addAction(new SWFAction("startDrag('/test', 0);"),
SWFBUTTON_MOUSEDOWN);
// '0' signifie : ne pas verrouiller la souris

  $b->addAction(new SWFAction("stopDrag();"),
SWFBUTTON_MOUSEUP | SWFBUTTON_MOUSEUPOUTSIDE);

  $p = new SWFSprite();
  $p->add($b);
  $p->nextFrame();

  $m = new SWFMovie();
  $i = $m->add($p);
  $i->setName('test');
  $i->moveTo(1000,1000);

  header('Content-type: application/x-shockwave-flash');
```

```
  $m->output();
?>
```

SWFbutton->addShape (unknown)

Ajoute une forme à un bouton

```
void swfbutton->addshape (ressource shape, int flags)
```

swfbutton->addshape() ajoute la forme *shape* au bouton. Les valeurs possibles de *flags* sont : SWFBUTTON_UP, SWFBUTTON_OVER, SWFBUTTON_DOWN ou SWFBUTTON_HIT. SWFBUTTON_HIT n'est même pas affiché, elle désigne la région du clic d'un bouton. C'est-à-dire que tous points où le bouton est déssiné est considéré comme accessible.

SWFbutton->setUp (unknown)

Alias de SWFbutton->addShape(shape, SWFBUTTON_UP)

```
void swfbutton->setup (ressource shape)
```

swfbutton->setup() est un alias pour `SWFbutton->addShape(shape, SWFBUTTON_UP)`.

Voir aussi : **swfbutton->addshape()** et **swfaction()**.

SWFbutton->setOver (unknown)

Alias de addShape(shape, SWFBUTTON_OVER)

```
void swfbutton->setover (ressource shape)
```

swfbutton->setover() est un alias pour `addShape(shape, SWFBUTTON_OVER)`.

Voir aussi : **swfbutton->addshape()** et **swfaction()**.

SWFbutton->setDown (unknown)

Alias de addShape(shape, SWFBUTTON_DOWN)

```
void swfbutton->setdown (ressource shape)
```

swfbutton->setdown() est un alias pour `addShape(shape, SWFBUTTON_DOWN)`.

Voir aussi : **swfbutton->addshape()** et **swfaction()**.

SWFbutton->setHit (unknown)

Alias de addShape(shape, SWFBUTTON_HIT)

```
void swfbutton->sethit (ressource shape)
```

swfbutton->sethit() est un alias pour `addShape(shape, SWFBUTTON_HIT)`.

Voir aussi : **swfbutton->addshape()** et **swfaction()**.

SWFbutton->addAction (unknown)

Ajoute une action au bouton

```
void swfbutton->addaction (ressource action, int flags)
```

swfbutton->setaction() ajoute l'action *`action`* (créée par **swfaction()**) au bouton courant, dans les conditions précisées par *`flags`*. Les valeurs valides de *`flags`* sont : SWFBUTTON_MOUSEOVER, SWFBUTTON_MOUSEOUT, SWFBUTTON_MOUSEUP, SWFBUTTON_MOUSEUPOUTSIDE, SWFBUTTON_MOUSEDOWN, SWFBUTTON_DRAGOUT et SWFBUTTON_DRAGOVER

Voir aussi : **swfbutton->addshape()** et **swfaction()**.

SWFbutton->setAction (unknown)

Assigne l'action du bouton

```
void swfbutton->setaction (ressource action)
```

swfbutton->setaction() assigne l'action qui sera exécutée lorsque le bouton sera cliqué. C'est un alias de `addAction(shape, SWFBUTTON_MOUSEUP)`. *`action`* est une **swfaction()**.

Voir aussi : **swfbutton->addshape()** et **swfaction()**.

SWFAction (PHP 4 >= 4.0.5)

Crée une nouvelle action.

```
new swfaction (string script)
```

swfaction() crée une nouvelle action, et compile le script *`script`*.

La syntaxe du script est basée sur le langage C, mais il utilise aussi beaucoup de notions propres à SWF : le bytecode SWF est trop simpliste pour faire l'essentiel de ce que l'on veut. Par exemple, il n'est pas possible de faire des fonctions sans descendre profondément dans les entrailles de la machine, car le bytecode de saut est écrit en dur. Pas moyen de pousser une adresse dans la pile, ou de dépiler - Chaque fonction doit savoir exactement où elle retourne.

Alors, que reste-t-il? Le compilateur reconnaît les mots suivants :

- break
- for
- continue
- if
- else
- do
- while

Il n'y a pas de type de données : toutes les valeurs de SWF sont stockées comme des chaînes de caractères. Les fonctions suivantes peuvent être utilisées dans les expressions :

time()

Retourne le nombre de milli-secondes depuis le début de l'animation.

random(seed)

Retourne un nombre pseudo-aléatoire, entre 0 et seed.

length(expr)

Retourne la taille de l'expression donnée.

int(number)

Retourne le nombre number, arrondi à l'entier inférieur le plus proche.

concat(expr, expr)

Retourne la concaténation des deux expressions.

ord(expr)

Retourne le code ASCII du caractère expr.

chr(num)

Retourne le caractère pour le code ASCII num.

substr(string, location, length)

Retourne la sous-chaîne, extraite de string, de longueur length et commençant au caractère location.

De plus, les commandes suivantes sont accessibles :

duplicateClip(clip, name, depth)

Duplique le sprite nommé clip. La nouvelle animation a le nouveau nom name et la profondeur depth.

removeClip(expr)

Supprime l'animation nommée expr.

trace(expr)

Ecrit l'expression expr dans le fichier de d'historique. Il est peut probable que le navigateur ou le lecteur ne fasse quoi que ce soit avec.

startDrag(target, lock, [left, top, right, bottom])

Comment à déplacer l'animation target. L'argument lock indique si le déplacement verrouille la souris (utilisez 0, `FALSE`) ou 1 (`TRUE`)). Les paramètres optionnels délimitent la zone de déplacement.

stopDrag()

Cesse le déplacement de l'animation.

callFrame(expr)

Appelle l'image expr comme une fonction.

getURL(url, target, [method])

Charge l'URL url dans l'objet target. target peut être l'image courante (je pense) ou une des valeurs magiques "_level0" (remplace l'animation courante) ou "_level1" (charge une nouvelle animation à la place de la courante). L'argument optionnel method peut être post ou get, si vous voulez envoyer des variables au serveur.

loadMovie(url, target)

La même chose que ci-dessus, plus ou moins. En fait, je ne sais pas trop quelle est la différence.

nextFrame()

Va à l'image suivante.

prevFrame()

Va à l'image précédente.

play()

Joue l'animation.

stop()

Cesse de jouer l'animation.

toggleQuality()

Passe de haute en basse qualité (et vice-versa).

stopSounds()

Cesse de jouer les sons.

gotoFrame(num)

Va à l'image numéro num. Les images sont numérotées à partir de 0.

gotoFrame(name)

Va à l'image nommée name. Ce qui est carrément cool, car les labels ne sont pas encore supportés pour les images.

setTarget(expr)

Modifie le contexte de l'action. C'est ce qu'ils disent, mais je n'ai pas trop d'idées là-dessus.

Et il y a un truc bizarre : l'expression frameLoaded(num) peut être utilise dans les conditions if et dans les boucles while pour vérifier si une image a été chargée. En tous cas, c'est ce qu'il est supposé faire, mais je ne l'ai jamais testé, et je doute sérieusement que cela fonctionne. Vous pouvez utiliser plutôt /:framesLoaded à la place.

Les sprites ont des propriétés. Vous pouvez les lire toutes (vraiment?), en modifier quelques unes. Les voici :

- x
- y
- xScale
- yScale
- currentFrame - (lecture seule)
- totalFrames - (lecture seule)
- alpha - Niveau de transparence
- visible - 1=on, 0=off (?)
- width - (lecture seule)
- height - (lecture seule)
- rotation
- target - (lecture seule) (???)
- framesLoaded - (lecture seule)
- name
- dropTarget - (lecture seule) (???)
- url - (lecture seule) (???)
- highQuality - 1=high, 0=low (?)
- focusRect - (???)
- soundBufTime - (???)

Modifier la position d'un sprite est aussi simple que `/box.x = 100;`. Pourquoi le slash initial? C'est comme cela que Flash garde la trace des sprites dans les animations, un peu comme des fichiers sous Unix. Si le sprite qui s'appelle box a lui-même un autre sprite appelé biff, vous pouvez y accéder avec la commande `/box/biff.x = 100;`. En tous cas, ça marche pour moi. Corrigez moi si c'est faux!.

Cet exemple simple va déplacer le gros carré rouge dans la fenêtre.

Exemple 1. Exemple avec swfaction()

```
<?php
 $s = new SWFShape();
  $f = $s->addFill(0xff, 0, 0);
  $s->setRightFill($f);

  $s->movePenTo(-500,-500);
  $s->drawLineTo(500,-500);
  $s->drawLineTo(500,500);
  $s->drawLineTo(-500,500);
  $s->drawLineTo(-500,-500);

  $p = new SWFSprite();
  $i = $p->add($s);
  $i->setDepth(1);
  $p->nextFrame();

  for($n=0; $n<5; ++$n)
  {
    $i->rotate(-15);
    $p->nextFrame();
  }

  $m = new SWFMovie();
  $m->setBackground(0xff, 0xff, 0xff);
  $m->setDimension(6000,4000);

  $i = $m->add($p);
  $i->setDepth(1);
  $i->moveTo(-500,2000);
  $i->setName("box");

  $m->add(new SWFAction("/box.x += 3;"));
  $m->nextFrame();
  $m->add(new SWFAction("gotoFrame(0); play();"));
  $m->nextFrame();

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

Cet exemple suit votre souris sur l'écran.

Exemple 2. Exemple avec swfaction()

```
<?php

  $m = new SWFMovie();
  $m->setRate(36.0);
  $m->setDimension(1200, 800);
  $m->setBackground(0, 0, 0);

  /* sprite de suivi de souris : vide, mais il suit la souris
  de manière à ce que nous connaissions ses coordonnées*/

  $i = $m->add(new SWFSprite());
  $i->setName('mouse');

  $m->add(new SWFAction("
    startDrag('/mouse', 1); /* '1' signifie verrouiller la souris */
  "));
```

```
/* On peut tout simplement virer l'anti-aliasing, car il n'y a
que des gros carrés, finalement */

$m->add(new SWFAction("
  this.quality = 0;
"));

/* boîte de morphing */
$r = new SWFMorph();
$s = $r->getShape1();

/* Notez que ce n'est pas pratique pour les formes habituelles.
Aucune idée de pourquoi */
$s->setLeftFill($s->addFill(0xff, 0xff, 0xff));
$s->movePenTo(-40, -40);
$s->drawLine(80, 0);
$s->drawLine(0, 80);
$s->drawLine(-80, 0);
$s->drawLine(0, -80);

$s = $r->getShape2();

$s->setLeftFill($s->addFill(0x00, 0x00, 0x00));
$s->movePenTo(-1, -1);
$s->drawLine(2, 0);
$s->drawLine(0, 2);
$s->drawLine(-2, 0);
$s->drawLine(0, -2);


/* sprite contenant la boîte de morphing -
   c'est juste un scénario avec une boîte de morphing */

$box = new SWFSprite();
$box->add(new SWFAction("
  stop();
"));
$i = $box->add($r);

for($n=0; $n<=20; ++$n)
{
  $i->setRatio($n/20);
  $box->nextFrame();
}


/* ce conteneur nous permet d'utiliser la même action plusieurs fois */

$cell = new SWFSprite();
$i = $cell->add($box);
$i->setName('box');

$cell->add(new SWFAction("

  setTarget('box');

  /* ...x signifie abscisse du parent, c'est-à-dire (..).x */
  dx = (/mouse.x + random(6)-3 - ...x)/5;
  dy = (/mouse.y + random(6)-3 - ...y)/5;
  gotoFrame(int(dx*dx + dy*dy));

"));

$cell->nextFrame();
```

```
  $cell->add(new SWFAction("

    gotoFrame(0);
    play();

  "));

  $cell->nextFrame();


  /* finalement, ajoutons quelques cellules à l'animation */

  for($x=0; $x<12; ++$x)
  {
    for($y=0; $y8; ++$y)
    {
      $i = $m->add($cell);
      $i->moveTo(100*$x+50, 100*$y+50);
    }
  }

  $m->nextFrame();

  $m->add(new SWFAction("

    gotoFrame(1);
    play();

  "));

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

La même chose que ci-dessus, mais en couleurs.

Exemple 3. Exemple avec swfaction()

```
<?php

  $m = new SWFMovie();
  $m->setDimension(11000, 8000);
  $m->setBackground(0x00, 0x00, 0x00);

  $m->add(new SWFAction("

this.quality = 0;
/frames.visible = 0;
startDrag('/mouse', 1);

  "));

  // sprite de suivi de souris
  $t = new SWFSprite();
  $i = $m->add($t);
  $i->setName('mouse');

  $g = new SWFGradient();
  $g->addEntry(0, 0xff, 0xff, 0xff, 0xff);
  $g->addEntry(0.1, 0xff, 0xff, 0xff, 0xff);
  $g->addEntry(0.5, 0xff, 0xff, 0xff, 0x5f);
  $g->addEntry(1.0, 0xff, 0xff, 0xff, 0);

  // gradient
```

```
  $s = new SWFShape();
  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
  $f->scaleTo(0.03);
  $s->setRightFill($f);
  $s->movePenTo(-600, -600);
  $s->drawLine(1200, 0);
  $s->drawLine(0, 1200);
  $s->drawLine(-1200, 0);
  $s->drawLine(0, -1200);

  // on en fait un sprite pour utiliser la fonction multColor()
  $p = new SWFSprite();
  $p->add($s);
  $p->nextFrame();

  // Ajoute la forme ici, chaque forme dans une couleur différente
  $q = new SWFSprite();
  $q->add(new SWFAction("gotoFrame(random(7)+1); stop();"));
  $i = $q->add($p);

  $i->multColor(1.0, 1.0, 1.0);
  $q->nextFrame();
  $i->multColor(1.0, 0.5, 0.5);
  $q->nextFrame();
  $i->multColor(1.0, 0.75, 0.5);
  $q->nextFrame();
  $i->multColor(1.0, 1.0, 0.5);
  $q->nextFrame();
  $i->multColor(0.5, 1.0, 0.5);
  $q->nextFrame();
  $i->multColor(0.5, 0.5, 1.0);
  $q->nextFrame();
  $i->multColor(1.0, 0.5, 1.0);
  $q->nextFrame();

  // Enfin, le code de l'action
  $p = new SWFSprite();
  $i = $p->add($q);
  $i->setName('frames');
  $p->add(new SWFAction("

dx = (/:mousex-/:lastx)/3 + random(10)-5;
dy = (/:mousey-/:lasty)/3;
x = /:mousex;
y = /:mousey;
alpha = 100;

  "));
  $p->nextFrame();

  $p->add(new SWFAction("

this.x = x;
this.y = y;
this.alpha = alpha;
x += dx;
y += dy;
dy += 3;
alpha -= 8;

  "));
  $p->nextFrame();

  $p->add(new SWFAction("prevFrame(); play();"));
  $p->nextFrame();
```

```
  $i = $m->add($p);
  $i->setName('frames');
  $m->nextFrame();

  $m->add(new SWFAction("

lastx = mousex;
lasty = mousey;
mousex = /mouse.x;
mousey = /mouse.y;

++num;

if(num == 11)
  num = 1;

removeClip('char' & num);
duplicateClip(/frames, 'char' & num, num);

  "));

  $m->nextFrame();
  $m->add(new SWFAction("prevFrame(); play();"));

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

Cet exemple simple gère le clavier (vous devrez cependant cliquer dans la fenêtre pour lui donner le focus, et vous devrez aussi laisser votre souris dans la fenêtre. Si vous savez comment faire cela automatiquement, dites-le moi!).

Exemple 4. Exemple avec swfaction()

```
<?php

  /* Le sprite n'a qu'une lettre par image */

  $p = new SWFSprite();
  $p->add(new SWFAction("stop();"));

  $chars = "abcdefghijklmnopqrstuvwxyz".
   "ABCDEFGHIJKLMNOPQRSTUVWXYZ".
           "1234567890!@#$%^&*()_+-=/[]{}|;:,.<>¿~";

  $f = new SWFFont("_sans");

  for($n=0; $nremove($i);
    $t = new SWFTextField();
    $t->setFont($f);
    $t->setHeight(240);
    $t->setBounds(600,240);
    $t->align(SWFTEXTFIELD_ALIGN_CENTER);
    $t->addString($c);
    $i = $p->add($t);
    $p->labelFrame($c);
    $p->nextFrame();
  }


  /* région de clic pour le bouton : toute la fenêtre */

  $s = new SWFShape();
  $s->setFillStyle0($s->addSolidFill(0, 0, 0, 0));
  $s->drawLine(600, 0);
```

```
  $s->drawLine(0, 400);
  $s->drawLine(-600, 0);
  $s->drawLine(0, -400);


  /* le bouton s'assure des touches pressées, et envoie le bon
     sprite à l'image de droite */

  $b = new SWFButton();
  $b->addShape($s, SWFBUTTON_HIT);

  for($n=0; $naddAction(new SWFAction("

setTarget('/char');
gotoFrame('$c');

    "), SWFBUTTON_KEYPRESS($c));
  }

  $m = new SWFMovie();
  $m->setDimension(600,400);
  $i = $m->add($p);
  $i->setName('char');
  $i->moveTo(0,80);

  $m->add($b);

  header('Content-type: application/x-shockwave-flash');
  $m->output();

?>
```

LII. Fonctions diverses

Ces fonctions ont été placées là, car elles ne rentraient dans aucune catégorie adéquate.

connection_aborted (PHP 3>= 3.0.7, PHP 4 >= 4.0b4)

Indique si le client a abandonné la connexion.

```
int connection_aborted (void )
```

connection_aborted() retourne `TRUE` si le client a abandonné la connexion. Reportez-vous à Gestion des connexions du chapitre Caractéristiques.

connection_status (PHP 3>= 3.0.7, PHP 4 >= 4.0b4)

Retourne les bits de status de la connexion.

```
int connection_status (void )
```

connection_status() retourne les bits de statut de la connexion. Reportez-vous à la section gestion des connexions pour plus de détails.

connection_timeout (PHP 3>= 3.0.7, 4.0b4 - 4.0.4 only)

Indique si le script a expiré.

```
bool connection_timeout (void )
```

connection_timeout() retourne `TRUE` si le script a expiré. Reportez-vous à la section gestion des connexions pour plus de détails.

define (PHP 3, PHP 4 >= 4.0b1)

Définit une constante.

```
int define (string name, mixed value [, int case_insensitive])
```

define() définit une constante, de la même façon qu'une variable, sauf que :

- Les constantes ne commencent pas par le signe '$'
- Les constantes sont accessibles partout, de manière globale.
- Les constantes ne peuvent pas être redéfinies, ou indéfinies, une fois qu'elles ont été définies.
- Les constantes ne représentent que des valeurs scalaires : il n'est pas possible de définir des tableaux ou des objets.

Le nom de la constante est donné par le paramètre *`name`*; sa valeur est donnée par *`value`*.

Le troisième paramètre optionnel *`case_insensitive`* est une valeur booléenne. S'il vaut `TRUE`, le nom de la constante sera insensible à la casse : `CONSTANT` et `Constant` représentent des valeurs identiques. Par défaut, ces constantes représenteront des valeurs différentes.

Exemple 1. Définition d'une constante

```
<?php
define("CONSTANTE", "Bonjour le monde.");
echo CONSTANTE;
// affiche "Bonjour le monde."
?>
```

define() retoune TRUE en cas de succès et FALSE sinon.

Voir aussi **defined()** et la section sur les constantes.

constant (PHP 4 >= 4.0.4)

Retourne la valeur d'une constante

```
mixed constant (string name)
```

constant() retourne la valeur de la constante *name*.

constant() est pratique lorsque vous devez lire la valeur d'une constante, mais que vous ne savez son nom que durant l'exécution du script. Par exemple, ce nom peut être le résultat d'une fonction.

Exemple 1. Exempe avec constant()

```
<php
  define ("MAXSIZE", 100);
  echo MAXSIZE;
  echo constant("MAXSIZE"); // same thing as the previous line
?>
```

Voir aussi **define()**, **defined()** et la section sur les constantes.

defined (PHP 3, PHP 4 >= 4.0b1)

Vérifie qu'une constante existe.

```
int defined (string name)
```

defined() retourne TRUE si la constante nommée *name* a été définie, et FALSE sinon.

Voir aussi **define()** et la section sur les constantes.

die (unknown)

Affiche un message et termine le script courant

```
void die (string message)
```

die() affiche la chaîne passée en paramètre, puis termine l'exécution du script. Il ne retourne rien de plus.

Exemple 1. Exemple avec die()

```
<?php
$filename = '/path/to/data-file';
$file = fopen ($filename, 'r')
    or die("impossible d'ouvrir le fichier ($filename)");
?>
```

eval (unknown)

Evalue une chaîne comme un script PHP.

```
void eval (string code_str)
```

eval() évalue la chaîne *code_str* comme un script PHP. Parmi les utilisations possibles, cette fonction permet de stocker du code dans une base de données, pour utilisation ultérieure.

Il faut bien garder en tête que le code passé à **eval()** doit être valide, y compris les points virgules de fin de ligne et les séquences d'échappement, sinon l'exécution se terminera.

N'oubliez pas que les variables utilisées dans la fonction **eval()** resteront accessibles dans le script principal.

Exemple 1. Exemple avec eval() - inclusion de texte

```
<?php
$string = 'tasse';
$name = 'café';
$str = 'Ceci est une $string avec mon $name dedans.<br>';
echo $str;
eval( "\$str = \"$str\";" );
echo $str;
?>
```

L'exemple ci-dessus devrait afficher : `Ceci est une $string avec mon $name dedans. Ceci est une tasse avec mon café dedans.`

exit (unknown)

Termine le script courant.

```
void exit (mixed status)
```

exit() termine l'éxécution du script courant. Elle n'a pas de valeur de retour (et pour cause!), mais elle utilisera le message *status* comme message de fin d'exécution.

get_browser (PHP 3, PHP 4 >= 4.0b1)

Indique de quoi est capable le navigateur client.

```
object get_browser ([string user_agent])
```

get_browser() essaie de déterminer les capacités du navigateur client. Cela se fait en lisant les informations dans le fichier `browscap.ini`. Par défaut, la valeur de $HTTP_USER_AGENT est utilisée. Cependant, vous pouvez passer n'importe quelle valeur avec le paramètre optionnel *`user_agent`* à **get_browser()**.

Les informations sont retournées sous forme d'un objet, dont les différents membres contiendront des informations, telles que les versions majeures et mineures et des chaînes d'identification; des booléens pour des caractéristiques telles que frames, JavaScript, et cookies; et ainsi de suite.

Même si `browscap.ini` contient des informations sur de nombreux clients, il compte sur les utilisateurs pour être mis à jour. Le format du fichier est facilement compréhensible.

L'exemple suivant montre comment on peut lister les informations disponibles :

Exemple 1. Exemple avec get_browser()

```
<?php
function list_array ($array) {
    while (list ($key, $value) = each ($array)) {
        $str .= "<B>$key:</B> $value<br>\n";
    }
    return $str;
}
echo "$HTTP_USER_AGENT<hr>\n";
$browser = get_browser();
echo list_array ((array) $browser);
?>
```

L'affichage devrait ressembler à ceci :

```
Mozilla/4.5 [en] (X11; U; Linux 2.2.9 i586)<hr>
<B>browser_name_pattern:</B> Mozilla/4\.5.*<br>
<B>parent:</B> Netscape 4.0<br>
<B>platform:</B> Unknown<br>
<B>majorver:</B> 4<br>
<B>minorver:</B> 5<br>
<B>browser:</B> Netscape<br>
<B>version:</B> 4<br>
<B>frames:</B> 1<br>
<B>tables:</B> 1<br>
<B>cookies:</B> 1<br>
<B>backgroundsounds:</B> <br>
<B>vbscript:</B> <br>
<B>javascript:</B> 1<br>
<B>javaapplets:</B> 1<br>
<B>activexcontrols:</B> <br>
<B>beta:</B> <br>
<B>crawler:</B> <br>
<B>authenticodeupdate:</B> <br>
<B>msn:</B> <br>
```

Pour fonctionner, votre configuration browscap doit mener au fichier `browscap.ini`.

Pour plus d'informations, (y compris pour les endroits où charger le fichier `browscap.ini`), suivez la FAQ PHP à http://www.php.net/FAQ.html (http://www.php.net/FAQ.php).

> **Note :** Browscap a été ajouté en PHP 3.0b2.

highlight_file (PHP 4 >= 4.0b1)

Colorisation de la syntaxe d'un fichier

```
boolean highlight_file (string filename)
```

highlight_file() affiche la syntaxe colorisée du fichier *filename*, en utilisant les couleurs définies dans le moteur interne de PHP.

Exemple 1. Colorisation d'URL

Pour configurer une URL qui peut coloriser n'importe quel script que vous lui passez, nous avons besoin d'utiliser la directive Apache "ForceType", pour générer une URL exploitable, puis utiliser la fonction **highlight_file()** pour afficher un code propre.

Dans votre configuration HTTP `httpd.conf`, vous pouvez ajouter le code suivant :

```
<Location /source>
    ForceType application/x-httpd-php
</Location>
```

Puis, faire un fichier appelé "source", que vous placez dans votre racine de site web.

```
<HTML>
<HEAD>
<TITLE>Affichage de Source</TITLE>
</HEAD>
<BODY BGCOLOR="white"?>
<?php
    $script = getenv ("PATH_TRANSLATED");
    if(!$script) {
    echo "<BR><B>ERROR: Script Name needed</B><BR>";
    } else {
    if (ereg("(\.php|\.inc)$",$script)) {
    echo "<H1>Source of: $PATH_INFO</H1>\n<HR>\n";
    highlight_file($script);
    } else {
    echo "<H1>ERREUR: Seuls les noms de fichier PHP ou de fichiers PH in-
clus sont autorisés</H1>";
    }
    }
    echo "<HR>Traité: ".date("Y/M/d H:i:s",time());
?>
</BODY>
</HTML>
```

Alors, vous pourrez utiliser une URL telle que celle ci-dessous pour afficher une version colorisée de votre script "/path/to/script.php".

```
http://your.server.com/source/path/to/script.php
```

Voir aussi **highlight_string()** et **show_source()**.

highlight_string (PHP 4 >= 4.0b1)

Colorisation d'une chaîne

```
void highlight_string (string str)
```

highlight_string() affiche la version colorisée de la chaîne *str*, en utilisant les couleurs définies dans le moteur interne de PHP.

Voir aussi **highlight_file()** et **show_source()**.

ignore_user_abort (PHP 3>= 3.0.7, PHP 4 >= 4.0b4)

Active l'option décidant si, lors de la déconnexion du client, le script doit poursuivre son exécution ou non.

```
int ignore_user_abort ([int setting])
```

ignore_user_abort() active l'option décidant si, lors de la déconnexion du client, le script doit poursuivre son exécution ou non. La fonction renvoie le paramétrage précédent et elle peut être appelée sans argument pour ne pas changer le paramétrage courant. Voir le paragraphe gestion des connexions dans le chapitre caractéristiques pour une description plus complète des manipulations de connexion en PHP.

iptcparse (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Analyse un bloc binaire IPTC http://www.iptc.org/ et recherche les balises simples.

```
array iptcparse (string iptcblock)
```

iptcparse() analyse un bloc binaire IPTC et recherche les balises simples. **iptcparse()** retourne un tableau avec les balises comme index et les valeurs de ces balises IPTC dans les valeurs de tableau correspondantes. En cas d'erreur, ou si aucune balise IPTC n'a été trouvée, retourne FALSE.

Voir **getimagesize()** pour un exemple.

leak (PHP 3, PHP 4 >= 4.0b1)

Fuite de mémoire.

```
void leak (int bytes)
```

leak() crée une fuite de mémoire.

leak() est pratique pour débugger le gestionnaire de mémoire, qui doit nettoyer automatiquement les fuites de mémoire après chaque requête.

pack (PHP 3, PHP 4 >= 4.0b1)

Compacte des données dans une chaîne binaire.

```
string pack (string format [, mixed args])
```

pack() compacte les arguments dans une chaîne binaire, suivant le format *`format`*. **pack()** retourne la chaîne binaire.

L'idée vient du Perl et tout le formatage fonctionne de la même façon qu'en Perl, mais quelques formats manquent encore (comme, "u"). La chaîne de format est composée d'une série de codes de formats, suivis par un quantificateur optionnel. Le quantificateur peut être un entier, ou * pour la répétition indéfinie. Pour les formats a, A, h et H, le quantificateur spécifie combien de caractères d'un argument sont pris; pour @, c'est la position absolue où placer les données, et pour le reste, c'est le nombre de répétitions. Actuellement, les formats suivants sont implémentés :

- Une chaîne complétée avec `NULL`
- Une chaîne complétée avec espace (SPACE)
- Chaîne hexadécimale h, bit de poids faible en premier.
- Chaîne hexadécimale H, bit de poids fort en premier.
- c caractère signé
- C caractère non signé
- s entier court signé (toujours sur 16 bits, ordre des bits dépendant de la machine).
- S entier court non signé (toujours 16 bits, ordre des bits dépendant de la machine).
- n entier court signé (toujours 16 bits, ordre des bits big endian)
- v entier cours non signé (toujours 16 bits, ordre des bits little endian)
- i entier signé (taille et ordre des bits dépendants de la machine)
- I entier non signé (taille et ordre des bits dépendants de la machine)
- l entier long signé (toujours 32 bits, ordre des bits dépendant de la machine)
- L entier long non signé (toujours 32 bits, ordre des bits dépendant de la machine)
- N entier long non signé (toujours 16 bits, ordre des bits big endian)
- V entier long non signé (toujours 16 bits, ordre des bits little endian)
- f nombre à virgule flottante (taille et représentation dépendantes de la machine)
- d nombre à virgule flottante double (taille et représentation dépendantes de la machine)
- x bit `NULL`
- X recule d'un octet
- @ rempli avec `NULL`, jusqu'à une position absolue

Exemple 1. Compactage d'une chaîne

```
<?php
  $binarydata = pack ("nvc*", 0x1234, 0x5678, 65, 66);
?>
```

La chaîne binaire résultante aura 6 octets de long, et contiendra la séquence 0x12, 0x34, 0x78, 0x56, 0x41, 0x42.

Notez que la distinction entre signé et non signé n'affecte que la fonction **unpack()**, tandis que la fonction **pack()** fournira le même résultat pour les deux formats.

De plus, notez que PHP enregistre de manière interne et intégrale les valeurs : cette représentation dépend de la machine. Si vous essayez d'enregistrer une valeur trop grande, elle risque d'être convertie et de donner lieu à des effets de bords vicieux.

show_source (PHP 4 >= 4.0b1)

Colorisation de la syntaxe d'un fichier

```
void show_source (string filename)
```

show_source() affiche la syntaxe colorisée du fichier *filename*, en utilisant les couleurs définies dans le moteur interne de PHP.

> **Note : show_source()** est un alias de **highlight_file()**

Voir aussi **highlight_string()** et **highlight_file()**.

sleep (PHP 3, PHP 4 >= 4.0b1)

Retarde l'exécution.

```
void sleep (int seconds)
```

sleep() retarde l'exécution du programme pendant *seconds* secondes.

Voir aussi **usleep()**.

uniqid (PHP 3, PHP 4 >= 4.0b1)

Génère un identifiant unique.

```
int uniqid (string prefix [, boolean lcg])
```

uniqid() retourne un identifiant préfixé unique, basé sur l'heure courante, en micro-secondes. Le préfixe peut servir à identifier facilement différents hôtes, si vous générez simultanément des fichiers depuis plusieurs hôtes, à la même micro-seconde. *prefix* peut prendre jusqu'à 114 caractères.

Si le paramètre optionnel *lcg* est TRUE, **uniqid()** ajoutera une entropie "combined LCG" à la fin de la valeur retournée, ce qui renforcera encore l'unicité de l'identifiant.

Sans *prefix* (préfixe vide), la chaîne retournée fera 13 caractères. Si *lcg* est à TRUE, elle fera 23 caractères.

> **Note :** Le paramètre *lcg* est utilisé à partir de PHP 4 et PHP 3.0.13 et ultérieurs.

Si vous voulez utiliser un identifiant unique, ou bien gérer des cookies, il est recommandé d'utiliser un code tel que celui-ci :

```
<?php
$token = md5 (uniqid ("")); // pas de section aléatoire.
$better_token = md5 (uniqid (rand())); // mieux, difficile à deviner
?>
```

Ceci va créer un identifiant de 32 caractères (un nombre hexadécimal de 128) qui sera très difficile à prédire.

unpack (PHP 3, PHP 4 >= 4.0b1)

Déconditionne des données depuis une chaîne binaire.

```
array unpack (string format, string data)
```

unpack() déconditionne des données depuis une chaîne binaire avec le format *`format`*. **unpack()** retourne un tableau contenant les éléments déconditionnés.

unpack() se comporte légèrement différemment de la version Perl car les données déconditionnées sont stockées dans un tableau. Pour cela, il faut donner un nom à chaque format utilisé et les séparer par des slash (/).

Exemple 1. Exemple avec unpack()

```
<?php
$array = unpack ("c2chars/nint", $binarydata);
?>
```

Le tableau résultant contiendra les entrées suivantes : "chars1", "chars2" et "int".

Pour plus de détails, reportez-vous à: **pack()**

Il faut noter que PHP gère les valeurs en interne sous forme signée. Si vous déconditionnez une valeur qui est aussi grande que la taille utilisée en interne par PHP, le résultat se trouvera être un nombre négatif, même s'il a été déconditionné avec l'option " non signé ".

usleep (PHP 3, PHP 4 >= 4.0b1)

Retarde l'exécution en micro-secondes

```
void usleep (int micro_seconds)
```

sleep() retarde l'exécution du programme pendant *`micro_seconds`* micro-secondes.

Voir aussi **sleep()**.

> **Note : sleep()** est inopérante sous Windows

LIII. mnoGoSearch

Ces fonctions donnent l'accès à mnoGoSearch (anciennement UdmSearch), moteur de recherche du monde libre. Pour pouvoir les utiliser, vous devez inclure le support en ajoutant l'option `--with-mnogosearch`. Si vous utilisez cette option sans indiquer le chemin jusqu'à mnogosearch, PHP essaiera de le trouver dans le dossier `/usr/local/mnogosearch`. Si vous avez installé mnogosearch dans un autre endroit, vous devez l'indiquer comme ceci : `--with-mnogosearch=DIR`.

mnoGoSearch est une moteur de recherche complet, destinés aux intranet et serveurs web, distribué sous licence GNU. mnoGoSearch offre des fonctionnalités unique, qui en font un excellent outil pour un grand nombre d'applications de recherche dans votre site : recherche de recettes de cuisines ou dans les journaux, recherche dans un site FTP, dans les groupes de news, etc... Il offre un système d'indexation de textes pour les fichiers HTML, PDF et documents textes. mnoGoSearch est constitué de deux parties : l'indexeur, qui effectue les recherches et le moteur de recherche. L'indexeur passe en revue récursivement les sites HTTP, FTP, NEWS ou encore les fichiers locaux, et enregistre des méta-données dans les bases MySQL, pour optimiser les recherches ultérieures. Une fois que tous les documents ont été référéncés, ils sont accessibles au moteur de recherche. Celui-ci est utilisable par interface web. Les langages C CGI, Perl et PHP sont supportés pour effectuer les recherches.

> **Note :** PHP supporte naturellement MySQL. Il faut savoir que mnoGoSearch n'est pas compatible avec la librairie interne de PHP, et ne peut fonctionner qu'avec les librairies génériques MySQL. Par conséquent, si vous utilisez mnoGoSearch avec MySQL, indiquez le dossier d'installation de MySQL durant la configuration avec l'option : `--with-mnogosearch --with-mysql=/usr`.

Pour utiliser ces fonctions, vous devez installer les versions 3.1.10 ou plus récente de mnoGoSearch.

Plus de détails sur le site officiel de mnoGoSearch : http://www.mnogosearch.ru/.

udm_add_search_limit (PHP 4 >= 4.0.5)

Ajoute différentes limitations de recherche

```
int udm_add_search_limit (int agent, int var, string val)
```

udm_add_search_limit() retourne TRUE en cas de succès, FALSE en cas d'erreur. **udm_add_search_limit()** ajoute différentes limitations de recherche.

agent - un identifiant d'Agent, obtenu après un appel à **udm_alloc_agent()**.

var - nom du paramètre de limitation.

val - Valeur du paramètre sus-cité.

var peut prendre les valeurs suivantes :

- UDM_LIMIT_URL - Définit les limitations sur les URL, pour limiter les recherches à une partie de la base. Ce paramètre supporte les jokers SQL '%' et '_' : '%' remplace n'importe quel nombre de caractères, même zéro caractères, et '_' remplace exactement un caractère. Par exemple, `'http://my.domain.__/catalog'` peut remplacer `http://my.domain.ru/catalog` ou `http://my.domain.ua/catalog`.
- UDM_LIMIT_TAG - Définit les limitations par étiquettes. Lors de l'indexation, vous pouvez assigner des étiquettes sur différentes parties d'un site. Les étiquettes de mnoGoSearch 3.1.x sont des lignes, qui peuvent contenir les jokers '%' et '_' : '%' remplace n'importe quel nombre de caractères, même zéro caractères, et '_' remplace exactement un caractère. Par exemple, si vous avez les étiquettes ABCD et ABCE, la limitation de recherche `ABC_` limitera la recherche à ces deux étiquettes;
- UDM_LIMIT_LANG - Définit les limitations par langue.
- UDM_LIMIT_CAT - Définit les limitations par catégorie. Les catégories sont similaires aux étiquettes, mais elles peuvent être imbriquées. Vous pouvez donc placer des catégories dans d'autres cattégories. Vous devez utiliser deux caractères pour chaque niveau. Vous pouvez utiliser des nombres héxadécimaux allant de 0 à F ou bien sÛr une base de 36, allant de 0 à Z. Par exemple la catégorie supérieure 'Auto' vaut 01. Si elle a une sous catégorie 'Renault', cette dernière sera repérée par 01 (catégorie mère) suivie de 01 (dans sa catégorie), ce qui donne "0101". Si 'Auto' a une autre sous-catégorie 'Peugeot', cette dernière aura le numéro 02, et sera identifiée par `0102`. Si 'Peugeot' a elle-même une autre sous-catégorie, 'Moteur', elle sera numéroté 01, et identifiée uniquement par `010201`. Si vous voulez restreindre les recherches à cette catégorie uniquement, passez `cat=010201`.
- UDM_LIMIT_DATE - Définit les limitations par date de modification du document.

  Format de la valeur : une chaîne de caractères, dont le premier caractère est < ou >, puis une date au format unixtimestamp. Par exemple :

  Udm_Add_Search_Limit($udm,UDM_LIMIT_DATE,"<908012006");

  Si > est utilisé, la recherche sera limitée aux documents dont la date de modification est plus grande que celle qui a été entrée. Avec <, c'est le contraire.

udm_cat_path (PHP 4 >= 4.0.6)

Lit le chemin de la catégorie courante.

```
array udm_cat_path (int agent, string category)
```

udm_cat_path() retourne un tableau listant les catégories depuis la racine jusqu'à la catégorie courante.

Le paramètre *agent* est un identifiant de résultat, obtenu après un appel à *Udm_Alloc_Agent*.

category - La catégorie courante : celle dont on veut le chemin.

udm_cat_path() retourne un tableau avec le format suivant :

Le tableau est constitué de paires. Les index pairs contiennent les chemins de catégories, les index impairs contiennent les noms des catégories correspondantes.

Par exemple, l'appel `$array=udm_cat_path($agent, '02031D');` peut retourner le tableau suivant :

$array[0] contiendra ''
$array[1] contiendra 'Root'
$array[2] contiendra '02'
$array[3] contiendra 'Sport'
$array[4] contiendra '0203'
$array[5] contiendra 'Foot'
$array[4] contiendra '02031D'
$array[5] contiendra 'PSG'

Exemple 1. Spécifier le chemin de la catégorie courante avec le format suivant : '> Root > Sport > Foot > PSG'

```
<?php
  $cat_path_arr=Udm_Cat_Path($udm_agent,$cat);
  $cat_path="";
  for ($i=0; $i<count($cat_path_arr); $i+=2) {
    $path=$cat_path_arr[$i];
    $name=$cat_path_arr[$i+1];
    $cat_path .= " > <a href=\"$PHP_SELF?cat=$path\">$name</a> ";
  }
>
```

udm_cat_list (PHP 4 >= 4.0.6)

Liste toutes les catégories soeurs d'une catégorie.

```
array udm_cat_list (int agent, string category)
```

udm_cat_list() retourne un tableau contenant la liste de toutes les catégories de même niveau que la catégorie courante.

Cette fonction est pratique pour réaliser des arbres à partir des catégories.

udm_cat_list() retourne un tableau avec le format suivant :

Le tableau est constitué de paires. Les index pairs contiennent les chemins de catégories, les index impairs contiennent les noms des catégories correspondantes.

$array[0] contiendra '020300'
$array[1] contiendra 'Marseille'
$array[2] contiendra '020301'
$array[3] contiendra 'Lille'
$array[4] contiendra '020302'
$array[5] contiendra 'Lyon'
...
etc.

Ce qui peut être affiché comme ceci :

Marseille
Lille
Lyon
...

```
<?php
  $cat_list_arr=Udm_Cat_List($udm_agent,$cat);
```

```
  $cat_list=”;
  for ($i=0; $i<count($cat_list_arr); $i+=2) {
    $path=$cat_list_arr[$i];
    $name=$cat_list_arr[$i+1];
    $cat_list .= "<a href=\"$PHP_SELF?cat=$path\">$name</a><br>";
  }
>
```

Udm_Alloc_Agent (PHP 4 >= 4.0.5)

Alloue une session mnoGoSearch

```
int udm_alloc_agent (string dbaddr, string [dbmode])
```

udm_alloc_agent() retourne un agent mnogosearch en cas de succès, FALSE en cas d’erreur. **udm_alloc_agent()** crée une session avec les paramètres de base de données.

dbaddr est une description de base de données formaté comme une URL. Les options (type, hôte, nom de base de données, port, utilisateur ou mot de passe) servent à se connecter à la base de données SQL. Ne passez aucune valeur si vous souhaitez utiliser le support des fichiers texte intégré. Sinon, utilisez le format : `DBAddr DBType:[//[DBUser[:DBPass]@]DBHost[:DBPort]]/DBName/`. Actuellement, les valeurs de DBType possibles sont : mysql, pgsql, msql, solid, mssql, oracle, ibase. En fait, si vous avez ajouté un support natif, cette option est inutile. Mais les utilisateurs ODBC doivent spécifier une des valeurs supportées. Si votre type de base de données n’est pas supporté, utilisez le terme "unknown".

dbmode - Vous pouvez sélectionner le mode de stockage des mots dans la base de données. Si vous indiquez "single", tous les mots seront sotckés dans la même table. Si vous indiquez "multi", les mots seront situés dans différentes tables, suivant leur taille. Le mode "multi" est généralement plus rapide, mais requiert plus de tables. Si le mode "crc" est sélectionné, mnoGoSearch enregistrera un entier de 32 bits, calculé avec l’algorithme CRC32, plutot que des des mots. Ce mode requiert moins d’espace disque, et il est beaucoup plus rapide que les modes "single" et "multi". "crc-multi" utilise la même technique de stockage que le mode "crc", mais il stocke aussi les mots dans différentes tables suivant leur taille. Format: DBMode single/multi/crc/crc-multi.

> **Note :** *dbaddr* et *dbmode* doit correspondre à ceux qui sont utilisés lors de l’indexation.

> **Note :** En réalité, **udm_alloc_agent()** n’ouvre pas de connexion, et donc, ne vérifie ni le nom d’utilisateur, ni le mot de passe.

udm_api_version (PHP 4 >= 4.0.5)

Lit la version des API mnoGoSearch.

```
int udm_api_version ()
```

udm_api_version() retourne le numéro de version des API mnoGoSearch. Par exemple, si mnoGosearch 3.1.10 est utilisé, **udm_api_version()** retournera `30110`.

udm_api_version() permet aux utilisateurs d’identifier quelles sont les API disponibles. Par exemple, **udm_get_doc_count()** n’est disponible qu’à partir de mnoGoSearch 3.1.11.

Exemple avec **udm_api_version()**

```
if (Udm_Api_Version() >= 30111) {
  print  "Total number of urls in database: ".Udm_Get_Doc_Count($udm)."<br>\n";
}
```

udm_clear_search_limits (PHP 4 >= 4.0.5)

Annule toutes les limitations de recherche

```
int udm_clear_search_limits (int agent)
```

udm_clear_search_limits() annule toutes les limitations de recherche imposées, et retourne TRUE.

Udm_Errno (PHP 4 >= 4.0.5)

Numéro d'erreur mnoGoSearch

```
int udm_errno (int agent)
```

udm_errno() retourne le numéro d'erreur mnoGoSearch, ou bien 0 sinon.

Le paramètre *agent* est un identifiant de résultat, obtenu après un appel à *Udm_Alloc_Agent*.

udm_errno() retourne le numéro de l'erreur généré par l'agent *agent*.

Udm_Error (PHP 4 >= 4.0.5)

Message d'erreur mnoGoSearch

```
string udm_error (int agent)
```

udm_errno() retourne le message d'erreur mnoGoSearch, ou bien une chaîne vide sinon.

Le paramètre *agent* est un identifiant de résultat, obtenu après un appel à *Udm_Alloc_Agent*.

udm_error() retourne le numéro de l'erreur généré par l'agent *agent*.

Udm_Find (PHP 4 >= 4.0.5)

Effectue une recherche

```
int udm_find (int agent, string query)
```

udm_add_search_limit() retourne TRUE en cas de succès, et FALSE en cas d'erreur.

La recherche en elle-même. Le premier argument *agent* est la session, le second est la *query*. Pour rechercher, entrez les mots avec lesquels que vous voulez faire une recherche, puis cliquez sur le bouton d'envoi. Par exemple, "mysql odbc". Vous ne devez pas utiliser de guillemets doubles ", car ils sont utilisés par mnoGoSearch pour séparer une requête en mots. Avec l'exemple ci-dessus, mnoGoSearch va rechercher les pages contenant "mysql" et/ou "odbc". Les meilleures réponses seront classées en premier, et affichées en tête de liste. Si vous sélectionnez le mode de recherche "tous" ("ALL"), la recherche va retourner les documents qui contiennent l'un ou l'autre des mots que vous avez entré. Dans le cas où vous utilisez le mode "ANY", la recheche retourne la lsite des documents qui contiennent l'un ou l'autre des mots. Si vous voulez accéder aux fonctions avancées de recherche, vous pouvez utiliser le mode "BOOL", qui vous permet d'entrer directement des requêtes.

mnoGoSearch utilise les opérateurs booléean suivants :

& - AND, ET logique. Par exemple, "mysql & odbc". mnoGoSearch recherche toutes les URL qui contiennent à la fois les mots "mysql" et "odbc".

| - OR, OU logique. Par exemple, "mysql | odbc". mnoGoSearch recherche toutes les URL qui contiennent soit "mysql", soit "odbc".

~ - NOT, NON logique. Par exemple, "mysql & ~odbc". mnoGoSearch recherche toutes les URL qui contiennent le mot "mysql" mais ne contiennent pas le mot "odbc". Attention : la requête "~odbc" ne trouvera rien!

() - Groupage de commandes pour les requêtes complexes : par exemple, "(mysql | msql) & ~postgres". Le mode par requête est simple et puissant à la fois. Vous pouvez utiliser les commandes booléennes habituelles avec ce mode.

Udm_Free_Agent (PHP 4 >= 4.0.5)

Détruit une session mnoGoSearch

```
int udm_free_agent (int agent)
```

udm_free_res() retourne TRUE en cas de succès, FALSE sinon.

Le paramètre *res* est un identifiant de résultat, obtenu après un appel à *Udm-Find*.

udm_free_agent() détruit l'agent de recherche créé par **udm_alloc_agent()**.

udm_free_ispell_data (PHP 4 >= 4.0.5)

Libère la mémoire allouée pour ispell

```
int udm_free_ispell_data (int agent)
```

udm_free_ispell_data() retourne toujours TRUE.

agent - Agent mnoGoSearch obtenu après un appel à **udm_alloc_agent()**.

> **Note : udm_free_ispell_data()** est supportée à partir de la version 3.1.12 de mnoGoSearch et elle ne fait strictement rien avec les versions précédentes.

Udm_Free_Res (PHP 4 >= 4.0.5)

Libère un résultat mnoGoSearch

```
int udm_free_res (int res)
```

udm_free_res() retourne TRUE en cas de succès, FALSE sinon.

Le paramètre *res* est un identifiant de résultat, obtenu après un appel à *Udm-Find*.

udm_free_res() libère la mémoire de tous les résultats générés.

udm_get_doc_count (PHP 4 >= 4.0.5)

Lit le nombre total de documents dans les bases.

```
int udm_get_doc_count (int agent)
```

udm_get_doc_count() retourne le nombre de document dans les bases de données.

agent - Agent mnoGoSearch obtenu après un appel à **udm_alloc_agent()**.

> **Note : udm_get_doc_count()** est supporté à partir de la version mnoGoSearch 3.1.11 ou plus récent.

Udm_Get_Res_Field (PHP 4 >= 4.0.5)

Lit un champs de résultat mnoGoSearch

```
int udm_get_res_field (int res, int row, int field)
```

udm_alloc_agent() retourne la valeur du champs *field* dans la ligne *row*, du résultat *res*, et FALSE sinon.

Le paramètre *res* est un identifiant de résultat, obtenu après un appel à *Udm-Find*.

Le paramètre *row* est le numéro du lien dans la page courante. Il peut valoir de 0 jusqu'à *UDM_PARAM_NUM_ROWS*.

Le paramètre *field* est l'identifiant de champs, et peut prendre l'une des valeurs suivantes :

- UDM_FIELD_URL - Champs URL
- UDM_FIELD_CONTENT - Champs "Content-type" (par exemple, "text/html").
- UDM_FIELD_TITLE - Titre du document.
- UDM_FIELD_KEYWORDS - Mots clés du document (balise META KEYWORDS).
- UDM_FIELD_DESC - Description du document (balise META DESCRIPTION).
- UDM_FIELD_TEXT - Corps du document (balise body, les premières lignes pour donner une idée du document).
- UDM_FIELD_SIZE - Taille du document.
- UDM_FIELD_URLID - Identifiant unique de l'URL.
- UDM_FIELD_RATING - Score de la page (calculé par mnoGoSearch).
- UDM_FIELD_MODIFIED - Date de modification au format unixtimestamp.
- UDM_FIELD_ORDER - Le nombre de documents trouvés.
- UDM_FIELD_CRC - La valeur CRC du document.

Udm_Get_Res_Param (PHP 4 >= 4.0.5)

Lit les paramètres de résultats mnoGoSearch

```
int udm_get_res_param (int res, int param)
```

udm_get_res_param() retourne les paramètres de résultat en cas de succès, FALSE en cas d'erreur.

Le paramètre *res* est un identifiant de résultat, obtenu après un appel à *Udm-Find*.

Le paramètre *param* peut prendre les valeurs suivantes :

- UDM_PARAM_NUM_ROWS - nombre de liens trouvés dans le groupe de résultat courant. C'est la valeur de UDM_PARAM_PAGE_SIZE pour tous les groupes, sauf le dernier.
- UDM_PARAM_FOUND - Nombre total de résultats trouvés.
- UDM_PARAM_WORDINFO - Informations sur les mots trouvés, c'est-à-dire que la recherche "un bon libre" retournera "un: stopword, bon:5637, livre: 120"
- UDM_PARAM_SEARCHTIME - Temps de recherche en secondes

- UDM_PARAM_FIRST_DOC - le numéro du premier document affiché dans le groupe.
- UDM_PARAM_LAST_DOC - le numéro du dernier document affiché dans le groupe.

udm_load_ispell_data (PHP 4 >= 4.0.5)

Charge les données ispell

```
int udm_load_ispell_data (int agent, int var, string val1, string val2, int flag)
```

udm_load_ispell_data() charge des données ispell. **udm_load_ispell_data()** retourne TRUE en cas de succès, et FALSE en cas d'erreur.

agent - Agent mnoGoSearch obtenu après un appel à **udm_alloc_agent()**.

var - paramètre indiquant la source des données ispell.

Après avoir utilisé cette fonction, pensez à libérer les données de la mémoire avec **udm_free_ispell_data()**, même si vous utilisez le mode UDM_ISPELL_TYPE_SERVER.

Le mode de plus rapide est UDM_ISPELL_TYPE_SERVER. UDM_ISPELL_TYPE_TEXT est plus lent, et UDM_ISPELL_TYPE_DB est le plus lent. Ce classement est vrai pour mnoGoSearch 3.1.10 - 3.1.11. Il est prévu d'accélérer le mode DB dans les versions futures, et cela sera plus rapide que le mode TEXT.

- UDM_ISPELL_TYPE_DB indique que les données ispell doivent être chargée depuis la base SQL. Dans ce cas, les paramètres *val1* et *val2* sont ignorés et doivent être laissés vides. *flag* doit valoir 1.

  > **Note :** *flag* indique qu'après le chargement des données ispell à partir de la source, elles doivent être triées (c'est nécessaire au bon fonctionnement d'ispell). Dans le cas où vous chargez les données depuis un fichier, il peut y avoir plusieurs appels à **udm_load_ispell_data()**, et il ne vaut pas la peine de trier les valeurs après chaque appel, mais uniquement à la fin. Etant donné qu'en mode DB, toutes les données sont chargées en une seule fois, ce paramètre doit avoir la valeur de 1. Dans ce mode, en cas d'erreur, par exemple si la table ispell est absente, la fonction retournera FALSE et le code d'erreur, avec son message, seront accessibles avec **udm_error()** et **udm_errno()**.

  Exemple avec **udm_load_ispell_data()**

  ```
  if (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_DB,”,”,1)) {
    printf("Error #%d: ’%s’\n",Udm_Errno($udm),Udm_Error($udm));
    exit;
  }
  ```

- UDM_ISPELL_TYPE_AFFIX indique que les données ispell doivent être chargée depuis un fichier et initie le chargement. Dans ce cas, *val1* définit le code de langue en deux lettre, et *val2* est le chemin jusqu'aux fichiers. Notez que si vous utilisez un chemin relatif, le module recherche les fichiers non pas dans UDM_CONF_DIR, mais directement avec le chemin courant, où le script est exécuté. En cas d'erreur avec ce mode, si le fichier est absent, la fonction retourne FALSE, et un message d'erreur sera affiché. Les messages d'erreur ne sont pas accessibles avec **udm_error()** et **udm_errno()**, puisque ces fonctions ne traitent que les messages SQL. Reportez-vous à la description du paramètre *flag*.

  Exemple avec **udm_load_ispell_data()**

  ```
  if ((! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_AFFIX,’en’,’/opt/ispell/en.aff’,0))
      (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_AFFIX,’ru’,’/opt/ispell/ru.aff’,0))
      (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_SPELL,’en’,’/opt/ispell/en.dict’,0))
      (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_SPELL,’ru’,’/opt/ispell/ru.dict’,1)))
     exit;
  }
  ```

Note : *flag* prend la valeur 1 si c'est le dernier appel à cette fonction.

- UDM_ISPELL_TYPE_SPELL indique que les données ispell doivent être chargées depuis un fichier, et initie le chargement du dictionnaire. Dans ce cas, *val1* définit le code langue sur deux lettres, et *val2* le chemin du fichier. Notez que si vous utilisez un chemin relatif, le module recherche les fichiers non pas dans UDM_CONF_DIR, mais directement avec le chemin courant, où le script est exécuté. En cas d'erreur avec ce mode, si le fichier est absent, la fonction retourne FALSE, et un message d'erreur sera affiché. Les messages d'erreur ne sont pas accessibles avec **udm_error()** et **udm_errno()**, puisque ces fonctions ne traitent que les messages SQL. Reportez-vous à la description du paramètre *flag*.

  Exemple avec **udm_load_ispell_data()**

```
if ((! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_AFFIX,'en','/opt/ispell/en.aff',0))
    (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_AFFIX,'ru','/opt/ispell/ru.aff',0))
    (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_SPELL,'en','/opt/ispell/en.dict',0))
    (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_SPELL,'ru','/opt/ispell/ru.dict',1)))
   exit;
}
```

  Note : *flag* prend la valeur 1 si c'est le dernier appel à cette fonction.

- UDM_ISPELL_TYPE_SERVER active le support des serveurs ispell. *val1* indique alors l'adresse de l'hôte qui supporte le serveur ispall. *val2* n'est pas encore utilisé, mais dans les cas futurs, il indiquera le numéro de port utilisé par le serveur ispell. *flag* n'est pas utile, car les données sont déjà triées.

  Les serveurs Spelld lisent les données d'orthographe dans une configuration séparée (par défaut /usr/local/mnogosearch/etc/spelld.conf), les trie et les stockes en mémoire. Avec les clients, le serveur communique de deux façons : vers les indexeurs, tout le contenu de la mémoire est transféré pour que l'indexeur travaille plus vite; vers le moteur de recherche, il reçoit les mots à normaliser, et les rend au client corrigés. Cela permet une plus grande rapidité d'éxécution, en comparaison des modes db et text (notamment, les tris et les chargements sont beaucoup plus rapides).

  udm_load_ispell_data() en mode UDM_ISPELL_TYPE_SERVER ne charge pas vraiment les données ispell, mais définit simplement l'adresse du serveur. En fait, le serveur sera automatiquement utilisé par **udm_find()** lors des recherches. En cas d'erreur, (par exemple si le serveur ispell ne fonctionne pas ou que l'hôte indiqué est invalide), la conversion sera annulée, mais aucun message d'erreur ne sera affiché.

  Note : Cette fonction est disponible à partir de mnoGoSearch 3.1.12.

  Exemple avec **udm_load_ispell_data()**

```
if (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_SERVER,"","",1)) {
    printf("Error loading ispell data from server<br>\n");
    exit;
}
```

udm_set_agent_param (PHP 4 >= 4.0.5)

Modifie les paramètres de l'agent mnoGoSearch

```
int udm_set_agent_param (int agent, int var, string val)
```

udm_set_agent_param() retourne `TRUE` en cas de succès et `FALSE` sinon. **udm_set_agent_param()** définit les paramètres de l'agent mnoGoSearch.

Les paramètres suivants et leurs valeurs sont disponibles :

- UDM_PARAM_PAGE_NUM - Utilisé pour choisir le numéro de groupe de résultat (les résultats sont retournés par groupe, commençant à 0, avec UDM_PARAM_PAGE_SIZE résultats par page).
- UDM_PARAM_PAGE_SIZE - Nombre de résultats affichés par page.
- UDM_PARAM_SEARCH_MODE - Mode de recherche. Les valeurs suivantes sont disponibles : UDM_MODE_ALL - recherche tous les mots; UDM_MODE_ANY - recherche l'un des mots; UDM_MODE_PHRASE - recherche une phrase; UDM_MODE_BOOL - recherche booléenne. Voir **udm_find()** pour plus de détails sur les recherches booléennes.
- UDM_PARAM_CACHE_MODE - Active/désactive le cache. Lorsque le cache est activé, le moteur de recherche va stocker les résultats sur le disque. Lorque deux requête seront similaires, il pourra retourner les résultats plus rapidement, sans recherche. Valeurs disponibles : UDM_CACHE_ENABLED, UDM_CACHE_DISABLED.
- UDM_PARAM_TRACK_MODE - Active le mode de suivi de requête. Depuis la version 3.1.2, mnoGoSearch dispose d'un suivi de requête. Notez que ce suivi n'est implémenté qu'avec les versions SQL et n'est pas disponible avec les bases de données intégrées. Pour utiliser ce suivi, vous devez créer des tables de suivi. Pour mysql, utilisez le script `create/mysql/track.txt`. Lorsque vous effectuez une recherche avec l'interface, ces tables stockeront les mots recherchés ainsi que le nombre de mots trouvés, et la date. Valeurs disponibles : UDM_TRACK_ENABLED, UDM_TRACK_DISABLED.
- UDM_PARAM_PHRASE_MODE - indique si les index des bases de données utilise des phrases(paramètre "phrase" dans indexer.conf). Valeurs disponibles : UDM_PHRASE_ENABLED and UDM_PHRASE_DISABLED. Notez bien que si la recherche par phrase est activé (UDM_PHRASE_ENABLED), il est toujours possible de faire des recherches dans d'autres modes, (ANY, ALL, BOOL ou PHRASE). En version 3.1.10 de mnoGoSearch, la recherche par phrase n'est supportée que pour les modes SQL et intégré, tandis qu'en 3.1.11, la recherche par phrase est supporté par le mode cache.

  Exemple de recherche par phrase :

  "Arizona desert" - Cette requete retourne tous les documents qui contiennent les mots "Arizona desert" comme une phrase. Notez que vous devez mettre des guillemets doubles autour des phrases.

- UDM_PARAM_CHARSET - Définit le jeu de caractères local. Valeurs disponibles : Tous les jeux supportés par mnoGoSearch. koi8-r, cp1251, ...
- UDM_PARAM_STOPFILE - Définit le nom et le chemin du fichier de mots ignorés. Il y a une petite différence avec mnoGoSearch : Avec mnoGoSearch, si le chemin est `NULL` ou relatif, il est utilisé à partir de UDM_CONF_DIR, alors qu'en PHP, le module va rechercher à partir du chemin courant, c'est-à-dire celui du script courant.
- UDM_PARAM_STOPTABLE - Charge la liste des mots ignorés depuis une table SQL. Vous pouvez utiliser plusieurs tables SQL. Cette commande n'a aucun effet si mnoGoSearch n'a pas été compilé avec le support de base de données.
- UDM_PARAM_WEIGHT_FACTOR - Représente le poids relatif des différentes parties d'un document. Actuellement, le corps, titre, mots clés, descriptions et url sont supportés. Pour activer cette fonctionnalité, utilisez le degré 2 de *Weight commands, dans le fichier `indexer.conf`. Imaginons que vous avez choisi les poids suivants :

  URLWeight 1

  BodyWeight 2

  TitleWeight 4

  KeywordWeight 8

  DescWeight 16

  Comme l'indexeur utiliser l'opérateur de bits OR pour mesurer le poids des mots, il est possible que le même mot soit trouvé plusieurs fois dans le même document lors des recherches. Un mot qui n'apparaît qu'une fois dans le corps sera défini par 00000010 (notation binaire). Un mot qui apparaîtra dans plusieurs parties pourra avoir la notation 00011111.

  La valeur de ce paramètre est une chaîne de chiffres héxadécimaux, sous la forme ABCDE. Chaque chiffre est un facteur correspondant à un poids affecté à une partie du document. Por la situation décrite ci-dessus,

  est le facteur de poids 1 (URL)

est le facteur de poids 2 (Corps)

est le facteur de poids 4 (Titre)

est le facteur de poids 8 (Mots clés)

est le facteur de poids 16 (Description)

Exemples:

UDM_PARAM_WEIGHT_FACTOR=00001 ne recherche que dans les URL.

UDM_PARAM_WEIGHT_FACTOR=00100 ne recherche que dans les Titres.

UDM_PARAM_WEIGHT_FACTOR=11100 recherche dans les Titres,Mots-clés,Description mais pas dans le corps ou les URL.

UDM_PARAM_WEIGHT_FACTOR=F9421 recherche dans :

Description avec un poids de 15 (F hex)

Keywords avec un poids de 9

Title avec un poids de 4

Body avec un poids de 2

URL avec un poids de 1

Si UDM_PARAM_WEIGHT_FACTOR est omis, la valeur par défaut est utilisée.

- UDM_PARAM_WORD_MATCH - Recherche des mots. Vous pouvez utiliser ce paramètre pour choisir le type de recherche de mots. Cette fonctionnalité n'est valable qu'en mode "single" et "multi", avec les bases SQL ou intégrée. Elle ne fonctionne pas en mode intégré, ni avec d'autres modes, car les CRC ne supportent pas les recherches de sous-chaînes. Les valeurs disponibles sont :

  UDM_MATCH_BEGIN - début de mot;

  UDM_MATCH_END - fin de mot;

  UDM_MATCH_WORD - tout le mot;

  UDM_MATCH_SUBSTR - une sous-partie de mots.

- UDM_PARAM_MIN_WORD_LEN - définit les tailles extrêmes de mots. Tout mot plus court que la limite inférieur est ignoré. Notez que ce paramètre est inclusif, c'est-à-dire que si UDM_PARAM_MIN_WORD_LEN=3, un mot de 3 caractères ne sera pas ignoré, alors qu'un mot de 2 caractères sera ignoré. Par défaut, la valeur est de 1.
- UDM_PARAM_ISPELL_PREFIXES - Valeurs possibles : UDM_PREFIXES_ENABLED et UDM_PREFIXES_DISABLED. Ces valeurs activent et désactivent le support des préfixes. Par exemple, si le mot "testé" est placé dans la requête de recherche, les mots tels que "test", "tester", etc.. seront aussi recherchés. Les suffixes sont supportés par défaut. Les préfixes modifie généralement le sens des mots. Par exemple, si vous cherchez "testé", vous ne souhaitez pas trouver "protesté" ou "contesté". Le support des préfixes peut cependant être utilisé pour des raisons d'orthographe. Pour activer ispell, vous devez charger les données ispell avec la fonction **udm_load_ispell_data()**.
- UDM_PARAM_CROSS_WORDS - Active ou désactive le support "CROSS_WORDS". Valeurs possibles : UDM_CROSS_WORDS_ENABLED et UDM_CROSS_WORDS_DISABLED.

  La fonctionnalité "CROSS_WORDS" vous permet d'effectuer des recherches dans les balises (entre), pour utiliser le nom du lien. Ce mode fonctionne avec les bases de données SQL et n'est pas supporté par les modes intégrés ou le cache.

  Note : CROSS_WORDS est supporté à partir de mnoGoSearch 3.1.11.

- UDM_PARAM_VARDIR - spécifie un chemin spécifique sur le disque où l'indexeur enregistre les données lorsqu'il utilise le cache et les bases de données internes. Par défaut, le dossier `/var` de l'installation de mnoGoSearch est utilisé. Ce paramètre est disponible en PHP 4.0.7 et plus récent.

LIV. mSQL

Ces fonctions vous permettent d'accéder aux bases de données mSQL. Pour cela, vous devez compiler PHP avec le support msql, en utilisant l'option de configuration `--with-msql[=dir]`. Par défaut, le chemin est '/usr/local/Hughes'.

Plus d'informations sur mSQL à http://www.hughes.com.au/.

msql (PHP 3, PHP 4 >= 4.0b1)

Exécute une requête mSQL.

```
resource msql (string database, string query, resource link_identifier)
```

msql() retourne un identifiant positif de résultat de requête, ou FALSE en cas d'erreur.

msql() sélectionne la base de données *database*, et y exécute la requête *query*. Si l'identifiant de connexion *link_identifier* n'est pas fourni, la fonction va rechercher un lien ouvert à un serveur mSQL, et sinon, il va tenter d'en créer une, avec **msql_connect()**, sans argument.

msql_affected_rows (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Retourne le nombre de ligne affectées.

```
int msql_affected_rows (resource query_identifier)
```

msql_affected_rows() retourne le nombre de lignes affectées par la dernière commande INSERT, UPDATE ou DELETE sur le serveur associé au *link_identifier*. Si ce dernier n'est pas précisé, la dernière connexion est utilisée.

Voir aussi **msql_query()**.

msql_close (PHP 3, PHP 4 >= 4.0b1)

Ferme une connexion mSQL.

```
int msql_close (resource link_identifier)
```

msql_close() retourne TRUE en cas de succès, FALSE en cas d'erreur.

msql_close() ferme la connexion au serveur de base de données mSQL référencé par l'identifiant fourni. Si aucun identifiant n'est fourni, la dernière connexion sera utilisée.

Notez bien qu'il n'est pas toujours nécessaire d'appeler cette fonction, car les connexions non persistantes seront automatiquement fermées à la fin du script.

msql_close() ne peut pas fermer les connexions persistantes, générées par **msql_pconnect()**.

Voir aussi: **msql_connect()** et **msql_pconnect()**.

msql_connect (PHP 3, PHP 4 >= 4.0b1)

Ouvre une connexion mSQL.

```
resource msql_connect (string [hostname], string [hostname[:port]], string [username],
string [password])
```

msql_connect() retourne un identifiant de connexion positif en cas de succès, et FALSE sinon.

msql_connect() établit une connexion à un serveur mSQL. Le nom d'hôte est optionnel, et lorsqu'il manque, localhost est utilisé.

Si un deuxième appel est fait à **msql_connect()**, avec les mêmes arguments, ce ne sera pas une nouvelle connexion qui va être ouverte, mais l'ancienne connexion qui sera utilisée, et son identifiant sera retourné.

Le lien au serveur sera fermé dès la fin du script, ou bien, manuellement, lors de l'appel de **msql_close()**.

Voir aussi **msql_pconnect()** et **msql_close()**.

msql_create_db (PHP 3, PHP 4 >= 4.0b1)

Crée une base de données mSQL.

```
int msql_create_db (string database name [, resource link_identifier])
```

msql_create_db() essaie de créer une nouvelle base de données sur le serveur référencé par l'identifiant *link_identifier*.

Voir aussi: **msql_drop_db()**.

msql_createdb (PHP 3, PHP 4 >= 4.0b1)

Crée une base de données mSQL.

```
int msql_createdb (string database name [, resource link_identifier])
```

Identique à **msql_create_db()**.

msql_data_seek (PHP 3, PHP 4 >= 4.0b1)

Déplace le pointeur interne.

```
int  (resource query_identifier, int row_number)
```

msql_data_seek() retourne TRUE en cas de succès, et FALSE en cas d'échec.

msql_data_seek() déplace le pointeur interne de résultat mSQL, et le place à l'offset donné. Le prochain appel à la fonction to **msql_fetch_row()** retournera cette ligne.

Voir aussi: **msql_fetch_row()**.

msql_dbname (PHP 3, PHP 4 >= 4.0b1)

Lit le nom de la base de données courante.

```
string msql_dbname (resource query_identifier, int i)
```

msql_dbname() retourne le nom de la base de données enregistré en position *i* du pointeur de résultat retourné par la fonction **msql_listdbs()**. La fonction **msql_numrows()** peut être utilisée pour déterminer le nombre de bases disponibles.

msql_drop_db (PHP 3, PHP 4 >= 4.0b1)

Efface une base de données mSQL.

```
int msql_drop_db (string database_name, int link_identifier)
```

msql_drop_db() retourne TRUE en cas de succès, et FALSE en cas d'échec.

msql_drop_db() essaie d'effacer une base de données entière sur le serveur référencé par l'identifiant fourni.

Voir aussi **msql_create_db()**.

msql_dropdb (PHP 3, PHP 4 >= 4.0b1)

Efface une base de données mSQL.

Voir **msql_drop_db()**.

msql_error (PHP 3, PHP 4 >= 4.0b1)

Retourne le message d'erreur

```
string msql_error (void)
```

Les erreurs générées par mSQL ne sont plus traitées comme des alertes. Au lieu de cela, elles sont stockées, et accessibles à partir de cette fonction.

msql_fetch_array (PHP 3, PHP 4 >= 4.0b1)

Lit une ligne sous la forme d'un tableau.

```
int msql_fetch_array (resource query_identifier [, int result_type])
```

msql_fetch_array() retourne un tableau qui contient la ligne demandée, ou FALSE, si il n'y a pas d'autres lignes.

msql_fetch_array() est une version évoluée de **msql_fetch_row()**. En plus d'enregistrer les données dans un tableau à indice numérique, il peut enregistrer les données dans un tableau associatif, en utilisant les noms des champs comme clés.

Le deuxième argument *result_type* de **msql_fetch_array()** est une constante, et peut prendre les valeurs suivantes : MSQL_ASSOC, MSQL_NUM, et MYSQL_BOTH.

Méfiez vous des requêtes qui retournent une ligne qui ne contient qu'un champs de valeur 0 (ou NULL, ou chaîne vide).

Il est important de noter que **msql_fetch_array()** est marginalement plus lent que **msql_fetch_row()**, alors qu'elle apporte un confort d'utilisation appréciable.

Voir aussi **msql_fetch_row()**.

msql_fetch_field (PHP 3, PHP 4 >= 4.0b1)

Lit la valeur d'un champs.

```
object msql_fetch_field (resource query_identifier, int field_offset)
```

msql_fetch_field() retourne un objet contenant les informations sur un champs.

msql_fetch_field() sert à lire les informations sur les champs, dans certaines requêtes. Si l'offset du champs n'est pas spécifié, le prochain champs sera retourné.

Les propriétés de l'objet sont :

- name - nom de la colonne
- table - nom de la table à qui appartient la colonne.

- not_null - 1 si la colonne ne peut être `NULL`
- primary_key - 1 si la colonne est une clé primaire
- unique - 1 la colonne est une clé unique
- type - le type de la colonne

Voir aussi **msql_field_seek()**.

msql_fetch_object (PHP 3, PHP 4 >= 4.0b1)

Lit une ligne sous la forme d'un objet.

```
int msql_fetch_object (resource query_identifier [, int result_type])
```

msql_fetch_object() retourne un objet, dont les propriétés seront affectées suivant les champs de la ligne lue, ou `FALSE` si il ne reste plus de lignes.

msql_fetch_object() est identique à **msql_fetch_array()**, avec une différence : c'est un objet qui est retourné, à la place d'un tableau. Par conséquent, cela signifie que vous ne pouvez accéder aux valeurs que par les noms des champs, et non plus avec leur offset. (les nombres sont interdits dans les noms de propriétés)

L'argument optionnel *`result_type`* de **msql_fetch_array()** est une constante qui peut prendre les valeurs suivantes : MSQL_ASSOC, MSQL_NUM, et MSQL_BOTH.

msql_fetch_object() est aussi rapide que **msql_fetch_array()**, et marginalement plus lente que **msql_fetch_row()** (la différence est non significative).

Voir aussi **msql_fetch_array()** et **msql_fetch_row()**.

msql_fetch_row (PHP 3, PHP 4 >= 4.0b1)

Retourne une ligne sous la forme d'un objet.

```
array msql_fetch_row (int query_identifier)
```

msql_fetch_row() retourne un tableau qui contient la ligne demandée, ou `FALSE`, si il n'y a plus de lignes à lire.

msql_fetch_row() retourne une ligne, extraite du résultat associé à l'identifiant de résultat *`query_identifier`*. La ligne est retournée sous la forme d'un tableau. Chaque résultat est enregistré dans un champs, indexé numériquement, à partir de 0.

Les appels ultérieurs à **msql_fetch_row()** retourneront les lignes suivantes, ou `FALSE`, lorsqu'il n'y aura plus de ligne.

Voir aussi **msql_fetch_array()**, **msql_fetch_object()**, **msql_data_seek()** et **msql_result()**.

msql_fieldname (PHP 3, PHP 4 >= 4.0b1)

Lit le nom d'un champs.

```
string msql_fieldname (resource query_identifier, int field)
```

msql_fieldname() retourne le nom du champs à l'index *`field`*. *`query_identifier`* est un identifiant de résultat, et *`field`* est un index de champs. `msql_fieldname($result, 2);` retournera le nom du deuxième champs, dans le résultat associé à *`query_identifier`*.

msql_field_seek (PHP 3, PHP 4 >= 4.0b1)

Fixe d'offset d'un champs.

```
int msql_field_seek (resource query_identifier, int field_offset)
```

msql_field_seek() recherche l'offset du champs *field_offset*. Le prochain appel à **msql_fetch_field()** qui d'aura pas d'argument *field_offset*, retournera ce champs.

Voir aussi **msql_fetch_field()**.

msql_fieldtable (PHP 3, PHP 4 >= 4.0b1)

Retourne le nom d'une table à partir d'un nom de champs.

```
int msql_fieldtable (resource query_identifier, int field)
```

msql_fieldtable() retourne le nom de la table d'où est le champs *field* a été extrait.

msql_fieldtype (PHP 3, PHP 4 >= 4.0b1)

Retourne le type de champs.

```
string msql_fieldtype (resource query_identifier, int i)
```

msql_fieldtype() est similaire à **msql_fieldname()**. Les arguments sont identiques, mais c'est le type du champs qui est retourné. Cela produira un résultat tel que "int", "string" ou "real".

msql_fieldflags (PHP 3, PHP 4 >= 4.0b1)

Retourne le flag d'un champs.

```
string msql_fieldflags (resource query_identifier, int i)
```

msql_fieldflags() retourne le flag du champs spécifié. Actuellement, il peut valoir soit "not `NULL`", "primary key", ou une combinaison des deux ou "" (chaîne vide).

msql_fieldlen (PHP 3, PHP 4 >= 4.0b1)

Retourne la longueur d'un champs.

```
int msql_fieldlen (resource query_identifier, int i)
```

msql_fieldlen() retourne la longueur du champs *i*.

msql_free_result (PHP 3, PHP 4 >= 4.0b1)

Libère le résultat de la mémoire.

```
int msql_free_result (resource query_identifier)
```

msql_free_result() libère de la mémoire le résultat associé à l'identifiant de résultat *query_identifier*. Lorsque PHP a terminé une requête, cette mémoire est libérée, ce qui fait que vous n'aurez pas besoin de cette fonction. Vous pouvez toujours l'utiliser pour vous assurez que vous n'utilisez pas trop de mémoire durant un script.

msql_freeresult (PHP 3, PHP 4 >= 4.0b1)

Libère le résultat de la mémoire.

Voir **msql_free_result()**

msql_list_fields (PHP 3, PHP 4 >= 4.0b1)

Liste les champs d'une table.

```
int msql_list_fields (string database, string tablename)
```

msql_list_fields() lit les informations de la table *tablename*. Les arguments sont le nom de la base de données, *database* et le nom de la table *tablename*. Cette fonction retourne un identifiant de résultat qui sera utilisé avec **msql_fieldflags()**, **msql_fieldlen()**, **msql_fieldname()** et **msql_fieldtype()**. Un identifiant de résultat est un entier positif. La fonction retourne -1 si une erreur survient. Une chaîne décrivant l'erreur sera placée dans la variable `$phperrmsg`, et à moins que cette fonction n'ai été appelée avec @ (`@msql_list_fields()`), alors cette erreur sera affichée.

Voir aussi **msql_error()**.

msql_listfields (PHP 3, PHP 4 >= 4.0b1)

Liste les champs d'une table.

Voir **msql_list_fields()**.

msql_list_dbs (PHP 3, PHP 4 >= 4.0b1)

Liste les bases de données mSQL sur un serveur.

```
int msql_list_dbs (void)
```

msql_list_dbs() retournea un pointeur de résultat, qui contiendra les noms des bases de données disponibles sur la connexion mSQL courante. Utilisez **msql_dbname()** pour passer en revue toutes les lignes.

msql_listdbs (PHP 3, PHP 4 >= 4.0b1)

Liste les bases de données mSQL sur un serveur.

Voir **msql_list_dbs()**.

msql_list_tables (PHP 3, PHP 4 >= 4.0b1)

Liste les tables mSQL sur une base de données

```
int msql_list_tables (string database)
```

msql_list_tables() prend un nom de base de données, et fourni un résultat, un peu comme la fonction **msql()**. La fonction **msql_tablename()** devrait être utilisée de préférence pour extraire les nom de table d'un pointeur de résultat.

msql_listtables (PHP 3, PHP 4 >= 4.0b1)

Liste les tables mSQL sur une base de données.

Voir **msql_list_tables()**.

msql_num_fields (PHP 3, PHP 4 >= 4.0b1)

Retourne le nombre de champs dans un résultat.

```
int msql_num_fields (resource query_identifier)
```

msql_num_fields() retourne le nombre de champs du résultat associé à l'identifiant *query_identifier*.

Voir aussi **msql()**, **msql_query()**, **msql_fetch_field()** et **msql_num_rows()**.

msql_num_rows (PHP 3, PHP 4 >= 4.0b1)

Retourne le nombre de lignes dans un résultat.

```
int msql_num_rows (resource query_identifier)
```

msql_num_rows() retourne le nombre de lignes du résultat associé à l'identifiant *query_identifier*.

Voir aussi: **msql()**, **msql_query()** et **msql_fetch_row()**.

msql_numfields (PHP 3, PHP 4 >= 4.0b1)

Retourne le nombre de champs dans un résultat.

```
int msql_numfields (resource query_identifier)
```

Identique à **msql_num_fields()**.

msql_numrows (PHP 3, PHP 4 >= 4.0b1)

Retourne le nombre de lignes dans un résultat.

```
int msql_numrows (void)
```

Identique à **msql_num_rows()**.

msql_pconnect (PHP 3, PHP 4 >= 4.0b1)

Ouvre une connexion persistante à un serveur mSQL.

```
int msql_pconnect (string [hostname], string [hostname[:port]], string [username], string [password])
```

Retourne un identifiant de connexion persistante à un serveur mSQL en cas de succès, et FALSE sinon.

msql_pconnect() se comporte presque comme **msql_connect()** mais avec deux différences majeures.

D'abord, lors de la connexion, **msql_pconnect()** cherche si une connexion persistante a déjà été ouverte sur le même hôte. Si une telle connexion est trouvée, elle sera utilisée.

Deuxièmement, la connexion au serveur SQL ne sera pas terminée lors de la fin de l'exécution du script. A la place, le lien restera ouvert pour d'autres connexions futures. (**msql_close()** ne fermera pas un lien ouvert par **msql_pconnect()**).

C'est pourquoi une telle connexion est considérée comme 'persistante'.

msql_query (PHP 3, PHP 4 >= 4.0b1)

Envoie une requête mSQL

```
resource msql_query (string query, int link_identifier)
```

msql_query() envoie une requête à la base de donnée active, sur le serveur associé à l'identifiant de connexion *link_identifier*. Si *link_identifier* n'est pas fourni, PHP tentera d'utiliser la dernière connexion ouverte. Si aucune connexion n'a été ouverte, la fonction tentera de se connecter par elle même, avec **msql_connect()** appelé sans argument.

Retourne un identifiant positif mSQL en cas de succès, et FALSE sinon.

Voir aussi: **msql()**, **msql_select_db()**, et **msql_connect()**.

msql_regcase (PHP 3, PHP 4 >= 4.0b1)

Prépare une chaîne pour une recherche par expression régulière insensible à la casse.

Voir **sql_regcase()**.

msql_result (PHP 3, PHP 4 >= 4.0b1)

Retourne les données de résultat.

```
int msql_result (resource query_identifier, int i, mixed field)
```

msql_result() retourne la valeur de la cellule, à la ligne *i* et l'offset spécifié, *field* dans le résultat mSQL *query_identifier*.

msql_result() retourne le contenu d'une cellule depuis un résultat mSQL *query_identifier*. L'argument de champs *field* peut être aussi bien un offset, qu'un nom de champs, ou encore le nom de la table point le nom du fichier (nom_table.nom_champs). Si la colonne est un alias, (par exemple 'select foo as bar from...'), utilisez de préférence l'alias au nom de colonne.

Lorsque vous travailler sur des résutats de grande taille, il est préférable d'utiliser les fonctions qui récupèrent toute la ligne (voir ci dessous). Comme ces fonctions retournent plusieurs cellules en même temps, elles sont beaucoup plus rapide que **msql_result()**. De plus, sachez qu'accéder à un champs avec son indice numérique est beaucoup plus rapide qu'en utilisant les autres méthodes.

Alternatives recommandées : **msql_fetch_row()**, **msql_fetch_array()** et **msql_fetch_object()**.

msql_select_db (PHP 3, PHP 4 >= 4.0b1)

Sélectionne une base de données mSQL.

```
int msql_select_db (string database_name, resource link_identifier)
```

msql_select_db() retourne `TRUE` en cas de succès, et `FALSE` en cas d'erreur.

msql_select_db() choisi la base de données courante sur le serveur associé à l'identifiant de connexion *`link_identifier`*. Si *`link_identifier`* n'est pas fourni, PHP tentera d'utiliser la dernière connexion ouverte. Si aucune connexion n'a été ouverte, la fonction tentera de se connecter par elle-même, avec **msql_connect()** appelée sans argument.

Les prochains appels à **msql_query()** seront fait dans la base de données active.

Voir aussi **msql_connect()**, **msql_pconnect()** et **msql_query()**.

msql_selectdb (PHP 3, PHP 4 >= 4.0b1)

Sélectionne une base de données mSQL.

Voir **msql_select_db()**.

msql_tablename (PHP 3, PHP 4 >= 4.0b1)

Retourne le nom d'une table à partir d'un nom de champs.

```
string msql_tablename (int query_identifier, int field)
```

msql_tablename() prend un pointeur de résultat (retourné par la fonction **msql_list_tables()**), ainsi qu'un index, et retourne le nom d'une table. La fonction **msql_numrows()** peut servir à déterminer le nombre de table dans le pointeur de résultat.

Exemple 1. Exemple msql_tablename()

```
<?php
msql_connect ("localhost");
$result = msql_list_tables ("limousin");
$i = 0;
while ($i < msql_numrows ($result)) {
    $tb_names[$i] = msql_tablename ($result, $i);
    echo $tb_names[$i] . "<BR>";
    $i++;
}
?>
```

LV. MySQL

Ces fonctions vous permettent d'accéder aux bases de données MySQL. Afin de pouvoir les utiliser, vous devez compiler PHP avec le support MySQL, en utilisant l'option `--with-mysql`. Si vous utilisez cette fonction sans préciser le chemin d'accès à la base MySQL, PHP utilisera les librairies cliente MySQL fournies en standard. Les utilisateurs qui font tourner d'autres applications qui utilisent elles-mêmes MySQL (par exemple, PHP 3 et PHP 4 utilisés comme des modules concurents apache, ou encore auth-mysql), devrait toujours spécifier le chemin jusqu'à MySQL : `--with-mysql=/path/to/mysql`. Cela va forcer PHP à utiliser les librairies clientes installées par MySQL et évitera les conflits.

Plus d'informations sont disponibles à http://www.mysql.com/.

La documentation de MySQL est disponibles à http://www.mysql.com/documentation/, ainsi qu'en français chez nexen (http://dev.nexen.net/docs/mysql/).

mysql_affected_rows (PHP 3, PHP 4 >= 4.0b1)

Retourne le nombre de lignes affectées lors de la dernière requête SQL.

```
int mysql_affected_rows (resource [link_identifier])
```

mysql_affected_rows() retourne le nombre de lignes affectées lors de la dernière requête INSERT, UPDATE ou DELETE sur le serveur associé à l'identifiant de connexion. Si cet identifiant n'est pas précisé, **mysql_affected_rows()** utilise la dernière connexion ouverte.

> **Note :** Si vous utilisez les transactions, vous devez appeler **mysql_affected_rows()** après votre INSERT, UPDATE, ou DELETE et non après la validation.

Si la dernière requête était un DELETE sans clause WHERE, tous les enregistrements ont été effacés, mais **mysql_affected_rows()** va retourner 0.

mysql_affected_rows() n'est pas possible après un SELECT, car elle ne fonctionne qu'après des commandes qui modifient les enregistrements. Pour connaître le nombre de lignes retournées par un SELECT, utilisez **mysql_num_rows()**.

Si la dernière requête a échoué, **mysql_affected_rows()** retourne -1.

mysql_change_user (PHP 3>= 3.0.13)

Change le nom de session de l'utilisateur actif.

```
int mysql_change_user (string user, string password, string [database], resource
[link_identifier])
```

mysql_change_user() change l'utilisateur en cours de la session active, ou sur la connexion spécifiée avec l'option *link_identifier*. Si une base est spécifiée, elle deviendra la base par défaut de l'utilisateur. Si une erreur de connexion survient, la connexion en cours restera active.

> **Note : mysql_change_user()** a été introduite en PHP 3.0.13 et requiert MySQL 3.23.3 ou plus récent.

mysql_close (PHP 3, PHP 4 >= 4.0b1)

Ferme la connexion MySQL.

```
bool mysql_close (resource [link_identifier])
```

mysql_close() retourne TRUE en cas de succès et FALSE sinon.

mysql_close() ferme la connexion au serveur MySQL associée à l'identifiant *link_identifier*. Si cet identifiant n'est pas spécifié, cette commande s'applique à la dernière connexion ouverte.

> **Note :** Notez que cette commande n'est pas nécessaire, car toutes les connexions non persistantes seront automatiquement fermées à la fin du script.

mysql_close() ne ferme pas les connexions persistantes générées par **mysql_pconnect()**.

Exemple 1. Exemple MySQL_close

```
<?php
    $link = mysql_connect ("kraemer", "marliesle", "secret") {
        or die ("Impossible de se connecter");
    }
    print ("Connexion réussie");
    mysql_close ($link);
?>
```

Voir aussi **mysql_connect()** et **mysql_pconnect()**.

mysql_connect (PHP 3, PHP 4 >= 4.0b1)

Ouvre une connexion à un serveur MySQL.

```
int mysql_connect (string [hostname [:port] [:/path/to/socket]], string [username], string [password])
```

mysql_connect() retourne un identifiant positif de connexion en cas de succès, et sinon FALSE.

mysql_connect() établit une connexion à un serveur MySQL. Tous les arguments sont optionnels, et s'ils manquent, les valeurs par défaut sont utilisées ('localhost', nom du propriétaire du process, mot de passe vide).

Le nom d'hôte peut aussi inclure un numéro de port, sous la forme : "hostname:port" ou un chemin jusqu'à une socket sous la forme ":/path/to/socket" pour l'hôte localhost.

> **Note :** Le support des ":port" a été ajouté à partir de la version 3.0B4.
>
> Le support de ":/path/to/socket" a été ajouté à partir de la version 3.0.10.
>
> Vous pouvez supprimer le message d'erreur de connexion en ajoutant un arobase '@' au nom de la fonction.

Si un second appel à **mysql_connect()** est fait avec les mêmes arguments, PHP ne va pas ouvrir une nouvelle connexion, mais va retourner l'identifiant de la connexion déjà ouverte.

Le lien sera fermé automatiquement dès que l'exécution du script sera terminée, à moins d'être fermé explicitement avec **mysql_close()**.

Exemple 1. Exemple MySQL connect

```
<?php
    $link = mysql_connect ("kraemer", "marliesle", "secret") {
        or die ("Connexion impossible");
    }
    print ("Connexion réussie");
    mysql_close ($link);
?>
```

Voir aussi **mysql_pconnect()** et **mysql_close()**.

mysql_create_db (PHP 3, PHP 4 >= 4.0b1)

Crée une base de données MySQL.

```
int mysql_create_db (string database name, resource [link_identifier])
```

mysql_create_db() tente de créer une nouvelle base de données sur le serveur associé à l'identifiant *link_identifier*, ou la dernière connexion ouverte.

Exemple 1. Exemple de création de base MySQL

```
<?php
    $link = mysql_pconnect ("kron", "jutta", "geheim") {
        or die ("Connexion impossible");
    }
    if (mysql_create_db ("my_db")) {
        print ("Base de données créée\n");
    } else {
        printf ("Erreur lors de la création de la base: %s\n", mysql_error());
    }
?>
```

Pour des raisons de compatibilité ascendante, mysql_createdb() est toujours utilisable.

Voir aussi **mysql_drop_db()**.

mysql_data_seek (PHP 3, PHP 4 >= 4.0b1)

Déplace le pointeur interne de résultat.

```
int mysql_data_seek (resource result_identifier, int row_number)
```

mysql_data_seek() retourne TRUE en cas de succès et FALSE sinon.

mysql_data_seek() déplace le pointeur interne de résultat, dans le résultat associé à l'identifiant de résultat *result_identifier*. Il le fait pointer à la ligne *row_number*. Le prochain appel à **mysql_fetch_row()** retournera cette ligne.

row_number commence à 0.

Exemple 1. Exemple mysql_data_seek()

```
<?php
    $link = mysql_pconnect ("kron", "jutta", "geheim") {
        or die ("Connexion impossible");
    }
    mysql_select_db ("samp_db") {
        or die ("Selection de base impossible");
    }
    $query = "SELECT last_name, first_name FROM friends";
    $result = mysql_query ($query) {
        or die ("Requête impossible");
    }
    # récupère les lignes dans l'ordre inverse
    for ($i = mysql_num_rows ($result) - 1; $i >=0; $i--) {
        if (!mysql_data_seek ($result, $i)) {
            printf ("Impossible d'atteindre la ligne %d\n", $i);
            continue;
        }
        if(!($row = mysql_fetch_object ($result)))
            continue;
        printf ("%s %s<BR>\n", $row->last_name, $row->first_name);
    }
    mysql_free_result ($result);
?>
```

mysql_db_name (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Lit les noms des bases de donné

```
int mysql_db_name (resource result_identifier, int row, mixed [field])
```

mysql_db_name() prend comme premier argument le pointeur de résultat *result_identifier*, issu de **mysql_list_dbs()**. *row* est l'index dans le résultat.

Si une erreur survient, FALSE est retourné. Utilisez **mysql_errno()** et **mysql_error()** pour connaître la nature de l'erreur.

Exemple 1. Exemple mysql_db_name()

```
<?php
error_reporting(E_ALL);
mysql_connect('dbhost', 'username', 'password');
$db_list = mysql_list_dbs();
$i = 0;
$cnt = mysql_num_rows($db_list);
while ($i < $cnt) {
    echo mysql_db_name($db_list, $i) . "\n";
    $i++;
}
?>
```

Pour des raisons de compatibilité ascendante, mysql_dbname() est aussi accepté, mais obsolète.

mysql_db_query (PHP 3, PHP 4 >= 4.0b1)

Envoie une requête MySQL à un serveur MySQL.

```
resource mysql_db_query (string database, string query, resource [link_identifier])
```

mysql_db_query() retourne un identifiant de résultat si la requête réussit et FALSE sinon.

mysql_db_query() sélectionne une base de données et exécute une requête. Si l'identifiant de lien *link_identifier* n'est pas précisé, **mysql_db_query()** prendra par défaut la dernière connexion ouverte sur le serveur et si elle n'en trouve pas, elle tentera de se connecter, en utilisant la fonction **mysql_connect()**, sans arguments.

Voir aussi **mysql_connect()**.

Pour des raisons de compatibilité ascendante, mysql() peut aussi être utilisé.

mysql_drop_db (PHP 3, PHP 4 >= 4.0b1)

Efface une base de données MySQL.

```
bool mysql_drop_db (string database_name, resource [link_identifier])
```

mysql_drop_db() retourne TRUE en cas de succès et FALSE sinon.

mysql_drop_db() essaie d'effacer une base de données entière sur le serveur associé à l'identifiant de connexion *link_identifier*.

Voir aussi **mysql_create_db()**.

Pour des raisons de comptabilité ascendante, mysql_drop_db() est toujours utilisable.

mysql_errno (PHP 3, PHP 4 >= 4.0b1)

Retourne le numéro de message d'erreur de la dernière opération MySQL.

```
int mysql_errno (resource [link_identifier])
```

mysql_errno() retourne le numéro de message d'erreur de la dernière opération MySQL sur la connexion courante, ou sur la connexion spécifiée avec l'option *link_identifier*. Les erreurs qui sont remontées depuis le serveur MySQL ne sont plus des alertes. A la place, il faut utiliser **mysql_errno()** pour obtenir le numéro d'erreur.

```
<?php
mysql_connect("marliesle");
echo mysql_errno().": ".mysql_error()."<BR>";
mysql_select_db("nonexistentdb");
echo mysql_errno().": ".mysql_error()."<BR>";
$conn = mysql_query("SELECT * FROM nonexistenttable");
echo mysql_errno().": ".mysql_error()."<BR>";
?>
```

Voir aussi **mysql_error()**.

mysql_error (PHP 3, PHP 4 >= 4.0b1)

Retourne le texte associée avec l'erreur générée lors de la dernière requête.

```
string mysql_error (resource [link_identifier])
```

mysql_error() retourne le dernier message d'erreur MySQL sur la connexion courante, ou sur la connexion spécifiée avec *link_identifier*.

Les erreurs générées par mySQL ne se transforment plus en alerte. A la place, elles sont accessibles via ces fonctions :

```
<?php
mysql_connect("marliesle");
echo mysql_errno().": ".mysql_error()."<BR>";
mysql_select_db("nonexistentdb");
echo mysql_errno().": ".mysql_error()."<BR>";
$conn = mysql_query("SELECT * FROM nonexistenttable");
echo mysql_errno().": ".mysql_error()."<BR>";
?>
```

Voir aussi **mysql_errno()**.

mysql_fetch_array (PHP 3, PHP 4 >= 4.0b1)

Retourne une ligne de résultat sous la forme d'un tableau associatif.

```
array mysql_fetch_array (resource result_identifier, int [result_type])
```

mysql_fetch_array() retourne un tableau qui contient la ligne demandée, ou FALSE si il ne reste plus de ligne.

mysql_fetch_array() est une version étendue de **mysql_fetch_row()**. En plus d'enregistrer les données sous forme d'un tableau à indice numérique, elle peut aussi les enregistrer dans un tableau associatif, en utilisant les noms des champs comme indices.

Si plusieurs colonnes ont le même nom, la dernière colonne aura la priorité. Pour accéder aux autres colonnes du même nom, vous devez utiliser l'index numériques, ou faire un alias pour chaque colonne.

```
select t1.f1 as foo t2.f1 as bar from t1, t2
```

Il est important de souligne que **mysql_fetch_array()** N'est PAS plus lente que **mysql_fetch_row()**, tandis qu'elle ajoute un confort d'utilisation notable.

L'option *result_type* de **mysql_fetch_array()** est une constant qui peut prendre les valeurs suivantes : MYSQL_ASSOC, MYSQL_NUM et MYSQL_BOTH.

Pour plus de détails, voir aussi **mysql_fetch_row()**.

Exemple 1. mysql fetch array

```
<?php
mysql_connect($host,$user,$password);
$result = mysql_db_query("database","select * from table");
while($row = mysql_fetch_array($result)) {
    echo $row["user_id"];
    echo $row["fullname"];
}
mysql_free_result($result);
?>
```

mysql_fetch_assoc (PHP 4 >= 4.0.3)

Lit une ligne de résultat dans un tableau associatif

```
array mysql_fetch_assoc (resource result_identifier)
```

mysql_fetch_assoc() retourne un tableau associatif qui contient la ligne lue, ou bien FALSE, si il ne reste plus de lignes.

mysql_fetch_assoc() est équivalente à **mysql_fetch_array()** utilisée avec l'option MYSQL_ASSOC. Elle ne retourne qu'un tableau associatif. C'est le fonctionnement original de **mysql_fetch_array()**. Si vous avez besoin d'indices numériques, utilisez **mysql_fetch_array()**.

Si plusieurs colonnes on le même nom, la dernière aura la priorité. Pour accéder aux autres colonnes du même nom, vous devez utiliser **mysql_fetch_array()** et les indices numériques.

Une chose importante à noter est que **mysql_fetch_assoc()** n'est PAS significativement plus lente que **mysql_fetch_row()**, alors qu'elle apporte un confort d'utilisation important.

Pour plus de détails, reportez vous à **mysql_fetch_row()** et **mysql_fetch_array()**.

Exemple 1. mysql_fetch_assoc()

```
<?php
mysql_connect ($host, $user, $password);
$result = mysql_db_query ("database","select * from table");
while ($row = mysql_fetch_assoc ($result)) {
    echo $row["user_id"];
    echo $row["fullname"];
}
mysql_free_result ($result);
?>
```

mysql_fetch_field (PHP 3, PHP 4 >= 4.0b1)

Retourne les données enregistrées dans une colonne sous forme d'objet.

```
object mysql_fetch_field (resource result_identifier, int [field_offset])
```

Retourne un objet contenant les données.

mysql_fetch_field() sert à obtenir des informations à propos des champs, dans certaines requêtes. Si l'offset du champs n'est pas spécifié, le champs suivant le dernier champs retourné, est retourné.

Les propriétés de l'objet sont :

- name - nom de la colonne
- table - nom de la table de la colonne
- max_length - taille maximale de la colonne
- not_null - 1 si la colonne ne peut pas être `NULL` (attribut NOT `NULL`)
- primary_key - 1 si la colonne est une clé primaire (attribut PRIMARY KEY)
- unique_key - 1 si la colonne est une clé unique (attribut UNIQUE)
- multiple_key - 1 si la colonne est une clé non-unique
- numeric - 1 si la colonne est numérique
- blob - 1 si la colonne est BLOB
- type - le type de la colonne
- unsigned - 1 si la colonne est non signé
- zerofill - 1 si la colonne est complétée par des zéros.

Voir aussi **mysql_field_seek()**

mysql_fetch_lengths (PHP 3, PHP 4 >= 4.0b1)

Retourne la taille de chaque colonne d'une ligne de résultat.

```
array mysql_fetch_lengths (resource result_identifier)
```

mysql_fetch_lengths() retourne un tableau avec la taille de chaque colonne de la dernière ligne retournée par **mysql_fetch_row()**, sinon `FALSE`.

mysql_fetch_lengths() stocke les tailles de chaque colonne de la dernière ligne retournée par **mysql_fetch_row()**, **mysql_fetch_array()** et **mysql_fetch_object()** dans un tableau, en commencant à la position.

Voir aussi **mysql_fetch_row()**.

mysql_fetch_object (PHP 3, PHP 4 >= 4.0b1)

Retourne les lignes résultats sous la forme d'un objet.

```
object mysql_fetch_object (resource result_identifier, int [result_type])
```

mysql_fetch_object() retourne un objet dont les propriétés correspondent à une ligne d'un résultat, ou `FALSE` si il n'y a plus d'autres lignes.

mysql_fetch_object() est identique à **mysql_fetch_array()**, à la différence qu'elle retourne un objet à la place d'un tableau. Vous pourrez ainsi accéder aux valeurs des champs par leur nom, mais plus par leur offset (les nombres ne sont pas des noms MySQL).

L'argument optionnel *result_type* est une constante qui peut prendre les valeurs suivantes : MYSQL_ASSOC, MYSQL_NUM et MYSQL_BOTH.

Concernant la vitesse, **mysql_fetch_object()** est aussi rapide que **mysql_fetch_array()** et presque aussi rapide que **mysql_fetch_row()** (la différence est insignifiante)

Exemple 1. mysql fetch object

```
<?php
mysql_connect($host,$user,$password);
$result = mysql_db_query("database","select * from table");
while($row = mysql_fetch_object($result)) {
    echo $row->user_id;
    echo $row->fullname;
}
mysql_free_result($result);
?>
```

Voir aussi **mysql_fetch_array()** et **mysql_fetch_row()**.

mysql_fetch_row (PHP 3, PHP 4 >= 4.0b1)

Retourne une ligne de résultat sous la forme d'un tableau.

```
array mysql_fetch_row (resource result_identifier)
```

Retourne un tableau énuméré qui correspond à la ligne demandée, ou FALSE si il ne reste plus de ligne.

mysql_fetch_row() va rechercher une ligne dans le résultat associé à l'identifiant de résultat spécifié. La ligne est retournée sous la forme d'un tableau. Chaque colonne est enregistré sous la forme d'un tableau commencant à la position 0.

Les appels suivants à **mysql_fetch_row()** retourneront la ligne suivante dans le résultat, ou FALSE si il n'y a plus de ligne disponible.

Voir aussi **mysql_fetch_array()**, **mysql_fetch_object()**, **mysql_data_seek()**, **mysql_fetch_lengths()** et **mysql_result()**.

mysql_field_flags (PHP 3, PHP 4 >= 4.0b1)

Retourne le sémaphore associé à la colonne spécifiée dans le résultat courant.

```
string mysql_field_flags (resource result_identifier, int field_offset)
```

mysql_field_flags() retourne le sémaphore associé au champs spécifié par *field_offset*. Les sémaphores sont retournés comme des mots, séparés par des espaces, ce qui les rend facile à séparer, avec la commande **explode()**.

Les valeurs suivantes (pour une version suffisamment récente de MySQL) sont disponibles : "not_null", "primary_key", "unique_key", "multiple_key", "blob", "unsigned", "zerofill", "binary", "enum", "auto_increment", "timestamp".

Pour des raisons de compatibilité ascendante, mysql_fieldflags() peut encore être utilisé.

mysql_field_name (PHP 3, PHP 4 >= 4.0b1)

Retourne le nom d'une colonne

```
string mysql_field_name (resource result_identifier, int field_index)
```

mysql_field_name() retourne le nom d'une colonne. Les arguments de la fonction sont un identifiant de résultat *result_identifier* et l'index du champs, ie.`mysql_field_name($result,2);`.

mysql_field_name() retournera le nom du deuxième champs dans le résultat associé à $result.

Pour des raisons de compatibilité ascendante, mysql_fieldname() peut encore être utilisé.

mysql_field_len (PHP 3, PHP 4 >= 4.0b1)

Retourne la longueur du champs spécifié.

```
int mysql_field_len (resource result_identifier, int field_offset)
```

mysql_field_len() retourne la taille du champs spécifié par son offset *field_offset*.

Pour des raisons de compatibilité ascendante, mysql_fieldlen() peut encore être utilisé.

mysql_field_seek (PHP 3, PHP 4 >= 4.0b1)

Place le pointeur de résultat à un offset donné

```
int mysql_field_seek (resource result_identifier, int field_offset)
```

Place le pointeur de résultat sur le champs spécifié. Lors du prochain appel à **mysql_fetch_field()** qui n'aura pas d'argument d'index de champs, le champs désormais pointé sera retourné.

Voir aussi **mysql_fetch_field()**.

mysql_field_table (PHP 3, PHP 4 >= 4.0b1)

Retourne le nom de la table oú se trouve une colonne

```
string mysql_field_table (resource result_identifier, int field_offset)
```

mysql_field_table() retourne le nom de la table oú se trouve une colonne. Pour des raisons de compatibilité ascendante, mysql_fieldtable() peut encore être utilisé.

mysql_field_type (PHP 3, PHP 4 >= 4.0b1)

Retourne le type de la colonne spécifiée dans le résultat courant.

```
string mysql_field_type (resource result_identifier, int field_offset)
```

mysql_field_type() est similaire à la fonction **mysql_field_name()**. Les arguments sont identiques, mais c'est le type du champs qui est retourné. Il vaudra "int", "real", "string", "blob", ou d'autres, comme détaillé dans la documentation MySQL.

Exemple 1. Types mysql field

```
<?php
mysql_connect("localhost:3306");
mysql_select_db("wisconsin");
$result = mysql_query("SELECT * FROM onek");
$fields = mysql_num_fields($result);
$rows   = mysql_num_rows($result);
$i = 0;
$table = mysql_field_table($result, $i);
echo "Your '".$table."' table has ".$fields." fields et ".$rows." records <BR>";
echo "The table has the following fields <BR>";
while ($i < $fields) {
    $type  = mysql_field_type($result, $i);
    $name  = mysql_field_name($result, $i);
    $len   = mysql_field_len($result, $i);
    $flags = mysql_field_flags($result, $i);
    echo $type." ".$name." ".$len." ".$flags."<BR>";
    $i++;
}
mysql_close();
?>
```

Pour des raisons de compatibilité ascendante, mysql_fieldtype() peut encore être utilisé.

mysql_free_result (PHP 3, PHP 4 >= 4.0b1)

Efface le résultat de la mémoire.

```
int mysql_free_result (resource result_identifier)
```

mysql_free_result() n'est à appeler que si vous avez peur d'utiliser trop de mémoire durant l'exécution de votre script. Toute la mémoire associée à l'identifiant de résultat sera automatiquement libérée.

Pour des raisons de compatibilité ascendante, mysql_freeresult() peut encore être utilisé.

mysql_insert_id (PHP 3, PHP 4 >= 4.0b1)

Retourne l'identifiant généré par la dernière requête INSERT.

```
int mysql_insert_id (resource [link_identifier])
```

mysql_insert_id() retourne le dernier identifiant généré par un champs de type AUTO_INCREMENT, sur la connexion MySQL courante, ou bien sur la connexion spécifiée par *link_identifier*. **mysql_insert_id()** ne prend aucun argument. Elle retourne le dernier identifiant généré par la dernière fonction INSERT effectuée.

mysql_list_dbs (PHP 3, PHP 4 >= 4.0b1)

Liste les bases de données disponibles sur le serveur MySQL.

```
resource mysql_list_dbs (resource [link_identifier])
```

mysql_list_dbs() retournera un identifiant de résultat, qui contiendra les noms des bases de données disponibles sur la connexion MySQL courante, ou bien sur la connexion spécifiée par *`link_identifier`*. Utilisez la fonction **mysql_tablename()** pour lire toutes les bases de données.

Pour des raisons de compatibilité ascendante, mysql_listdbs() est encore disponible.

mysql_list_fields (PHP 3, PHP 4 >= 4.0b1)

Liste les champs du résultat MySQL.

```
resource mysql_list_fields (string database_name, string table_name, resource
[link_identifier])
```

mysql_list_fields() recherche les informations à propos de la table spécifiée sur la connexion MySQL courante, ou bien sur la connexion spécifiée par *`link_identifier`*. Les arguments sont la base de données et le nom de la table. Un pointeur de résultat est retourné et pourra être passé à **mysql_field_flags()**, **mysql_field_len()**, **mysql_field_name()** et **mysql_field_type()**.

Un identifiant de résultat est un entier positif. La fonction retourne -1 si une erreur survient. Une chaîne décrivant le problème rencontré sera placée dans la variable `$phperrmsg` et, à moins que la fonction n'ait été appelée sous la forme `@mysql()`, cette erreur sera aussi affichée.

Pour des raisons de compatibilité ascendante, mysql_listfields() est encore disponible.

mysql_list_tables (PHP 3, PHP 4 >= 4.0b1)

Liste les tables d'une base de données.

```
resource mysql_list_tables (string database, resource [link_identifier])
```

mysql_list_tables() prend le nom d'une base de données *`database`* et retourne un identifiant de résultat, qui contiendra la liste des tables sur la connexion MySQL courante, ou bien sur la connexion spécifiée par *`link_identifier`*. La fonction **mysql_tablename()** est le meilleur moyen d'extraire les noms des tables depuis l'identifiant de résultat.

Pour des raisons de compatibilité ascendante, mysql_listtables() est encore disponible.

mysql_num_fields (PHP 3, PHP 4 >= 4.0b1)

Retourne le nombre de champs d'un résultat.

```
int mysql_num_fields (resource result_identifier)
```

mysql_num_fields() retourne le nombre de champs d'un résultat.

Voir aussi **mysql_db_query()**, **mysql_query()**, **mysql_fetch_field()**, **mysql_num_rows()**.

Pour des raisons de compatibilité ascendante mysql_numfields() est encore disponible.

mysql_num_rows (PHP 3, PHP 4 >= 4.0b1)

Retourne le nombre de ligne d'un résultat.

```
int mysql_num_rows (resource result_identifier)
```

mysql_num_rows() retourne le nombre de lignes d'un résultat. Cette commande n'est valide que pour les commandes SELECT . Pour connaître le nombre de lignes retournées par INSERT, UPDATE ou DELETE, utiliseez **mysql_affected_rows()**.

Exemple 1. Exemple mysql_num_rows() par crubel@trilizio.org

```
<?php
$conn = mysql_connect("adresse de l'hote", "utilisateur", "mot de passe");
mysql_select_db("base",$conn); // nécessaire si vous avez plusieurs bases
$Resultfornummembers = mysql_query("SELECT * FROM Accounts",$conn);
$NumMembers = mysql_num_rows($Resultfornummembers);
echo "$NumMembers Membres";
?>
```

Voir aussi **mysql_db_query()**, **mysql_query()** et **mysql_fetch_row()**.

Pour des raisons de compatibilité ascendante mysql_numrows() est encore disponible.

mysql_pconnect (PHP 3, PHP 4 >= 4.0b1)

Ouvre une connexion persistante à un serveur MySQL.

```
resource mysql_pconnect (string [hostname [:port] [:/path/to/socket]] [, string username [, string password]])
```

Retourne un lien persistant positif en cas de succès et sinon FALSE en cas d'erreur.

mysql_pconnect() établit une connexion persistante à un serveur MySQL. Tous les arguments sont optionnels et des valeurs par défaut seront utilisérs en cas d'omission ('localhost', nom d'utilisateur propriétaire du processus, mot de passe vide).

Le nom de l'hôte peut aussi inclure le numéro de port, c'est à dire "hostname:port" ou un chemin jusqu'à la socket ":/path/to/socket" pour l'hôte local.

> **Note :** Le support de ":port" a été ajouté à partir de la version 3.0B4.
>
> Le support de ":/path/to/socket" a été ajouté à partir de la version 3.0.10.

mysql_pconnect() se comporte exactement comme **mysql_connect()**, mais avec deux différences majeures :

Premièrement, lors de la connexion, la fonction essaie de trouver une connexion permanante déjà ouverte sur cet hote, avec le même nom d'utilisateur et de mot de passe. Si une telle connexion est trouvée, son identifiant est retourné, sans ouvrir de nouvelle connexion.

Deuxièmement, la connexion au serveur MySQL ne sera pas terminée avec la fin du script. Au lieu de cela, le lien sera conservé pour un prochain accès (**mysql_close()** ne terminera pas une connexion persistante établie par **mysql_pconnect()**).

C'est pourquoi ce type de connexion est dite 'persistante'.

mysql_query (PHP 3, PHP 4 >= 4.0b1)

Envoie une requête SQL à un serveur MySQL.

```
resource mysql_query (string query [, resource link_identifier])
```

mysql_query() envoie une requête SQL à la base de données actuellement active sur le serveur MysQL. Si *link_identifier* n'est pas précisé, la dernière connexion est utilisée. Si aucune connexion n'a été ouverte, la fonction

tentera d'en ouvrir une, avec la fonction **mysql_connect()** mais sans aucun paramètre (c'est à dire avec les valeurs par défaut).

mysql_connect()

mysql_query() retourne `TRUE` ou `FALSE`, pour indiquer le succès ou l'échec de la requête. En cas de retour `TRUE`, la requête était valide et a pu être exécuté sur le serveur. Cela n'indique pas le nombre de ligne affectées, ou retournées. Il est parfaitement possible qu'une requête valide n'affecte aucune ligne ou ne retourne aucune ligne.

L'exemple suivant est syntaxiquement invalide, ce qui conduit **mysql_query()** à l'échec et retourne `FALSE`:

Exemple 1. mysql_query()

```
<?php
$result = mysql_query ("SELECT * WHERE 1=1")
    or die ("Requete invalide");
?>
```

L'exemple suivant est sémantiquement invalide si `my_col` n'est pas une colonne de la table `my_tbl`, ce qui conduit **mysql_query()** à l'échec et retourne `FALSE` :

Exemple 2. mysql_query()

```
<?php
$result = mysql_query ("SELECT my_col FROM my_tbl")
    or die ("Invalid query");
?>
```

mysql_query() échouera aussi et retournera aussi `FALSE` si les droits d'accès ne sont pas suffisants.

En supposant que la requête réussisse, vous pouvez appeler **mysql_affected_rows()** pour connaître le nombre de lignes affectées (pour les commandes DELETE, INSERT, REPLACE, ou UPDATE). Pour les commandes SELECT , **mysql_query()** retourne un identifiant de résultat que vous pouvez passer à **mysql_result()**. Lorsque vous avez terminé avec le résultat, libérez la mémoire avec **mysql_free_result()**.

Voir aussi **mysql_affected_rows()**, **mysql_db_query()**, **mysql_free_result()**, **mysql_result()**, **mysql_select_db()** et **mysql_connect()**.

mysql_result (PHP 3, PHP 4 >= 4.0b1)

Retourne un champs d'un résultat.

```
mixed mysql_result (resource result_identifier, int row, mixed [field])
```

mysql_result() retourne le contenu d'un champs dans le résultat MySQL *`result_identifier`*. L'argument *`row`* peut-être un offset de champs, ou le nom d'un champs, ou le nom de la table + point + le nom du champs ("table.champs"). Si la colonne a été aliasée, utilisez de préférence l'alias.

Lorsque vous travaillez sur des résultats de grande taille, vous devriez utiliser une des fonctions qui vont rechercher une ligne entière dans un tableau. Ces fonctions sont NETTEMENT plus rapides. De plus, l'utilisation d'un offset numériques est aussi beaucoup plus rapide que de spécifier un nom litéral.

Les appels **mysql_result()** ne devraient pas être mélangés avec d'autres fonctions qui travaillent aussi sur le résultat.

Alternatives à haut rendement, RECOMMANDEES : **mysql_fetch_row()**, **mysql_fetch_array()** et **mysql_fetch_object()**.

mysql_select_db (PHP 3, PHP 4 >= 4.0b1)

Sélectionne une base de données MySQL.

```
int mysql_select_db (string database_name, resource [link_identifier])
```

mysql_select_db() retourne `TRUE` en cas de succès, `FALSE` sinon.

mysql_select_db() change la base de données active sur la connexion représentée par *`link_identifier`*. Si aucun identifiant n'est spécifié, la dernière connexion est utilisée. S'il n'y a pas de dernière connexion, la fonction tentera de se connecter seule, avec **mysql_connect()** et les paramètres par défaut.

Toutes les requêtes suivantes avec **mysql_query()** seront faites avec la base de données active.

Voir aussi **mysql_connect()**, **mysql_pconnect()** et **mysql_query()**.

Pour des raisons de compatibilité ascendante mysql_selectdb() est encore disponible.

mysql_tablename (PHP 3, PHP 4 >= 4.0b1)

Lit le nom de la table qui contient le champs spécifié.

```
string mysql_tablename (resource result_identifier, int i)
```

mysql_tablename() prend le pointeur de résultat obtenu avec **mysql_list_tables()** ou bien un index entier et retourne le nom de la table. La fonction **mysql_num_rows()** peut être utilisée pour déterminer le nombre de tables dans le pointeur de résultat.

Exemple 1. Exemple mysql_tablename()

```
<?php
mysql_connect ("localhost:3306");
$result = mysql_list_tables ("wisconsin");
$i = 0;
while ($i < mysql_num_rows($result)) {
    $tb_names[$i] = mysql_tablename ($result, $i);
    echo $tb_names[$i] . "<BR>";
    $i++;
}
?>
```

LVI. Réseau

checkdnsrr (PHP 3, PHP 4 >= 4.0b1)

Résolution DNS d'une adresse IP.

```
int checkdnsrr (string host [, string type])
```

checkdnsrr() recherche l'enregistrement DNS de type *type* correspondant à l'hôte *host*. Retourne TRUE si un record a été trouvé, et FALSE en cas d'erreur ou d'échec.

type peut prendre les valeurs suivantes : A, MX, NS, SOA, PTR, CNAME, ou ANY. Par défaut, la valeur est : MX.

host peut être soit une adresse IP de la forme x.x.x.x avec x entre 0 et 256, soit un nom d'hôte.

Voir aussi **getmxrr()**, **gethostbyaddr()**, **gethostbyname()**, **gethostbynamel()** et la page de manuel named(8).

closelog (PHP 3, PHP 4 >= 4.0b1)

Ferme la connexion à l'historique système.

```
int closelog (void)
```

closelog() ferme le pointeur qui sert à écrire dans l'historique système. L'utilisation de **closelog()** est optionnelle.

Voir aussi **define_syslog_variables()**, **syslog()** et **openlog()**.

debugger_off (PHP 3)

Inactive le debugger interne de PHP.

```
int debugger_off (void)
```

debugger_off() inactive le debuggeur interne de PHP. Le debuggeur est toujours en cours de développement.

debugger_on (PHP 3)

Active le debugger interne de PHP.

```
int debugger_on (string address)
```

debugger_on() active le debugger interne de PHP, et le connecte à l'adresse *address*. Le debuggeur est toujours en cours de développement.

define_syslog_variables (PHP 3, PHP 4 >= 4.0b1)

Initialise toutes les constantes liées au syslog

```
void define_syslog_variables (void)
```

define_syslog_variables() initialise toutes les constantes utilisées par les fonctions de syslog.

Voir aussi **openlog()**, **syslog()** et **closelog()**.

fsockopen (PHP 3, PHP 4 >= 4.0b1)

Ouvre une socket de connexion Internet ou Unix.

```
int fsockopen (string [udp://]hostname, int port [, int errno [, string errstr [, double
timeout]]])
```

fsockopen() créer un flot de connexion à l'Internet (AF_INET) ou à un domaine Unix (AF_UNIX). Via Internet, cette fonction va ouvrir une socket de connexion TCP avec l'hôte *hostname* sur le port *port*. Pour les connexions UDP, vous devez explicitement spécifier le protocole : *udp://hostname*. Via un domaine Unix, *hostname* représente le chemin jusqu'à la socket, et *port* doit être mis à 0. L'option *timeout* sert à donner une durée maximale à cet appel.

fsockopen() retourne un pointeur de fichier qui peut être utilisé avec d'autres fonctions fichiers, telles que **fgets()**, **fgetss()**, **fputs()**, **fclose()** et **feof()**.

Si l'appel échoue, **fsockopen()** retourne FALSE, et si les options *errno* et *errstr* ont été fournies, elles contiennent désormais les raisons de l'échec. Si l'erreur retournée est 0 et que la fonction retourne FALSE, c'est une indication d'erreur. C'est probablement du à une erreur d'initialisation de la socket. Notez que *errno* et *errstr* sont passées par référence.

Suivant les environnements, le type 'domaine Unix' ou l'option *timeout* ne sont pas toujours disponibles.

La socket sera ouverte par défaut en mode bloquant. Vous pouvez changer de mode en utilisant : **socket_set_blocking()**.

Exemple 1. Exemple avec fsockopen()

```
<?php
$fp = fsockopen("www.php.net", 80, &$errno, &$errstr, 30);
if(!$fp) {
echo "$errstr ($errno)<br>\n";
} else {
fputs($fp,"GET / HTTP/1.0\n\n");
while(!feof($fp)) {
echo fgets($fp,128);
}
fclose($fp);
}
?>
```

L'exemple ci-dessous décrit comment lire la date et l'heure grâce à un service UDP "daytime" (port 13), sur votre propre machine.

Exemple 2. Utilisation d'une connexion UDP

```
<?php
$fp = fsockopen("udp://127.0.0.1", 13, &$errno, &$errstr);
if (!$fp) {
    echo "ERREUR: $errno - $errstr<br>\n";
} else {
    fwrite($fp,"\n");
    echo fread($fp, 26);
    fclose($fp);
}
?>
```

Note : Le paramètre *timeout* a été introduit en PHP 3.0.9 et le support UDP en PHP 4.

Voir aussi: **pfsockopen()**, **socket_set_blocking()**, **socket_set_timeout()**, **fgets()**, **fgetss()**, **fputs()**, **fclose()**, et **feof()**.

gethostbyaddr (PHP 3, PHP 4 >= 4.0b1)

Retourne le nom d'hôte correspondant à une IP.

```
string gethostbyaddr (string ip_address)
```

gethostbyaddr() retourne le nom d'hôte correspondant à l'IP *ip_address*. Si une erreur survient, retourne *ip_address*.

Voir aussi **gethostbyname()**.

gethostbyname (PHP 3, PHP 4 >= 4.0b1)

Retourne l'adresse IP correspondant à un hôte.

```
string gethostbyname (string hostname)
```

gethostbyname() retourne l'adresse IP correspondant à l'hôte *hostname*.

Voir aussi **gethostbyaddr()**.

gethostbynamel (PHP 3, PHP 4 >= 4.0b1)

Retourne la liste d'IP correspondants à un hôte.

```
array gethostbynamel (string hostname)
```

gethostbynamel() retourne la liste d'IP correspondant à l'hôte *hostname*.

Voir aussi **gethostbyname()**, **gethostbyaddr()**, **checkdnsrr()**, **getmxrr()** et la page 8 du manuel.

getmxrr (PHP 3, PHP 4 >= 4.0b1)

Retourne les enregistrements MX d'un hôte.

```
int getmxrr (string hostname, array mxhosts [, array weight])
```

getmxrr() effectue une recherche DNS pour obtenir les enregistrements MX de l'hôte *hostname*. Retourne TRUE si des enregistrements sont trouvés, et FALSE si une erreur est rencontrée, ou si la recherche échoue.

La liste des enregistrements MX est placée dans le tableau *mxhosts*. Si le tableau *weight* est fourni, il sera rempli par les informations de poids.

Voir aussi **checkdnsrr()**, **gethostbyname()**, **gethostbynamel()**, **gethostbyaddr()**, et la page 8 du manuel.

getprotobyname (PHP 4 >= 4.0b4)

Retourne le numéro de protocole associé au nom de protocole

```
int getprotobyname (string name)
```

getprotobyname() retourne le numéro de protocole associé avec le nom de protocole *name*, comme dans `/etc/protocols`.

Voir aussi **getprotobynumber()**.

getprotobynumber (PHP 4 >= 4.0b4)

Retourne le nom de protocole associé au numéro de protocole

```
string getprotobynumber (int number)
```

getprotobynumber() retourne le numéro de protocole associé avec le nom de protocole *name*, comme dans `/etc/protocols`.

Voir aussi **getprotobyname()**.

getservbyname (PHP 4 >= 4.0b4)

Retourne le numéro de port associé à un service Internet, et un protocole.

```
int getservbyname (string service, string protocol)
```

getservbyname() retourne le numéro de port associé à au service *service* et protocole *protocol*, comme dans `/etc/services`. *protocol* vaut soit `tcp` ou `udp`.

Voir aussi **getservbyport()**.

getservbyport (PHP 4 >= 4.0b4)

Retourne le service Internet qui correspond au port et protocole.

```
string getservbyport (int port, string protocol)
```

getservbyport() le service internet associé au port *port* pour le protocole *protocol* comme dans `/etc/services`. *protocol* vaut soit `tcp` ou `udp`.

Voir aussi **getservbyname()**.

ip2long (PHP 4 >= 4.0RC1)

Convertit une chaîne contenant une adresse (IPv4) IP numérique en adresse litérale.

```
int ip2long (string ip_address)
```

ip2long() génére une adresse IPv4 à partir de son équivalent numérique.

Exemple 1. Exemple ip2long()

```
<?php
$ip = gethostbyname("www.php.net");
$out = "Les URLS suivantes sont équivalentes :<br>\n";
$out .= "http://www.php.net/, http://".$ip."/, et http://".ip2long($ip)."/<br>\n";
echo $out;
?>
```

Voir aussi: **long2ip()**

long2ip (PHP 4 >= 4.0RC1)

Convertit une adresse IP (IPv4) en adresse IP numérique

```
string long2ip (int proper_address)
```

long2ip() génére une adresse IP (format aaa.bbb.ccc.ddd) à partir de sa représentation litérale.

Voir aussi: **ip2long()**

openlog (PHP 3, PHP 4 >= 4.0b1)

Ouvre la connexion à l'historique système.

```
int openlog (string ident, int option, int facility)
```

openlog() ouvre la connexion à l'historique système. La chaîne *ident* sera ajouté à chaque message. Les valeurs de *option* et *facility* sont données ci-dessous. L'utilisation de **openlog()** est optionnelle; cette fonction sera automatiquement appelée par **syslog()** si nécessaire, et dans ce cas, l'identification sera mise par défaut à FALSE. *facility* sert à indiquer quel programme enregistre ce message. Cela vous permet de spécifier (sur la machine d'historique) comment traiter les messages venant de plusieurs serveurs.

Tableau 1. Options openlog()

Constante	Description
LOG_CONS	Si une erreur survient lors de l'envoi des données au gestionnaire d'historique, écrire directement l'erreur sur la console.
LOG_NDELAY	Ouvre immédiatement une connexion au gestionnaire d'historique
LOG_ODELAY	(par défaut) retarde l'ouverture de la connexion jusqu'à ce que le premier message soit enregistré
LOG_PERROR	Envoie le message au gestionnaire standard
LOG_PID	Inclus le PID à chaque message

Vous pouvez utiliser une ou plusieurs de ces options. Pour les combiner, utiliser l'opérateur OR. Par exemple, pour ouvrir immédiatement la connexion, écrire sur la console et inclure le PID de chaque message, utilisez : `LOG_CONS | LOG_NDELAY | LOG_PID`.

Tableau 2. Paramètre *facility* de openlog()

Constante	Description
LOG_AUTH	securité/messages d'autorisation (utilisez LOG_AUTHPRIV, pour remplacer cette constante sur les systèmes où elle est définie).
LOG_AUTHPRIV	securité/messages d'autorisation (privé)
LOG_CRON	démon horologe (cron et at)
LOG_DAEMON	autres démons système
LOG_KERN	noyau (kernel)
LOG_LOCAL0 ... LOG_LOCAL7	reservé pour utilisation ultérieure

Constante	Description
LOG_LPR	imprimante (line printer subsystem)
LOG_MAIL	messagerie mail
LOG_NEWS	USENET newsgroup
LOG_SYSLOG	messages generé en interne par syslogd
LOG_USER	messages utilisateurs générique
LOG_UUCP	UUCP subsystem

Voir aussi **define_syslog_variables()**, **syslog()** et **closelog()**.

pfsockopen (PHP 3>= 3.0.7, PHP 4 >= 4.0b1)

Ouvre une socket de connexion Internet ou Unix persistante.

```
int pfsockopen (string hostname, int port [, int errno [, string errstr [, int timeout]]])
```

pfsockopen() se comporte exactement comme **fsockopen()** mais la connexion ouverte le reste, même après la fin du script. C'est la version persistante de **fsockopen()**.

socket_get_status (PHP 4 >= 4.0b4)

Retourne les informations sur une socket

```
array socket_get_status (resource socket_get_status)
```

socket_get_status() retourne les informations sur la socket *socket_get_status*, et fournit la réponse sous la forme d'un tableau à quatre entrées:

- *timed_out* (bool) - La socket a éxpirée en attendant des données
- *blocked* (bool) - La socket a été bloquée
- *eof* (bool) - Indique un événement fin de fichier (EOF)
- *unread_bytes* (int) - Nombre d'octets restant dans les buffers de la socket.

Voir aussi **accept_connect()**, **bind()**, **connect()**, **listen()** et **strerror()**.

socket_set_blocking (PHP 4 >= 4.0b4)

Active/désactive le mode bloquant d'une socket.

```
int socket_set_blocking (int socket descriptor, int mode)
```

Si *mode* est FALSE, la socket est mise en mode non bloquant, et si il est TRUE, la socket est mise en mode bloquant. Cela affecte des appels tels que **fgets()** qui lisent depuis une socket. En mode non bloquant, un appel **fgets()** retournera immediatement toujours TRUE tandis qu'en mode bloquant, elle va attendre que des données arrivent pour répondre TRUE.

socket_set_timeout (PHP 4 >= 4.0b4)

Fixe la durée de vie de la socket

```
bool socket_set_timeout (int socket descriptor, int seconds, int microseconds)
```

socket_set_timeout() fixe la durée de vie de la socket *socket descriptor*, exprimée comme la somme de *seconds* secondes et *microseconds* micro-secondes.

Exemple 1. Exemple socket_set_timeout()

```
<?php
$fp = fsockopen("http://www.php.net", 80);
if(!$fp) {
    echo "Unable to open\n";
} else {
    fputs($fp,"GET / HTTP/1.0\n\n");
    $start = time();
    socket_set_timeout($fp, 2);
    $res = fread($fp, 2000);
    var_dump(socket_get_status($fp));
    fclose($fp);
    print $res;
}
?>
```

Cette fonction s'appelait set_socket_timeout() mais elle est désormais obsolète.

Voir aussi: **fsockopen()** et **fopen()**.

syslog (PHP 3, PHP 4 >= 4.0b1)

Génére un message dans l'historique système.

```
int syslog (int priority, string message)
```

syslog() génére un message qui sera inscrit dans l'historique par le système. *priority* est une combinaison des valeurs d'accès et de niveau, qui seront décrites dans la prochaîne section. Les derniers arguments sont le message à envoyer. Attention : les caractères `%m` seront remplacés par l'erreur (sous forme de chaîne), présente dans errno.

Tableau 1. Priorités syslog()(en ordre décroissant)

Constante	Description
LOG_EMERG	système inutilisable
LOG_ALERT	une décision doit être prise immédiatement
LOG_CRIT	conditions critiques
LOG_ERR	conditions d'erreur
LOG_WARNING	conditions d'alerte
LOG_NOTICE	condition normale, mais significative
LOG_INFO	message d'information
LOG_DEBUG	message de débuggage

Exemple 1. Utilisation de syslog()

```
<?php
define_syslog_variables();
// ouverture de syslog, ajout du PID et envoie simultané du
// message à la sortie standard et à un mecanisme
// spécifique
openlog("myScripLog", LOG_PID | LOG_PERROR, LOG_LOCAL0);
// quelques lignes de code
if (authorized_client()) {
    // faire quelquechose
} else {
    // client non autorisé!
    // notation de la tentative
    $access = date("Y/m/d H:i:s");
    syslog(LOG_WARNING,"Client non autorisé: $access $REMOTE_ADDR ($HTTP_USER_AGENT)");
}
closelog();
?>
```

Pour plus d'informations sur comment mettre en place un gestionnaire d'historique, reportez vous au manuel Unix, page 5 syslog.conf(5). D'autres informations sur les systèmes d'historique et leurs options sont aussi disponibles dans le manuel syslog(3) des machines Unix.

Avec Windows NT, l'historique est pris en charge par Event Log.

LVII. ODBC unifié

En plus du support de l'ODBC normal, l'ODBC unifié de PHP vous donne accès à diverses bases de données qui ont emprunté la sémantique des API ODBC pour implémenter leur propres API. Au lieu de maintenir de multiples pilotes qui sont similaires, ces pilotes ont été rassemblés dans un jeu de fonctions ODBC uniques.

Les bases de données suivantes sont supportées par l'ODBC unifié : Adabas D (http://www.adabas.com/), IBM DB2 (http://www.ibm.com/db2/), iODBC (http://www.iodbc.org/), Solid (http://www.solidtech.com/), et Sybase SQL Anywhere (http://www.sybase.com/).

Reportez-vous à Installation sous Unix pour plus de détails sur les configurations de ces serveurs.

> **Note :** Il n'y a pas d'ODBC utilisé lors des connexions aux bases de données ci-dessus. Les fonctions que vous utiliserez portent des noms évocateurs, et utilisent les mêmes syntaxes que leurs cousines d'ODBC. L'exception à ceci est iODBC. En compilant PHP avec le support iODBC, vous pourrez utiliser n'importe quel pilote compatible ODBC avec vos applications PHP. iODBC est mis à jour à OpenLink Software (http://www.openlinksw.com/). Plus d'informations sur iODBC, ainsi qu'un HOWTO (en anglais), est disponible à www.iodbc.org (http://www.iodbc.org/).

odbc_autocommit (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Active le mode auto-validation

```
int odbc_autocommit (resource connection_id [, int OnOff])
```

Sans le paramètre *OnOff*, **odbc_autocommit()** retourne le statut d'auto-validation de la connexion *connection_id*. TRUE si le mode est activé, FALSE s'il ne l'est pas, ou si une erreur survient.

Si *OnOff* vaut TRUE, l'auto-validation est activée. S'il est FALSE, l'auto-validation est désactivée. **odbc_autocommit()** retourne TRUE en cas de succès, FALSE en cas d'échec.

Par défaut, l'auto-validation est activée. Désactiver l'auto-validation est équivalent à démarrer une transaction.

Voir aussi **odbc_commit()** et **odbc_rollback()**.

odbc_binmode (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Modifie la gestion des colonnes de données binaires.

```
int odbc_binmode (resource result_id, int mode)
```

Types ODBC SQL affectés: BINARY, VARBINARY, LONGVARBINARY.

- ODBC_BINMODE_PASSTHRU: Mode Passthru
- ODBC_BINMODE_RETURN: Retourne tel quel.
- ODBC_BINMODE_CONVERT: Convertit en char et retourne la valeur.

Lorsqu'une donnée SQL est convertie en caractère C, les 8 bits du caractère source sont représentés par deux caractères ASCII. Ces caractères sont des représentations ASCII des nombres au format hexadécimal. Par exemple, le binaire 00000001 est converti en "01" et le binaire 11111111 est converti en "FF".

Tableau 1. Conversion des LONGVARBINARY

mode	longueur	résultat
ODBC_BINMODE_PASSTHRU	0	passthru
ODBC_BINMODE_RETURN	0	passthru
ODBC_BINMODE_CONVERT	0	passthru
ODBC_BINMODE_PASSTHRU	0	passthru
ODBC_BINMODE_PASSTHRU	>0	passthru
ODBC_BINMODE_RETURN	>0	Tel quel
ODBC_BINMODE_CONVERT	>0	Caractère

Si **odbc_fetch_into()** est utilisé, passthru signifie qu'une chaîne vide sera retournée pour ces colonnes.

Si *result_id* vaut 0, ces paramètres seront appliqués aux nouveaux résultats.

> **Note :** La valeur par défaut de 4096 est 4096 et les valeurs par défaut de odbc_binmode est ODBC_BINMODE_RETURN. La gestion des colonnes binaires est aussi modifiée par **odbc_longreadlen()**.

odbc_close (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Ferme une connexion ODBC.

```
void odbc_close (resource connection_id)
```

odbc_close() ferme la connexion avec la source de données représentée par l'identifiant de connexion *connection_id*.

> **Note : odbc_close()** échouera s'il y a des transactions en cours sur cette connexion. Dans ce cas, la connexion restera ouverte.

odbc_close_all (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Ferme toutes les connexions ODBC

```
void odbc_close_all (void)
```

odbc_close_all() ferme toutes les connexions ODBC à des sources de données.

> **Note : odbc_close_all()** échouera s'il y a des transactions en cours sur cette connexion. Dans ce cas, la connexion restera ouverte.

odbc_commit (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Valide une transaction ODBC

```
int odbc_commit (resource connection_id)
```

odbc_commit() retourne TRUE en cas de succès, FALSE en cas d'erreur. Toutes les connexions en cours sur *connection_id* sont validées.

odbc_connect (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Connexion à une source

```
resource odbc_connect (string dsn, string user, string password [, int cursor_type])
```

odbc_connect() retourne un identifiant de connexion ODBC ou 0 (FALSE) en cas d'erreur.

L'identifiant de connexion retournée par cette fonction est nécessaire pour toutes les autres fonctions ODBC. Vous pouvez avoir de multiples connexions en même temps. Le quatrième paramètre fixe le type de pointeur de résultat utilisé pour cette connexion. Ce paramètre n'est généralement pas nécessaire, mais il peut être utile pour contourner certains problèmes ODBC.

Avec certains pilotes ODBC, l'exécution de procédures enregistrées complexes peut produire l'erreur suivante : "Cannot open a cursor on a stored procedure that has anything other than a single select statement in it", ce qui signifie : "Impossible de créer un pointeur de résultat dans une procédure enregistrée qui est réduite à une simple sélection (SELECT)). Utiliser l'option SQL_CUR_USE_ODBC permet d'éviter cette erreur. De plus, certains pilotes ne supportent

le paramètre optionnel de numéro de ligne dans **odbc_fetch_row()**. SQL_CUR_USE_ODBC peut aussi permettre de résoudre ces problèmes.

Les constantes suivantes sont définies comme type de pointeur :

- SQL_CUR_USE_IF_NEEDED
- SQL_CUR_USE_ODBC
- SQL_CUR_USE_DRIVER
- SQL_CUR_DEFAULT

Pour les connexions persistantes, reportez-vous à **odbc_pconnect()**.

odbc_cursor (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Lecture du pointeur de fiche courante (cursorname).

```
string odbc_cursor (resource result_id)
```

odbc_cursor() lit le pointeur de fiche courante (cursorname) pour le résultat *result_id*.

odbc_do (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Synonyme de **odbc_exec()**

```
string odbc_do (resource connection_id, string query)
```

odbc_do() exécute la requête *query* avec la connexion *connection_id*.

odbc_error (PHP 4 >= 4.0.5)

Lit le dernier code d'erreur

```
string odbc_error ([resource connection_id])
```

odbc_error() retourne un état ODVBC sur 6 chiffres, ou une chaîne vide s'il n'y avait plus d'erreurs. Si *connection_id* est spécifié, le dernier état ODBC de cette connexion est retourné. Si *connection_id* est omis, c'est le dernier état de n'importe quelle connexion qui est retourné.

Voir aussi **odbc_errormsg()** et **odbc_exec()**.

odbc_errormsg (PHP 4 >= 4.0.5)

Lit le dernier message d'erreur

```
string odbc_errormsg ([int connection_id])
```

odbc_errormsg() retourne une chaîne contenant le dernier message d'erreur ODBC, ou une chaîne vide s'il n'y avait pas d'erreur. Si *connection_id* est spécifié, le dernier état ODBC de cette connexion est retourné. Si *connection_id* est omis, c'est le dernier état de n'importe quelle connexion qui est retourné.

Voir aussi **odbc_error()** et **odbc_exec()**.

odbc_exec (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Prépare et exécute une requête SQL.

```
int odbc_exec (resource connection_id, string query_string)
```

odbc_exec() retourne FALSE en cas d'erreur, ou bien retourne un identifiant de résultat ODBC en cas d'exécution réussie.

odbc_exec() envoie une commande SQL à la source de données représentée par *connection_id*. Ce paramètre doit être un identifiant valide de connexion, retourné par **odbc_connect()** ou **odbc_pconnect()**.

Voir aussi : **odbc_prepare()** et **odbc_execute()** pour les éxecutions multiples de requêtes SQL.

odbc_execute (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Exécute une requête SQL préparée.

```
int odbc_execute (resource result_id [, array parameters_array])
```

odbc_execute() exécute une requête SQL préparée par **odbc_prepare()**. **odbc_execute()** retourne TRUE en cas d'exécution réussie, et FALSE sinon. Le tableau de paramètres *parameters_array* ne sert que si vous avez besoin de paramètrer votre requête.

odbc_fetch_into (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Lit une ligne de résultat, et la place dans un tableau.

```
int odbc_fetch_into (resource result_id [, int rownumber, array result_array])
```

odbc_fetch_into() retourne le nombre de colonnes dans le résultat, ou FALSE en cas d'erreur. *result_array* doit avoir été passé par référence, mais il peut être de n'importe quel type, étant donné qu'il sera converti en tableau. Le tableau contiendra les valeurs des colonnes, ces dernières étant numérotées à partir de 0.

Exemple 1. Exemple avec odbc_fetch_into() (avant PHP 4.0.6)

```
<?php
  $rc = odbc_fetch_into($res_id, $my_array);
?>
```

ou

```
<?php
  $rc = odbc_fetch_into($res_id, $row, $my_array);
  $rc = odbc_fetch_into($res_id, 1, $my_array);
?>
```

Jusqu'en PHP 4.0.5, le paramètre *result_array* n'a plus besoin d'être passé par référence.

Depuis PHP 4.0.6, le paramètre *rownumber* ne peut pas être passé comme une constante, mais comme une variable.

Exemple 2. Exemple avec odbc_fetch_into() (après PHP 4.0.6)

```
<?php
  $rc = odbc_fetch_into($res_id, $my_array);
?>
```

ou

```
<?php
  $row = 1;
  $rc = odbc_fetch_into($res_id, $row, $my_array);
?>
```

Evolution ultérieure : en PHP 4.1, **odbc_fetch_into()** aura le format suivant :

```
int odbc_fetch_into (int result_id, array result_array [, int rownumber])
```

Notez que le paramètre *rownumber* sera optionnel, tandis que *result_array* ne l'est pas.

odbc_fetch_row (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Lit une ligne de résultat.

```
int odbc_fetch_row (resource result_id [, int row_number])
```

Si **odbc_fetch_row()** a réussi, TRUE est retourné. S'il n'y avait plus de ligne, ou en cas d'erreur, FALSE est retourné.

odbc_fetch_row() lit une ligne dans le résultat identifié par *result_id* et retourné par **odbc_do()** ou **odbc_exec()**. Après **odbc_fetch_row()**, les champs seront accessibles avec la fonction **odbc_result()**.

Si *row_number* est omis, *row_number* va tenter de lire la prochaine ligne dans le résultat. Des appels répétés à **odbc_fetch_row()** avec et sans paramètre *row_number* peuvent être combinés librement.

Pour passer en revue toutes les lignes d'un résultat plusieurs fois, vous pouvez appeler **odbc_fetch_row()** avec row_number = 1, puis continue à appeler **odbc_fetch_row()** sans le paramètre *row_number* pour passer en revue tout le résultat. Si un pilote ne supporte pas la lecture des lignes par numéro, le paramètre sera ignoré.

odbc_field_name (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Lit le nom de la colonne.

```
string odbc_field_name (resource result_id, int field_number)
```

odbc_field_name() lit le nom de la colonne dont l'index est *field_number*. La numérotation des champs commence à 1. FALSE est retourné en cas d'erreur.

odbc_field_num (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Numéro de colonne

```
int odbc_field_num (resource result_id, string field_name)
```

odbc_field_num() retourne le numéro de la colonne nommée *field_name*. Ce numéro correspond à l'index du champs dans le résultat ODBC. La numérotation commence à 1. FALSE est retourné en cas d'erreur.

odbc_field_type (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Type de données d'un champs.

```
string odbc_field_type (resource result_id, int field_number)
```

odbc_field_type() retourne le type de données SQL d'un champs, identifié par son index. La numérotation des champs commence à 1.

odbc_field_len (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Lit la longueur d'un champs.

```
int odbc_field_len (resource result_id, int field_number)
```

odbc_field_len() retourne la longueur du champs référence par le nombre *field_number*, dans la connexion ODBC *result_id*. Les numéros de champs commencent à 1.

odbc_field_precision (PHP 4 >= 4.0.0)

Alias de **odbc_field_len()**

```
string odbc_field_precision (resource result_id, int field_number)
```

odbc_field_precision() retourne la précision du champs référéncé par son numéro *field_number*, dans le résultat ODBC *result_id*.

Voir aussi : **odbc_field_scale()** pour connaître l'échelle d'un nombre à virgule flottante.

odbc_field_scale (PHP 4 >= 4.0.0)

Lit l'échelle d'un champs

```
string odbc_field_scale (resource result_id, int field_number)
```

odbc_field_precision() retourne l'échelle du champs référéncé par son numéro de champs *field_number* dans le résultat ODBC *result_id*.

odbc_free_result (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Libère les ressources associées à un résultat

```
int odbc_free_result (resource result_id)
```

odbc_free_result() retourne toujours TRUE.

odbc_free_result() n'est nécessaire que si vous craignez d'utiliser trop de mémoire lors de l'exécution de votre script. Tous les résultats en mémoire seront libérés dès la fin du script. Mais, si vous êtes sÛr que vous n'aurez plus besoin d'un résultat jusqu'à la fin de votre script, vous pouvez appeler **odbc_free_result()**, et la mémoire associée à *result_id* sera libérée.

Note : Si auto-validation est désactivée (voir **odbc_autocommit()**) et que vous appelez **odbc_free_result()** avant de valider vos requêtes, toutes les transactions préparées seront annulées.

odbc_longreadlen (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Gestion des colonnes de type LONG.

```
int odbc_longreadlen (resource result_id, int length)
```

Types ODBC SQL affectés: LONG, LONGVARBINARY.

Le nombre d'octets retournés à PHP est contrôlé par le paramètre *length*. Si sa valeur est 0, les colonnes de type Long seront transformées en chaîne vide.

Note : La gestion des types LONGVARBINARY est aussi affectée par **odbc_binmode()**.

odbc_num_fields (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Nombre de colonnes dans un résultat

```
int odbc_num_fields (resource result_id)
```

odbc_num_fields() retourne le nombre de colonnes dans un résultat ODBC. **odbc_num_fields()** retournera -1 en cas d'erreur. L'argument est un identifiant de résultat valide, retourné par **odbc_exec()**.

odbc_pconnect (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Ouvre une connexion persistante à une source de données.

```
resource odbc_pconnect (string dsn, string user, string password [, int cursor_type])
```

odbc_pconnect() retourne un identifiant de connexion ODBC ou 0 (FALSE) en cas d'erreur. **odbc_pconnect()** se comporte de manière similaire à **odbc_connect()**, mais la connexion ouverte n'est pas vraiment terminée lorsque le script est terminé. Les prochaines requêtes qui se feront sur une connexion dont les *dsn*, *user*, *password* sont les mêmes que celle-ci (avec **odbc_connect()** et **odbc_pconnect()**) réutiliseront la connexion ouverte.

Note : Les connexions persistantes n'ont aucun effet si PHP est utilisé comme CGI.

Pour plus de détails sur le paramètre optionnel *cursor_type*, voyez **odbc_connect()**. Pour plus de détails sur les connexions persistantes, reportez-vous à la FAQ PHP.

odbc_prepare (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Prépare une commande pour l'exécution

```
resource odbc_prepare (resource connection_id, string query_string)
```

odbc_prepare() prépare une commande pour l'exécution.

odbc_prepare() retourne un identifiant de résultat ODBC si la commande SQL a été préparée avec succès. L'identifiant peut être utilisé plus tard pour exécuter la commande avec **odbc_execute()**.

odbc_num_rows (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Nombre de ligne dans un résultat.

```
int odbc_num_rows (odbc_prepare result_id)
```

odbc_num_rows() retourne le nombre de lignes dans un résultat ODBC. **odbc_num_rows()** retournera -1 en cas d'erreur. Pour les commandes INSERT, UPDATE et DELETE, **odbc_num_rows()** retourne le nombre de ligne affectées. Pour les commandes SELECT, ce PEUT le nombre de lignes disponibles, mais ce n'est pas certain.

Note: **odbc_num_rows()** après un SELECT retournera -1 avec de nombreux pilotes.

odbc_result (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Lit les données de résultat.

```
string odbc_result (odbc_prepare result_id, mixed field)
```

odbc_result() retourne le contenu d'un champs.

field peut être aussi bien un entier, contenant le numéro de colonne du champs, dans le résultat, ou bien une chaîne de caractère, qui représente le nom du champs. Par exemple:

```
<?php
$item_3 = odbc_result($Query_ID, 3 );
$item_val = odbc_result($Query_ID, "val");
?>
```

Le premier appel à **odbc_result()** retourne la valeur du troisième champs de la ligne courante, du résultat *result_id*. Le deuxième appel à **odbc_result()** retourne la valeur du troisième champs dont le nom est "val" de la ligne courante, du résultat *result_id*. Une erreur survient si le paramètre de colonne est inférieur à 1, ou dépasse le nombre de colonnes du résultat. De la même manière, une erreur survient si le nom du champs passé ne correspond à aucun champs dans le résultat.

Les index de champs commencent à 1. Pour plus d'informations sur la façon de lire des colonnes de type binaire ou long, reportez-vous à **odbc_binmode()** et **odbc_longreadlen()**.

odbc_result_all (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Affiche le résultat sous la forme d'une table HTML.

```
int odbc_result_all (odbc_prepare result_id [, string format])
```

odbc_result_all() retourne le nombre de lignes dans le résultat, ou FALSE en cas d'erreur.

odbc_result_all() affiche toutes les lignes d'un résultat. L'affichage se fait au format HTML. Avec l'option *format*, il est possible de modifier l'aspect global de la table.

odbc_rollback (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Annule une transaction

```
int odbc_rollback (odbc_prepare connection_id)
```

odbc_rollback() annule toutes les transactions sur la connexion *connection_id*. **odbc_rollback()** retourne TRUE en cas de succès, et FALSE en cas d'échec.

odbc_setoption (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Modifie les paramètres ODBC.

```
int odbc_setoption (resource id, int function, int option, int param)
```

odbc_setoption() donne accès aux options ODBC pour une connexion particulière ou un résultat de requête. Elle a été écrite pour aider à la résolution de problème liés aux pilotes ODBC récalcitrants. Vous aurez sÛrement à utiliser **odbc_setoption()** si vous êtes un programmeur ODBC et que vous comprenez les divers effets des options disponibles. Vous aurez aussi besoin d'un bon manuel de référence pour comprendre les options et leur usage. Différentes versions de pilotes supportent différentes versions d'options.

Etant donné que les effets peuvent varier d'un pilote à l'autre, l'utilisation de **odbc_setoption()** dans des scripts voués à être livrés au public est très fortement déconseillée. De plus, certaines options ODBC ne sont pas disponibles car elles doivent être fixées avant l'établissement de la connexion. Cependant, si dans un cas bien spécifique, **odbc_setoption()** vous permet d'utiliser PHP sans que votre patron vous pousse à utiliser un produit commercial, alors cela n'a pas d'importance.

Id est un identifiant de connexion, ou un identifiant de résultat, pour lequel vous souhaitez modifier des options. Pour SQLSetConnectOption(), c'est un identifiant de connexion. Pour SQLSetStmtOption(), c'est un identifiant de résultat.

function est la fonction ODBC à utiliser. La valeur doit être de 1 pour utiliser SQLSetConnectOption() et 2 pour SQLSetStmtOption().

Le paramètre *option* est l'option à modifier.

Le paramètre *param* est la valeur de l'option *option*.

Exemple 1. Exemple de modification d'option ODBC

```
<?php
// 1. L'option 102 de SQLSetConnectOption() est SQL_AUTOCOMMIT.
// 1 de SQL_AUTOCOMMIT est SQL_AUTOCOMMIT_ON.
//    Cet exemple a le meme effet que
//    odbc_autocommit($conn, TRUE);
odbc_setoption($conn, 1, 102, 1);
// 2. Option 0 de SQLSetStmtOption() est SQL_QUERY_TIMEOUT.
//    Cet exemple fixe le délai d'expiration à 30 secondes.
$result = odbc_prepare($conn, $sql);
odbc_setoption($result, 2, 0, 30);
odbc_execute($result);
?>
```

odbc_tables (PHP 3>= 3.0.17, PHP 4 >= 4.0b4)

Liste les tables d'une source.

```
int odbc_tables (odbc_setoption connection_id [, string qualifier [, string owner [,
string name [, string types]]]])
```

odbc_tables() liste toutes les tables de la source et retourne un identifiant de résultat ODBC, ou bien FALSE en cas d'erreur.

Le résultat contient les colonnes suivantes :

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- TABLE_TYPE
- REMARKS

Le résultat est ordonné grâce aux options TABLE_TYPE, TABLE_QUALIFIER, TABLE_OWNER et TABLE_NAME.

Les paramètres *owner* et *name* acceptent des masques de recherche ('%' pour remplacer zéro ou plus caractères, et '_' pour n'en remplacer qu'un seul).

Pour supporter les énumérations de qualificateurs propriétaires et types de table, la sémantique suivante pour les paramètres *qualifier*, *owner*, *name* et *table_type* sont disponibles :

- Si *qualifier* est un signe de pourcentage (%), et *owner* et *name* sont des chaînes vides, alors le résultat contient la liste des qualifiers valides pour la source. (toutes les colonnes hormis TABLE_QUALIFIER contiennent NULL).
- Si *owner* est un signe de pourcentage (%), et *qualifier* et *name* sont des chaînes vides, alors le résultat contient la liste des propriétaires de la source (toutes les colonnes hormis TABLE_OWNER contiennent NULL).
- Si *table_type* est un signe de pourcentage (%), et *qualifier*, *owner* et *name* sont des chaînes vides, alors le résultat contient la liste des types de tables de la source (toutes les colonnes hormis TABLE_TYPE contiennent NULL).

Si *table_type* n'est pas une chaîne vide, il doit contenir une liste de valeurs, séparées par des virgules, qui representent les types recherchés. Chaque valeur peut être insérée entre guillemets simples ('), ou sans guillemets. Par exemple "'TABLE','VIEW'" ou "TABLE, VIEW". Si la source de données ne supporte pas un type de table donné, **odbc_tables()** ne retournera aucun résultat pour ce type.

Voir aussi **odbc_tableprivileges()** pour connaître les droits associés.

odbc_tableprivileges (PHP 4 >= 4.0b4)

Liste les tables et leurs privilèges

```
int odbc_tableprivileges (resource connection_id [, string qualifier [, string owner [,
string name]]])
```

odbc_tableprivileges() liste les tables de la source et leurs droits associés. **odbc_tableprivileges()** retourne un identifiant de résultat ODBC, ou bien FALSE en cas d'erreur.

Le résultat possède les colonnes suivantes :

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME

- GRANTOR
- GRANTEE
- PRIVILEGE
- IS_GRANTABLE

Le résultat est ordonné par TABLE_QUALIFIER, TABLE_OWNER et TABLE_NAME.

Les paramètres *owner* et *name* acceptent des masques de recherche (’%’ pour remplacer zéro ou plus caractères, et ’_’ pour n’en remplacer qu’un seul).

odbc_columns (PHP 4 >= 4.0b4)

Liste les colonnes d’une table

```
int odbc_columns (resource connection_id [, string qualifier [, string owner [, string
table_name [, string column_name]]]])
```

odbc_columns() liste toutes les colonnes de la source dedonnées. **odbc_columns()** retourne un identifiant de résultat ODBC, ou bien FALSE en cas d’erreur.

Le résultat possède les colonnes suivantes :

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- COLUMN_NAME
- DATA_TYPE
- TYPE_NAME
- PRECISION
- LENGTH
- SCALE
- RADIX
- NULLABLE
- REMARKS

Le résultat est ordonné par TABLE_QUALIFIER, TABLE_OWNER et TABLE_NAME.

Les paramètres *owner*, *column_name* et *table_name* acceptent des masques de recherche (’%’ pour remplacer zéro ou plus caractères, et ’_’ pour n’en remplacer qu’un seul).

Voir aussi **odbc_columnprivileges()** pour connaître les droits associés.

odbc_columnprivileges (PHP 4 >= 4.0b4)

Liste les colonnes et leurs droits associés

```
int odbc_columnprivileges (resource connection_id [, string qualifier [, string owner [,
string table_name [, string column_name]]]])
```

odbc_columnprivileges() liste les colonnes et leurs droits associés pour la table *table_name*. **odbc_columnprivileges()** retourne un identifiant de résultat ODBC, ou bien FALSE en cas d'erreur.

Le résultat possède les colonnes suivantes :

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- GRANTOR
- GRANTEE
- PRIVILEGE
- IS_GRANTABLE

Le résultat est ordonné par TABLE_QUALIFIER, TABLE_OWNER et TABLE_NAME.

Le paramètre *column_name* accepte des masques de recherche ('%' pour remplacer zéro ou plus caractères, et '_' pour n'en remplacer qu'un seul).

odbc_gettypeinfo (PHP 4 >= 4.0b4)

Liste les types de données supportés par une source

```
int odbc_gettypeinfo (resource connection_id [, int data_type])
```

odbc_gettypeinfo() liste les types de données qui sont supportées par une source. **odbc_gettypeinfo()** retourne un identifiant de résultat, ou FALSE en cas d'erreur. L'argument optionnel *data_type* peut être utilisé pour restreindre les informations à un seul type de données.

Le résultat possède les colonnes suivantes :

- TYPE_NAME
- DATA_TYPE
- PRECISION
- LITERAL_PREFIX
- LITERAL_SUFFIX
- CREATE_PARAMS
- NULLABLE
- CASE_SENSITIVE
- SEARCHABLE
- UNSIGNED_ATTRIBUTE
- MONEY
- AUTO_INCREMENT
- LOCAL_TYPE_NAME
- MINIMUM_SCALE
- MAXIMUM_SCALE

Le résultat est ordonné par DATA_TYPE et TYPE_NAME.

odbc_primarykeys (PHP 4 >= 4.0b4)

Liste les colonnes utilisées dans une clé primaire

```
int odbc_primarykeys (resource connection_id, string qualifier, string owner, string table)
```

odbc_primarykeys() liste les colonnes utilisées dans une clé primaire de la table *table*. **odbc_primarykeys()** retourne un identifiant de résultat, ou FALSE en cas d'erreur.

Le résultat possède les colonnes suivantes :

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- COLUMN_NAME
- KEY_SEQ
- PK_NAME

odbc_foreignkeys (PHP 4 >= 4.0b4)

Liste les clés étrangères

```
int odbc_foreignkeys (resource connection_id, string pk_qualifier, string pk_owner, string pk_table, string fk_qualifier, string fk_owner, string fk_table)
```

odbc_foreignkeys() liste les clés étrangèes utilisées dans la table *pk_table*. **odbc_foreignkeys()** retourne un identifiant de résultat, ou FALSE en cas d'erreur.

Le résultat possède les colonnes suivantes :

- PKTABLE_QUALIFIER
- PKTABLE_OWNER
- PKTABLE_NAME
- PKCOLUMN_NAME
- FKTABLE_QUALIFIER
- FKTABLE_OWNER
- FKTABLE_NAME
- FKCOLUMN_NAME
- KEY_SEQ
- UPDATE_RULE
- DELETE_RULE
- FK_NAME
- PK_NAME

Si *pk_table* contient un nom de table, **odbc_foreignkeys()** retourne la clé primaire de la table *pk_table*, et toutes les clés étrangères qui y font référence.

Si *fk_table* contient un nom de table, **odbc_foreignkeys()** retourne la liste des clés étrangères de la table *fk_table*, et les clés primaires (d'autres tables) qui y font référence.

Si *pk_table* et *fk_table* contiennent des noms de tables, **odbc_foreignkeys()** retourne la liste des clés étrangères de la table *fk_table* qui utilisent la clé primaire de la table *pk_table*. Cette liste devrait ne contenir qu'une clé au mieux.

odbc_procedures (PHP 4 >= 4.0b4)

Liste les procédure stockées

```
int odbc_procedures (resource connection_id [, string qualifier [, string owner [, string
name]]])
```

odbc_procedures() liste toutes les procédures stockées dans la source de données. **odbc_procedures()** retourne un identifiant de résultat, ou FALSE en cas d'erreur.

Le résultat possède les colonnes suivantes :

- PROCEDURE_QUALIFIER
- PROCEDURE_OWNER
- PROCEDURE_NAME
- NUM_INPUT_PARAMS
- NUM_OUTPUT_PARAMS
- NUM_RESULT_SETS
- REMARKS
- PROCEDURE_TYPE

Les paramètres *owner* et *name* acceptent des masques de recherche ('%' pour remplacer zéro ou plus caractères, et '_' pour n'en remplacer qu'un seul).

odbc_procedurecolumns (PHP 4 >= 4.0b4)

Liste les paramètres des procédures

```
int odbc_procedurecolumns (resource connection_id [, string qualifier [, string owner [,
string proc [, string column]]]])
```

odbc_procedurecolumns() list les paramètres d'entrée et de sortie, ainsi que les colonnes utilisées dans les procédures désignées par les paramètres. **odbc_procedurecolumns()** etourne un identifiant de résultat, ou FALSE en cas d'erreur.

Le résultat possède les colonnes suivantes :

- PROCEDURE_QUALIFIER
- PROCEDURE_OWNER
- PROCEDURE_NAME
- COLUMN_NAME
- COLUMN_TYPE
- DATA_TYPE
- TYPE_NAME
- PRECISION

- LENGTH
- SCALE
- RADIX
- NULLABLE
- REMARKS

Le résultat est ordonné par PROCEDURE_QUALIFIER, PROCEDURE_OWNER, PROCEDURE_NAME et COLUMN_TYPE.

Les paramètres *owner*, *proc* et *column* acceptent des masques de recherche ('%' pour remplacer zéro ou plus caractères, et '_' pour n'en remplacer qu'un seul).

odbc_specialcolumns (PHP 4 >= 4.0b4)

Retourne l'ensemble optimal de colonnes, qui permettent de définir uniquement une ligne dans une table

```
int odbc_specialcolumns (resource connection_id, int type, string qualifier, string owner,
string table, int scope, int nullable)
```

Lorsque le *type* est SQL_BEST_ROWID, **odbc_specialcolumns()** retourne la ou les colonnes qui permettent de repérer uniquement chaque ligne d'une table.

Lorsque le type *type* est SQL_ROWVER, **odbc_specialcolumns()** retourne l'ensemble optimal de colonne tel qu'en lisant les valeurs de ces colonnes, on puisse spécifier n'importe quelle ligne de manière unique.

odbc_specialcolumns() retourne un identifiant de résultat, ou FALSE en cas d'erreur.

Le résultat possède les colonnes suivantes :

- SCOPE
- COLUMN_NAME
- DATA_TYPE
- TYPE_NAME
- PRECISION
- LENGTH
- SCALE
- PSEUDO_COLUMN

Le résultat est ordonné par SCOPE.

odbc_statistics (PHP 4 >= 4.0b4)

Calcule des statistiques sur une table

```
int odbc_statistics (resource connection_id, string qualifier, string owner, string
table_name, int unique, int accuracy)
```

odbc_statistics() effectue quelques statistiques sur une tables et ses index. **odbc_statistics()** retourne un identifiant de résultat, ou FALSE en cas d'erreur.

Le résultat possède les colonnes suivantes :

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- NON_UNIQUE
- INDEX_QUALIFIER
- INDEX_NAME
- TYPE
- SEQ_IN_INDEX
- COLUMN_NAME
- COLLATION
- CARDINALITY
- PAGES
- FILTER_CONDITION

Le résultat est ordonné par NON_UNIQUE, TYPE, INDEX_QUALIFIER, INDEX_NAME et SEQ_IN_INDEX.

LVIII. Oracle 8

Ces fonctions vous permettront d'accéder aux serveurs Oracle8 et Oracle7. Elles utilisent l'interface Oracle8 Call-Interface (oci8). Vous aurez donc besoin des librairies clientes Oracle8 pour pouvoir les utiliser.

Il faut noter que cette extension est plus souple que l'extension Oracle officielle. Elle supporte notamment les liaisons entre les variables globales et locales de PHP avec des emplacements Oracle; elle supporte complètement les types LOB, FILE et ROWID et vous permet d'utiliser des variables de définitions personnalisables.

Avant d'utiliser cette extension, assurez-vous que vous avez bien paramétré vos variables d'environnement Oracle, ainsi que votre démon utilisateur. Les variables dont vous pouvez avoir besoin sont :

- ORACLE_HOME
- ORACLE_SID
- LD_PRELOAD
- LD_LIBRARY_PATH
- NLS_LANG
- ORA_NLS33

Après avoir configuré ces variables pour votre utilisateur "serveur web", assurez-vous aussi d'ajouter cet utilisateur (nobody, www) au group Oracle.

Si votre serveur web ne démarre pas, ou crashe au démarrage : Vérifiez que Apache a bien été compilé avec la librairie pthread :

```
# ldd /www/apache/bin/httpd
    libpthread.so.0 => /lib/libpthread.so.0 (0x4001c000)
    libm.so.6 => /lib/libm.so.6 (0x4002f000)
    libcrypt.so.1 => /lib/libcrypt.so.1 (0x4004c000)
    libdl.so.2 => /lib/libdl.so.2 (0x4007a000)
    libc.so.6 => /lib/libc.so.6 (0x4007e000)
    /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Si la libpthread n'est pas listée, vous devez réinstaller Apache :

```
# cd /usr/src/apache_1.3.xx
# make clean
# LIBS=-lpthread ./config.status
# make
# make install
```

Exemple 1. Aide oci

```
<?php
// par sergo@bacup.ru
// Utilisez l'option : oci_DEFAULT pour éxécuter les commandes avec un délai
ociExecute($stmt, oci_DEFAULT);
// pour lire les données après lecture, utilisez :
$result = ociResult($stmt, $n);
if (is_object ($result)) $result = $result->load();
// Pour les commandes INSERT ou UPDATE utilisez:
$sql = "insert into table (field1, field2) values (field1 = 'value',
 field2 = empty_clob()) returning field2 into :field2";
ociparse($conn, $sql);
$clob = ociNewDescriptor($conn, oci_D_LOB);
```

```
ociBindByName ($stmt, ":field2", &$clob, -1, oci_B_CLOB);
ociexecute($stmt, oci_DEFAULT);
$clob->save ("Du texte");
ocicommit($conn);
?>
```

Vous pouvez facilement accéder aux procédures stockées, de la même façon que vous le feriez par ligne de commande :

Exemple 2. Utilisation de procédures stockées

```
<?php
// par webmaster@remoterealty.com
$sth = ociparse ( $dbh, "begin sp_newaddress( :address_id, '$firstname',
 '$lastname', '$company', '$address1', '$address2', '$city', '$state',
 '$postalcode', '$country', :error_code );end;" );
// Ce script appelle la procédure stockée sp_newaddress, avec address_id qui est
// une variable entrante/sortante et :error_code une variable sortante.
// Lorsque vous les liez :
   ociBindByName ( $sth, ":address_id", $addr_id, 10 );
   ociBindByName ( $sth, ":error_code", $errorcode, 10 );
   ociExecute ( $sth );
?>
```

ociDefineByName (PHP 3>= 3.0.7, PHP 4 >= 4.0b1)

Utilise une variable PHP pour la phase de définition, dans une commande SELECT.

```
int ocidefinebyname (resource stmt, string Column-Name, mixed variable [, int type])
```

ocidefinebyname() copie les valeurs issues de colonnes SQL *Column-Name* dans les variables PHP. Méfiez-vous des colonnes Oracle qui sont toutes en majuscule, tandis que dans les SELECT, vous pouvez aussi les écrire en minuscules. **ocidefinebyname()** s'attends à ce que *Column-Name* soit en majuscules. Si vous définissez une variable qui n'existe pas dans la commande SELECT, vous ne serez pas prévenu par une erreur.

Si vous avez besoin de définir un type de données abstrait, tel que (LOB/ROWID/BFILE), vous devez lui allouer la mémoire avec **ocinewdescriptor()**. Reportez-vous aussi à **ocibindbyname()**.

Exemple 1. ociDefineByName

```
<?php
/* Exemple ociDefineByPos par thies@thieso.net (980219) */
$conn = ociLogon("scott","tiger");
$stmt = ociparse($conn,"select empno, ename from emp");
/* La définition DOIT être faîte AVANT ociexecute! */
ociDefineByName($stmt,"EMPNO",&$empno);
ociDefineByName($stmt,"ENAME",&$ename);
ociexecute($stmt);
while (ociFetch($stmt)) {
    echo "empno:".$empno."\n";
    echo "ename:".$ename."\n";
}
ociFreeStatement($stmt);
ociLogoff($conn);
?>
```

ociBindByName (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Utilise une variable PHP pour la phase de définition, dans un SELECT.

```
int ocibindbyname (resource stmt, string ph_name, mixed &variable, int length, int [type])
```

ocibindbyname() relie la variable PHP *variable* à l'emplacement Oracle *ph_name*. Son utilisation (comme entrée ou comme sortie) sera définie à l'exécution, et l'espace nécessaire sera alloué. Le paramètre de longueur *length* fixe la taille maximum pour la liaison. Si vous affectez une longueur de -1, **ocibindbyname()** utilisera la longueur de variable comme maximum.

Si vous devez lier des types abstraits de données (LOB/ROWID/BFILE), vous devrez l'allouer dans un premier temps, avec **ocinewdescriptor()**. La longueur *length* ne sert pas pour ces types et devrait être fixée à -1. La variable *type* indique au serveur Oracle, quel type de pointeur va être utilisé. Les valeurs possibles sont : oci_B_FILE (Fichier binaires), oci_B_CFILE (Fichier texte), oci_B_CLOB (LOB- texte), oci_B_BLOB (LOB binaire) et oci_B_ROWID (ROWID).

Exemple 1. ociDefineByName

```
<?php
/* Exemple ociBindByPos par thies@thieso.net (980221)
  Insère 3 lignes dans emp, et utilise ROWID pour mettre à jour
  les lignes, juste après l'insertion.
*/
$conn = ociLogon("scott","tiger");
$stmt = ociparse($conn,"insert into emp (empno, ename) ".
   "values (:empno,:ename) ".
   "returning ROWID into :rid");
$data = array(1111 => "Larry", 2222 => "Bill", 3333 => "Jim");
```

```
$rowid = ociNewDescriptor($conn,oci_D_ROWID);
ociBindByName($stmt,":empno",&$empno,32);
ociBindByName($stmt,":ename",&$ename,32);
ociBindByName($stmt,":rid",&$rowid,-1,oci_B_ROWID);
$update = ociparse($conn,"update emp set sal = :sal where ROWID = :rid");
ociBindByName($update,":rid",&$rowid,-1,oci_B_ROWID);
ociBindByName($update,":sal",&$sal,32);
$sal = 10000;
while (list($empno,$ename) = each($data)) {
ociexecute($stmt);
ociexecute($update);
}
$rowid->free();
ociFreeStatement($update);
ociFreeStatement($stmt);
$stmt = ociparse($conn,"select * from emp where empno in (1111,2222,3333)");
ociexecute($stmt);
while (ociFetchInto($stmt,&$arr,oci_ASSOC)) {
var_dump($arr);
}
ociFreeStatement($stmt);
/* Effacement des lignes inutiles dans la table emp .... */
$stmt = ociparse($conn,"delete from emp where empno in (1111,2222,3333)");
ociexecute($stmt);
ociFreeStatement($stmt);
ociLogoff($conn);
?>
```

ociLogon (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Etablit une connexion à un serveur Oracle.

```
resource ocilogon (string username, string password [, string db])
```

ocilogon() retourne un identifiant de connexion, nécessaire à la plupart des fonctions oci. Si l'option ORACLE_SID n'est pas précisée, PHP utilisera la variable d'environnement ORACLE_SID pour déterminer le serveur de connexion.

Les connexions sont partagées, à l'intérieur d'une même page avec **ocilogon()**. Cela signifie que COMMIT et ROLLBACK s'appliquent à toutes les transactions commencées à l'intérieur d'une même page, même si vous avez créé de multiples connexions.

Cet exemple montre comment les connexions sont partagées :

Exemple 1. ociLogon

```
<?php
print "<HTML><PRE>";
$db = "";
$c1 = ocilogon("scott","tiger",$db);
$c2 = ocilogon("scott","tiger",$db);
function create_table($conn)
{ $stmt = ociparse($conn,"create table scott.hallo (test
varchar2(64))");
  ociexecute($stmt);
  echo $conn." created table\n\n";
}
function drop_table($conn)
{ $stmt = ociparse($conn,"drop table scott.hallo");
  ociexecute($stmt);
  echo $conn." dropped table\n\n";
}
function insert_data($conn)
```

```
{ $stmt = ociparse($conn,"insert into scott.hallo va-
lues('$conn' || ' ' || to_char(sysdate,'DD-MON-YY HH24:MI:SS'))");
  ociexecute($stmt,oci_DEFAULT);
  echo $conn." inserted hallo\n\n";
}
function delete_data($conn)
{ $stmt = ociparse($conn,"delete from scott.hallo");
  ociexecute($stmt,oci_DEFAULT);
  echo $conn." deleted hallo\n\n";
}
function commit($conn)
{ ocicommit($conn);
  echo $conn." commited\n\n";
}
function rollback($conn)
{ ocirollback($conn);
  echo $conn." rollback\n\n";
}
function select_data($conn)
{ $stmt = ociparse($conn,"select * from scott.hallo");
  ociexecute($stmt,oci_DEFAULT);
  echo $conn."---selecting\n\n";
  while (ocifetch($stmt))
    echo $conn." <".ociresult($stmt,"TEST").">\n\n";
  echo $conn."---done\n\n";
}
create_table($c1);
insert_data($c1);   // Insertion d'une ligne avec c1
insert_data($c2);   // Insertion d'une ligne avec c2
select_data($c1);   // Les résultats des deux insertions sont retournés
select_data($c2);
rollback($c1);      // Annulation avec c1
select_data($c1);   // Les résultats des deux insertions sont annulés
select_data($c2);
insert_data($c2);   // Insertion d'une ligne avec c2
commit($c2);        // Validation avec using c2
select_data($c1);   // Le résultat de c2 est retourné
delete_data($c1);   // Effacement de toutes les lignes avec c1
select_data($c1);   // Aucune ligne n'est retournée
select_data($c2);   // Aucune ligne n'est retournée
commit($c1);        // Validation avec c1
select_data($c1);   // Aucune ligne n'est retournée
select_data($c2);   // Aucune ligne n'est retournée
drop_table($c1);
print "</PRE></HTML>";
?>
```

Voir aussi **ociplogon()** et **ocinlogon()**.

ociPLogon (PHP 3>= 3.0.8, PHP 4 >= 4.0b1)

Connection persistante à un serveur Oracle.

```
resource ociplogon (string username, string password [, string db])
```

ociplogon() crée une connexion persistante à un serveur Oracle 8 et s'authentifie. Si l'option ORACLE_SID n'est pas spécifiée, PHP utilisera la variable d'environnement ORACLE_SID pour déterminer le serveur de connexion.

Voir aussi **ocilogon()** et **ocinlogon()**.

ociNLogon (PHP 3>= 3.0.8, PHP 4 >= 4.0b1)

Se connecte à un serveur Oracle avec une nouvelle connexion.

```
resource ocinlogon (string username, string password [, string db])
```

ocinlogon() crée une nouvelle connexion à un serveur Oracle et s'authentifie. Si l'option ORACLE_SID n'est pas spécifié, PHP utilisera la variable d'environnement ORACLE_SID pour déterminer le serveur de connexion.

ocinlogon() force le serveur à établir une nouvelle connexion. Cette fonction ne doit être utilisée que si vous voulez isoler un ensemble de transactions. Par défaut, les connexions sont partagées au niveau de la page, si vous utilisez la fonction **ocinlogon()** ou bien au niveau du processus web, si vous utilisez **ociplogon()**. Si vous avez de multiples connexions ouvertes avec **ocinlogon()**, les validations et annulations ne s'appliquent qu'à la connexion spécifée.

L'exemple ci-dessous montre l'utilisation des connexions séparées.

Exemple 1. ociNLogon

```
<?php
print "<HTML><PRE>";
$db = "";
$c1 = ocilogon("scott","tiger",$db);
$c2 = ocinlogon("scott","tiger",$db);
function create_table($conn)
{ $stmt = ociparse($conn,"create table scott.hallo (test
varchar2(64))");
  ociexecute($stmt);
  echo $conn." created table\n\n";
}
function drop_table($conn)
{ $stmt = ociparse($conn,"drop table scott.hallo");
  ociexecute($stmt);
  echo $conn." dropped table\n\n";
}
function insert_data($conn)
{ $stmt = ociparse($conn,"insert into scott.hallo va-
lues('$conn' || ' ' || to_char(sysdate,'DD-MON-YY HH24:MI:SS'))");
  ociexecute($stmt,oci_DEFAULT);
  echo $conn." inserted hallo\n\n";
}
function delete_data($conn)
{ $stmt = ociparse($conn,"delete from scott.hallo");
  ociexecute($stmt,oci_DEFAULT);
  echo $conn." deleted hallo\n\n";
}
function commit($conn)
{ ocicommit($conn);
  echo $conn." commited\n\n";
}
function rollback($conn)
{ ocirollback($conn);
  echo $conn." rollback\n\n";
}
function select_data($conn)
{ $stmt = ociparse($conn,"select * from scott.hallo");
  ociexecute($stmt,oci_DEFAULT);
  echo $conn."---selecting\n\n";
  while (ocifetch($stmt))
    echo $conn." <".ociresult($stmt,"TEST").">\n\n";
  echo $conn."---done\n\n";
}
create_table($c1);
insert_data($c1);
select_data($c1);
select_data($c2);
```

```
rollback($c1);
select_data($c1);
select_data($c2);
insert_data($c2);
commit($c2);
select_data($c1);
delete_data($c1);
select_data($c1);
select_data($c2);
commit($c1);
select_data($c1);
select_data($c2);
drop_table($c1);
print "</PRE></HTML>";
?>
```

Voir aussi **ocilogon()** et **ociplogon()**.

ociLogOff (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Déconnexion d'un serveur Oracle

```
int ocilogoff (resource connection)
```

ocilogoff() ferme la connexion Oracle.

ociexecute (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Exécute une commande

```
int ociexecute (resource statement [, int mode])
```

ociexecute() éxécute une commande déjà préparée (voir **ociparse()**). L'option *mode* vous permet de spécifier le mode d'exécution (par défaut, il est à oci_COMMIT_ON_SUCCESS). Si vous ne voulez pas que la commande soit automatiquement validée, utilisez le mode oci_DEFAULT.

ociCommit (PHP 3>= 3.0.7, PHP 4 >= 4.0b1)

Valide les transactions en cours.

```
int ocicommit (resource connection)
```

ocicommit() valide toutes les transactions en cours sur la connexion Oracle *connection*.

ociRollback (PHP 3>= 3.0.7, PHP 4 >= 4.0b1)

Annule les transactions en cours

```
int ocirollback (resource connection)
```

ocirollback() annule les transactions en cours sur la connexion Oracle *`connection`*.

ociNewDescriptor (PHP 3>= 3.0.7, PHP 4 >= 4.0b1)

Initialise un nouveau pointeur vide de LOB/FILE

```
resource ocinewdescriptor (resource connection [, int type])
```

ocinewdescriptor() alloue l'espace nécessaire pour stocker un descripteur, ou un pointeur de LOB. Les valeurs acceptées pour type sont oci_D_FILE, oci_D_LOB et oci_D_ROWID.

Exemple 1. ociNewDescriptor

```
<?php
    /* Ce script est fait pour être appelé dans un formulaire HTML
     * Il attends les variables $user, $password, $table, $where, et $commitsize
     * Le script efface alors les lignes sélectionnées avec ROWID et valide
     * l'effacement après chaque groupe de $commitsize lignes.
     * (Utilisez avec prudences, car il n'y a pas d'annulation possible).
     */
    $conn = ociLogon($user, $password);
    $stmt = ociparse($conn,"select rowid from $table $where");
    $rowid = ociNewDescriptor($conn,oci_D_ROWID);
    ociDefineByName($stmt,"ROWID",&$rowid);
    ociexecute($stmt);
    while ( ociFetch($stmt) ) {
       $nrows = ociRowCount($stmt);
       $delete = ociparse($conn,"delete from $table where ROWID = :rid");
       ociBindByName($delete,":rid",&$rowid,-1,oci_B_ROWID);
       ociexecute($delete);
       print "$nrows\n";
       if ( ($nrows % $commitsize) == 0 ) {
           ociCommit($conn);
       }
    }
    $nrows = ociRowCount($stmt);
    print "$nrows effacées...\n";
    ociFreeStatement($stmt);
    ociLogoff($conn);
?>

<?php
    /* Ce script est fait pour être appelé depuis un formulaire HTML.
     * Il attends les variables $user, $password, $table, $where, et $commitsize,
     * données par le formulaire. Le script efface
     * les lignes sélectionnées avec ROWID est valide les transactions
     * à chaque jeu de $commitsize lignes. (Attention : il n'y plus d'annulation) */
  if(!isset($lob_upload) || $lob_upload == 'none'){
?>
<form action="upload.php3" method="post" enctype="multipart/form-data">
Upload file: <input type="file" name="lob_upload"><br>
<input type="submit" value="Upload"> - <input type="reset">
</form>
<?php
  } else {
     // $lob_upload contains the temporary filename of the uploaded file
     $conn = ociLogon($user, $password);
     $lob = ociNewDescriptor($conn, oci_D_LOB);
     $stmt = ociparse($conn,"insert into $table (id, the_blob) val-
ues(my_seq.NEXTVAL, EMPTY_BLOB()) returning the_blob into :the_blob");
     ociBindByName($stmt, ':the_blob', &$lob, -1, oci_B_BLOB);
     ociexecute($stmt, oci_DEFAULT);
     if($lob->savefile($lob_upload)){
```

```
        ociCommit($conn);
        echo "Blob sauvé!\n";
     }else{
        echo "Impossible de sauver le Blob\n";
     }
     ociFreeDescriptor($lob);
     ociFreeStatement($stmt);
     ociLogoff($conn);
  }
?>
```

ociRowCount (PHP 3>= 3.0.7, PHP 4 >= 4.0b1)

Retourne le nombre de lignes affectées.

```
int ocirowcount (resource statement)
```

ocirowcount() retourne le nombre de lignes affectées par une commande de modification. Cette fonction ne vous indiquera pas le nombre de lignes retournées par un SELECT : il faut que les lignes aient été modifiées.

Exemple 1. ociRowCount

```
<?php
    print "<HTML><PRE>";
    $conn = ociLogon("scott","tiger");
    $stmt = ociparse($conn,"create table emp2 as select * from emp");
    ociexecute($stmt);
    print ociRowCount($stmt) . " rows inserted.<br>";
    ociFreeStatement($stmt);
    $stmt = ociparse($conn,"delete from emp2");
    ociexecute($stmt);
    print ociRowCount($stmt) . " rows deleted.<br>";
    ociCommit($conn);
    ociFreeStatement($stmt);
    $stmt = ociparse($conn,"drop table emp2");
    ociexecute($stmt);
    ociFreeStatement($stmt);
    ociLogOff($conn);
    print "</PRE></HTML>";
?>
```

ociNumCols (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Retourne le nombre de colonnes dans un résultat

```
int ocinumcols (resource stmt)
```

ocinumcols() retourne le nombre de colonnes dans un résultat.

Exemple 1. ociNumCols

```
<?php
    print "<HTML><PRE>\n";
    $conn = ociLogon("scott", "tiger");
    $stmt = ociparse($conn,"select * from emp");
    ociexecute($stmt);
    while ( ociFetch($stmt) ) {
        print "\n";
        $ncols = ociNumCols($stmt);
        for ( $i = 1; $i <= $ncols; $i++ ) {
            $column_name  = ociColumnName($stmt,$i);
            $column_value = ociResult($stmt,$i);
            print $column_name . ': ' . $column_value . "\n";
        }
        print "\n";
    }
    ociFreeStatement($stmt);
    ociLogoff($conn);
    print "</PRE>";
    print "</HTML>\n";
?>
```

ociResult (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Retourne la valeur d'une colonne dans une ligne lue

```
mixed ociresult (resource statement, mixed column)
```

ociresult() retourne les données de la colonne *column* dans la ligne courante (voir **ocifetch()**). **ocifetch()** retournera tout les types, sauf les types abstraits (ROWIDs, LOBs et FILEs).

ociFetch (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Modifie la prochaine ligne dans le pointeur interne de résultat.

```
int ocifetch (resource statement)
```

ocifetch() place la prochaine ligne (d'une commande SELECT) dans le pointeur interne de résultat.

ociFetchInto (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Retourne la ligne suivante dans un tableau.

```
int ocifetchinto (resource stmt, array &result [, int mode])
```

ocifetchinto() retourne la ligne suivante (pour une commande SELECT) dans le tableau *result*. **ocifetchinto()** écrasera le contenu de *result*. Par défaut, *result* sera un tableau à index numérique, commençant à 1, et qui contiendra toute les colonnes qui ne sont pas NULL.

L'option *mode* vous permet de modifier le comportement par défaut de la fonction. Vous pouvez passer plusieurs modes simplement en les additionnant (i.e. oci_ASSOC+oci_RETURN_NULLS). Les modes valides sont :

oci_ASSOC Retourne un tableau associatif.
oci_NUM Retourne un tableau à index numérique (DEFAULT, valeur par défaut)

`oci_RETURN_NULLS` Retourne les colonnes vides.
`oci_RETURN_LOBS` Retourne la valeur des objets LOB plutôt que leur descripteur.

ociFetchStatement (PHP 3>= 3.0.8, PHP 4 >= 4.0b1)

Retourne toutes les lignes d'un résultat.

```
int ocifetchstatement (resource stmt, array &variable)
```

ocifetchstatement() retourne toutes les lignes d'un résultat dans le tableau variable. **ocifetchstatement()** retourne le nombre de lignes retournées.

Exemple 1. ociFetchStatement

```
<?php
/* exemple ociFetchStatement par mbritton@verinet.com (990624) */
$conn = ociLogon("scott","tiger");
$stmt = ociparse($conn,"select * from emp");
ociexecute($stmt);
$nrows = ociFetchStatement($stmt,$results);
if ( $nrows > 0 ) {
   print "<TABLE BORDER=\"1\">\n";
   print "<TR>\n";
   while ( list( $key, $val ) = each( $results ) ) {
      print "<TH>$key</TH>\n";
   }
   print "</TR>\n";
   for ( $i = 0; $i < $nrows; $i++ ) {
      reset($results);
      print "<TR>\n";
      while ( $column = each($results) ) {
         $data = $column['value'];
         print "<TD>$data[$i]</TD>\n";
      }
      print "</TR>\n";
   }
   print "</TABLE>\n";
} else {
   echo "Rien n'a été trouvé<br>\n";
}
print "$nrows Records Selected<br>\n";
ociFreeStatement($stmt);
ociLogoff($conn);
?>
```

ociColumnIsNULL (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Teste si la valeur d'une colonne est NULL

```
int ocicolumnisnull (resource stmt, mixed column)
```

ocicolumnisnull() retourne `TRUE` si la colonne *`col`* du résultat *`stmt`* est NULL. Vous pouvez utiliser le numéro de colonne (l'indexation des colonnes commence à 1) ou le nom de la colonne, pour le paramètre *`col`*.

ociColumnName (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Retourne le nom d'une colonne.

```
string ocicolumnname (resource stmt, int col)
```

ocicolumnname() retourne le nom de la colonne numéro *col* (en commençant à 1).

Exemple 1. ociColumnName

```
<?php
    print "<HTML><PRE>\n";
    $conn = ociLogon("scott", "tiger");
    $stmt = ociparse($conn,"select * from emp");
    ociexecute($stmt);
    print "<TABLE BORDER=\"1\">";
    print "<TR>";
    print "<TH>Name</TH>";
    print "<TH>Type</TH>";
    print "<TH>Length</TH>";
    print "</TR>";
    $ncols = ociNumCols($stmt);
    for ( $i = 1; $i <= $ncols; $i++ ) {
        $column_name  = ociColumnName($stmt,$i);
        $column_type  = ociColumnType($stmt,$i);
        $column_size  = ociColumnSize($stmt,$i);
        print "<TR>";
        print "<TD>$column_name</TD>";
        print "<TD>$column_type</TD>";
        print "<TD>$column_size</TD>";
        print "</TR>";
    }
    ociFreeStatement($stmt);
    ociLogoff($conn);
    print "</PRE>";
    print "</HTML>\n";
?>
```

Voir aussi **ocinumcols()**, **ocicolumntype()** et **ocicolumnsize()**.

ociColumnSize (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Retourne la taille de la colonne.

```
int ocicolumnsize (resource stmt, mixed column)
```

ocicolumnsize() retourne la taille de la colonne. Vous pouvez utiliser l'index de colonne (l'indexation commence à 1) ou le nom de la colonne dans le paramètre *col*.

Exemple 1. ociColumnSize

```
<?php
    print "<HTML><PRE>\n";
    $conn = ociLogon("scott", "tiger");
    $stmt = ociparse($conn,"select * from emp");
    ociexecute($stmt);
    print "<TABLE BORDER=\"1\">";
    print "<TR>";
```

```
    print "<TH>Name</TH>";
    print "<TH>Type</TH>";
    print "<TH>Length</TH>";
    print "</TR>";
    $ncols = ociNumCols($stmt);
    for ( $i = 1; $i <= $ncols; $i++ ) {
        $column_name  = ociColumnName($stmt,$i);
        $column_type  = ociColumnType($stmt,$i);
        $column_size  = ociColumnSize($stmt,$i);
        print "<TR>";
        print "<TD>$column_name</TD>";
        print "<TD>$column_type</TD>";
        print "<TD>$column_size</TD>";
        print "</TR>";
    }
    print "</TABLE>";
    ociFreeStatement($stmt);
    ociLogoff($conn);
    print "</PRE>";
    print "</HTML>\n";
?>
```

Voir aussi **ocinumcols()**, **ocicolumnname()** et **ocicolumnsize()**.

ociColumnType (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Retourne le type de données d'une colonne.

```
mixed ocicolumntype (resource stmt, int col)
```

ocicolumntype() retourne le type de données de la colonne correspondant au numéro de colonne *col* dans le résultat *stmt* (les colonnes sont indexées à partir de 1).

Exemple 1. Exemple avec ocicolumntype()

```
<?php
    print "<HTML><PRE>\n";
    $conn = ociLogon("scott", "tiger");
    $stmt = ociparse($conn,"select * from emp");
    ociexecute($stmt);
    print "<TABLE BORDER=\"1\">";
    print "<TR>";
    print "<TH>Name</TH>";
    print "<TH>Type</TH>";
    print "<TH>Length</TH>";
    print "</TR>";
    $ncols = ociNumCols($stmt);
    for ( $i = 1; $i <= $ncols; $i++ ) {
        $column_name  = ociColumnName($stmt,$i);
        $column_type  = ociColumnType($stmt,$i);
        $column_size  = ociColumnSize($stmt,$i);
        print "<TR>";
        print "<TD>$column_name</TD>";
        print "<TD>$column_type</TD>";
        print "<TD>$column_size</TD>";
        print "</TR>";
    }
    ociFreeStatement($stmt);
    ociLogoff($conn);
    print "</PRE>";
```

```
    print "</HTML>\n";
?>
```

Voir aussi **ocinumcols()**, **ocicolumnname()** et **ocicolumnsize()**.

ociServerVersion (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Retourne une chaîne contenant les informations de version du serveur.

```
string ociserverversion (resource conn)
```

ociserverversion() retourne une chaîne contenant les informations de version du serveur

Exemple 1. Exemple avec ociserverversion()

```
<?php
   $conn = ociLogon("scott","tiger");
   print "Version du serveur : " . ociServerVersion($conn);
   ociLogOff($conn);
?>
```

ociStatementType (PHP 3>= 3.0.5, PHP 4 >= 4.0b1)

Retourne le type de commande oci.

```
string ocistatementtype (resource stmt)
```

ocistatementtype() retourne une des valeurs suivantes :

1. "SELECT"
2. "UPDATE"
3. "DELETE"
4. "INSERT"
5. "CREATE"
6. "DROP"
7. "ALTER"
8. "BEGIN"
9. "DECLARE"
10. "UNKNOWN"

Exemple 1. Exemples

```
<?php
    print "<HTML><PRE>";
    $conn = ociLogon("scott","tiger");
    $sql  = "delete from emp where deptno = 10";
    $stmt = ociparse($conn,$sql);
    if ( ociStatementType($stmt) == "DELETE" ) {
```

```
        die "Vous n'etes pas autorisé à effacer dans cette table.<BR>";
    }
    ociLogoff($conn);
    print "</PRE></HTML>";
?>
```

ociNewCursor (PHP 3>= 3.0.8, PHP 4 >= 4.0b1)

Retourne un nouveau pointeur à utiliser pour lier les pointeurs de références

```
int ocinewcursor (resource conn)
```

ocinewcursor() alloue un nouveau pointeur de commande, pour la connexion *conn*.

Exemple 1. Utiliser un REF CURSOR issue d'une procédure enregistrée.

```
<?php
// supposons que votre procédure stoquée info.output retourne un pointeur
// de curseur dans : data
$conn = ociLogon("scott","tiger");
$curs = ociNewCursor($conn);
$stmt = ociparse($conn,"begin info.output(:data); end;");
ocibindbyname($stmt,"data",&$curs,-1,oci_B_CURSOR);
ociexecute($stmt);
ociexecute($curs);
while (ociFetchInto($curs,&$data)) {
    var_dump($data);
}
ociFreeStatement($curs);
ociFreeCursor($stmt);
ociLogoff($conn);
?>
```

Exemple 2. Utiliser un REF CURSOR issue d'une commande SELECT

```
<?php
print "<HTML><BODY>";
$conn = ociLogon("scott","tiger");
$count_cursor = "CURSOR(select count(empno) num_emps from emp " .
                "where emp.deptno = dept.deptno) as EMPCNT from dept";
$stmt = ociparse($conn,"select deptno,dname,$count_cursor");
ociexecute($stmt);
print "<TABLE BORDER=\"1\">";
print "<TR>";
print "<TH>DEPT NAME</TH>";
print "<TH>DEPT #</TH>";
print "<TH># EMPLOYEES</TH>";
print "</TR>";
while (ociFetchInto($stmt,&$data,oci_ASSOC)) {
    print "<TR>";
    $dname  = $data["DNAME"];
    $deptno = $data["DEPTNO"];
    print "<TD>$dname</TD>";
    print "<TD>$deptno</TD>";
    ociexecute($data[ "EMPCNT" ]);
    while (ociFetchInto($data[ "EMPCNT" ],&$subdata,oci_ASSOC)) {
```

```
        $num_emps = $subdata["NUM_EMPS"];
        print  "<TD>$num_emps</TD>";
    }
    print "</TR>";
}
print "</TABLE>";
print "</BODY></HTML>";
ociFreeStatement($stmt);
ociLogoff($conn);
?>
```

ociFreeStatement (PHP 3>= 3.0.5, PHP 4 >= 4.0b1)

Libère toutes les ressources occupées par une commande.

```
int ocifreestatement (resource stmt)
```

ocifreestatement() retourne TRUE en cas de succès, et FALSE en cas d'échec.

ociFreeCursor (PHP 3>= 3.0.8, PHP 4 >= 4.0b1)

Libère toutes les ressources occupées par un pointeur.

```
boolean ocifreecursor (resource stmt)
```

ocifreecursor() retourne TRUE en cas de succès, et FALSE en cas d'échec.

ociFreeDesc (PHP 4 >= 4.0b4)

Supprime un descripteur de LOB

```
int ocifreedesc (object lob)
```

ocifreedesc() retourne TRUE en cas de succès, et FALSE en cas d'échec.

ociparse (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Analyse une requête.

```
int ociparse (resource conn, string query)
```

ociparse() analyse la requête *query* sur la connexion *conn*, et retourne TRUE si la requête *query* est valide, et FALSE, si ce n'est pas le cas. *query* peut être n'importe quelle requête SQL.

ociError (PHP 3>= 3.0.7, PHP 4 >= 4.0b1)

Retourne la dernière erreur de stmt|conn|global.

```
array ocierror ([int stmt|conn])
```

ocierror() retourne la dernière erreur trouvée. Si l'option *stmt|conn* n'est pas fournie, la dernière erreur rencontrée est retournée. Si aucune erreur n'est trouvée, **ocierror()** retourne FALSE.

ociinternaldebug (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Active ou désactive l'affichage des données de debuggage.

```
void ociinternaldebug (int onoff)
```

ociinternaldebug() active ou désactive l'affichage des informations de debuggage. Pour les afficher, mettez *onoff* à 1, ou sinon mettez *onoff* à 0 pour les cacher.

OCICancel (PHP 3>= 3.0.8, PHP 4 >= 4.0b1)

Bientôt documenté....

```
int ocicancel (resource stmt)
```

ocicancel() détruit les ressources liées au dernier résultat *stmt*. Si vous ne souhaitez plus lire d'informations dans ce résultat, utilisez cette fonction.

ocisetprefetch (PHP 3>= 3.0.12, PHP 4 >= 4.0b1)

Indique le nombre de lignes qui doivent être pré-lues

```
int ocisetprefetch (resource stmt, int rows)
```

ocisetprefetch() indique le nombre des premières lignes qui doivent être pré-lues. La valeur par défaut est de 1.

OCIWriteLobToFile (PHP 4 >= 4.0b4)

Bientôt documenté....

```
void ociwritelobtofile (object lob [, string filename [, int start [, int lenght]]])
```

Bientôt documenté....

OCISaveLobFile (PHP 4 >= 4.0b4)

Bientôt documenté....

```
string ocisavelobfile (object lob)
```

Bientôt documenté....

OCISaveLob (PHP 4 >= 4.0b4)

Bientôt documenté....

```
string ocisavelob (object lob)
```

Bientôt documenté....

OCILoadLob (PHP 4 >= 4.0b4)

Bientôt documenté....

```
string ociloadlob (object lob)
```

Bientôt documenté....

OCIColumnScale (PHP 4 >= 4.0RC1)

Bientôt documenté....

```
int ocicolumnscale (resource stmt, int col)
```

Bientôt documenté....

OCIColumnPrecision (PHP 4 >= 4.0RC1)

Bientôt documenté....

```
int ocicolumnprecision (resource stmt, int col)
```

Bientôt documenté....

OCIColumnTypeRaw (PHP 4 >= 4.0RC1)

Bientôt documenté....

```
mixed ocicolumntyperaw (resource stmt, int col)
```

Bientôt documenté....

OCINewCollection (PHP 4 >= 4.0.6)

Bientôt documenté....

```
string ocinewcollection (int conn, string tdo [, string shema])
```

Bientôt documenté....

OCIFreeCollection (PHP 4 CVS only)

Bientôt documenté....

```
string ocifreecollection (object lob)
```

Bientôt documenté....

OCICollAssign (PHP 4 >= 4.0.6)

Bientôt documenté....

```
string ocicollassign (object collection, object object)
```

Bientôt documenté....

OCICollAssignElem (PHP 4 >= 4.0.6)

Bientôt documenté....

```
string ocicollassignelem (object collection, string ndx, string val)
```

Bientôt documenté....

OCICollGetElem (PHP 4 >= 4.0.6)

Bientôt documenté....

```
string ocicollgetelem (object collection, string ndx)
```

Bientôt documenté....

OCICollMax (PHP 4 >= 4.0.6)

Bientôt documenté....

```
string ocicollmax (object collection)
```

Bientôt documenté....

OCICollSize (PHP 4 >= 4.0.6)

Bientôt documenté....

```
string ocicollsize (object collection)
```

Bientôt documenté....

OCICollTrim (PHP 4 >= 4.0.6)

Bientôt documenté....

```
string ocicolltrim (object collection, int num)
```

Bientôt documenté....

LIX. OpenSSL

> **Avertissement**
>
> Ce module est *EXPERIMENTAL*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

Introduction

Cette extension utilise les fonctions de OpenSSL (http://www.openssl.org/) pour générer et vérifier les signatures, ainsi que pour sceller (chiffrer) et ouvrir (déchiffrer) les données. Vous avez besoin de OpenSSL >= 0.9.5 pour utiliser ce module.

Cette extension supporte aussi la signature et le cryptage des courrier éléctroniques. Il est aussi possible de spécifier des couples clés/certificats d'un grand nombre de cas, qui rendent le code PHP plus facile à lire. Ces fonctionnalités sont disponibles en développement sur CVS, et probablement avec PHP 4.0.6. *ATTENTION : cette extension est encore expérimentale!*

OpenSSL offre de nombreuses fonctionnalités qui ne sont pas encore supportées par ce module. Elle seront ajoutées ultérieurement.

Paramètres clés/certificats

Un bon nombre de fonctions OpenSSL demandent une clé et un certificat comme paramètres. PHP 4.0.5 et plus récent utilisait des clés ou certificats sous forme de ressource, retournée par l'une des fonctions openssl_get_xxx(). Les versions ultérieures utilisent l'une des méthodes suivantes :

- Certificats
  1. Une ressource X.509 retournée par **openssl_x509_read()**
  2. Une chaîne au format `file://path/to/cert.pem`; Le fichier ainsi repéré doit contenir un certificat, encodé au format PEM
  3. Une chaîne contenant le contenu d'un certificat, encodé au format PEM.
- Clés publiques/privée
  1. Une ressource clé, retournée par la fonction **openssl_get_publickey()** ou **openssl_get_privatekey()**
  2. Pour les clés publiques seulement : une ressource X.509
  3. Une chaîne avec le format : `file://path/to/file.pem`. Le fichier doit contenir une clé privé ou un certificat, encodé au format PEM (il peut contenir les deux).
  4. Une chaîne contenant une clé ou un certificat encodé au format PEM
  5. Pour les clés privées, vous pouvez aussi utiliser la syntaxe *array($key, $passphrase)*, où $key représente une clé spécifiée par un fichier ou une représentation textuelle comme cité ci-dessus, et $passphrase représente une chaîne contenant la passe-phrase de cette clé privée.

Vérification de certificats

Lorsque vous appelez une fonction qui va vérifier une signature ou un certificat, le paramètre *cainfo* doit être un tableau contenant les noms d'un dossier et d'un fichier contenant les tiers de confiance. Si un dossier est spécifié, il doit être correct, car **openssl** va l'utiliser.

Constantes/flags PKCS7

Les fonctions S/MIME utilisent des flags qui sont spécifiés par un champs de bits. Les valeurs valides sont :

Tableau 1. Constantes PKCS7

Constante	Description
PKCS7_TEXT	Ajoute le texte plein en clair dans les entêtes du message signé/chiffré. Lors du déchiffrement ou la vérification, il supprime purement et simplement ces données. Si le message chiffré ou signé n'est pas du type MIME, une erreur surviendra.
PKCS7_BINARY	Normalement, le message est converti au format canonique qui utilise effectivement des CR et LF comme fin de ligne, comme demandé dans les spécification de S/MIME. Lorsque cette option est activée, le message ne sera pas converti. Cela sert lorsque vous manipulez des données binaires qui ne sont pas au format MIME.
PKCS7_NOINTERN	Lors de la vérification d'un message, les certificats (s'il y en a) inclus dans le message sont normalement utilisé pour rechercher le certificat de signature. Avec cette option, seul le certificat spécifié par le paramètre `extracerts` de la fonction **openssl_pkcs7_verify()** est utilisé. Les certificats fournis peuvent toujours être utilisé, avec un niveau de confiance réduit.
PKCS7_NOVERIFY	Ne vérifie pas les certificats des signataires d'un message signé.
PKCS7_NOCHAIN	N'enchaine pas les vérifications des signataires de certificats. C'est à dire, n'utilise pas les certificats contenu dans le message.
PKCS7_NOCERTS	Lors de la signature d'un message, le certificat du signataire est normalement inclus. Avec cette option, c'est désactivé. Cela va réduire la taille du message, mais le vérificateur devra avoir une copie local du certificat du signataire (passée au paramètre `extracerts`, avec la fonction **openssl_pkcs7_verify()**).
PKCS7_NOATTR	Normalement, lorsqu'un message est signé, un jeu d'attributs contenant l'heure de signature et l'algorithme symétrique supporté, est inclus dans le message. Avec cette option, il n'est pas inclus.
PKCS7_DETACHED	Lors de la signature d'un message, utilise la signature en texte claire, avec le type MIME "multipart/signed". C'est la valeur par défaut du paramètre `flags` pour la fonction **openssl_pkcs7_sign()**. Si vous annulez cette optoin, le message sera signé de manière opaque, ce qui resiste mieux à la traduction des relais mails (certains serveur mail anciens corrompent les messages), mais empêche la lecture par les client mails qui ne connaissent pas S/MIME.
PKCS7_NOSIGS	Ne vérifie pas les signatures d'une message

Note : Ces constantes ont été ajoutées en PHP 4.0.6.

openssl_error_string (PHP 4 >= 4.0.6)

Retourne le message d'erreur OpenSSL

```
mixed openssl_error_string (void)
```

openssl_error_string() retourne un message d'erreur, sous forme de chaîne de caractères, ou FALSE s'il n'y a pas de message d'erreur.

openssl_error_string() retourne la dernière erreur de la librairie OpenSSL. Les messages d'erreurs sont empilés, et **openssl_error_string()** doit être appelé plusieurs fois pour afficher toutes les erreurs.

Les paramètres et le type de retour de cette fonction risquent d'evoluer d'ici à la prochaine version de PHP.

Exemple 1. Exemple avec openssl_error_string()

```
<?php
// Imaginons que vous avez appelé une fonction qui a émis une erreur
while($msg = openssl_error_string)
    echo $msg . "<br>";
?>
```

Note : Ces constantes ont été ajoutées en PHP 4.0.6.

openssl_free_key (PHP 4 >= 4.0.4)

Libère les ressources

```
void openssl_free_key (resource key_identifier)
```

openssl_free_key() libère les ressources associées à *key_identifier*.

openssl_get_privatekey (PHP 4 >= 4.0.4)

Prépare une clé privée au format PEM

```
resource openssl_get_privatekey (mixed key [, string passphrase])
```

openssl_get_privatekey() retourne un identifiant de clé positif, ou FALSE en cas d'erreur.

openssl_get_privatekey() analyse la clé privée *key*, au format PEM, et la prépare pour à être utilisée par d'autres fonctions. Le paramètre optionnel *passphrase* doit être utilisé si la clé est chiffrée (protégée par un mot de passe).

openssl_get_publickey (PHP 4 >= 4.0.4)

Extrait une clé publique d'un certificat

```
resource openssl_get_publickey (mixed certificate)
```

openssl_get_publickey() retourne un identifiant de clé positif, ou FALSE en cas d'erreur.

openssl_get_publickey() extrait la clé publique du certificat *certificate* (format X.509), et la prépare à être utilisée ultérieurement.

openssl_open (PHP 4 >= 4.0.4)

Ouvre des données scellées

```
bool openssl_open (string sealed_data, string open_data, string env_key, mixed
priv_key_id)
```

openssl_open() TRUE en cas de succès, et FALSE sinon. En cas de succès, les données déchiffrées sont placées dans *open_data*.

openssl_open() ouvre (déchiffre) les données *sealed_data* en utilisant la clé prviée *priv_key_id* et la clé d'enveloppe *env_key* et remplis *open_data* avec les données déchiffrées. La clé d'enveloppe est générée lorsque les données sont scellées, et ne peut être utilisée qu'avec la clé privée spécifique. Reportez vous à **openssl_seal()** pour plus d'informations.

Exemple 1. Exemple avec openssl_open()

```
<?php
// On suppose que $sealed et $env_key contiennent les données scellées
// et la clé d'enveloppe, fournies par l'expéditeur
// lecture de la clé privée dans un fichier
$fp = fopen("/src/openssl-0.9.6/demos/sign/key.pem", "r");
$priv_key = fread($fp, 8192);
fclose($fp);
$pkeyid = openssl_get_privatekey($priv_key);
// déchiffrage des données : elles sont placées dans $open
if (openssl_open($sealed, $open, $env_key, $pkeyid))
    echo "Voici les données déchiffrées : ", $open;
else
    echo "Impossible de déchiffrer les données";
// libération des ressources
openssl_free_key($pkeyid);
?>
```

Voir aussi **openssl_seal()**.

openssl_seal (PHP 4 >= 4.0.4)

Scelle des données

```
int openssl_seal (string data, string sealed_data, array env_keys, array pub_key_ids)
```

openssl_seal() retourne la longueur des données scellées en cas de succès, et FALSE sinon. En cas de succès, les données scellées sont placées dans le paramètre *sealed_data*, et les clés d'enveloppe dans *env_keys*.

openssl_seal() scelle (chiffre) les données *data* en utilisant l'algorithme RC4 avec une clé secrètre générée aléatoirement. La clé est chiffrée avec chaque clé publique associée à *pub_key_ids* et chaque clé ainsi encryptée est retournée dans *env_keys*. Cela signifique que vous pouvez envoyez des données scellées à plusieurs destinataires (en supposant que chacun ait recu la clé publique). Chaque destinataire doit recevoir les données encryptées et la clé d'enveloppe, qui a été encryptée avec la clé publique du destinataire.

Exemple 1. Exemple avec openssl_seal()

```
<?php
// On suppose que $data contient les données à sceller
// lecture de la clé publique pour chaque destinataire
$fp = fopen("/src/openssl-0.9.6/demos/maurice/cert.pem", "r");
$cert = fread($fp, 8192);
fclose($fp);
$pk1 = openssl_get_publickey($cert);
// pour le deuxième destinataire
$fp = fopen("/src/openssl-0.9.6/demos/sign/cert.pem", "r");
$cert = fread($fp, 8192);
fclose($fp);
$pk2 = openssl_get_publickey($cert);
// scelle le message : seuls, les possessuers de $pk1 et $pk2 peuvent déchiffrer
// le message $sealed avec les clés $ekeys[0] et $ekeys[1] (respectivement).
openssl_seal($data, $sealed, $ekeys, array($pk1,$pk2));
// libère les clés de la mémoire
openssl_free_key($pk1);
openssl_free_key($pk2);
?>
```

Voir aussi **openssl_open()**.

openssl_sign (PHP 4 >= 4.0.4)

Signe les données

```
bool openssl_sign (string data, string signature, mixed priv_key_id)
```

openssl_sign() retourne `TRUE` en cas de succès, et `FALSE` sinon. En cas de succès, la signature est placée dans *`signature`*.

openssl_sign() calcule la signature des données *`data`* en utilisant l'algorithme SHA1 (hashing) suivi du chiffrage avec la clé privée *`priv_key_id`*. Notez que les données elles-mêmes ne sont pas chiffrées.

Exemple 1. Exemple avec openssl_sign()

```
<?php
// On suppose que $data contient les données à signer
// lecture de la clé publique pour chaque destinataire
$fp = fopen("/src/openssl-0.9.6/demos/sign/key.pem", "r");
$priv_key = fread($fp, 8192);
fclose($fp);
$pkeyid = openssl_get_privatekey($priv_key);
// calcule de la signature
openssl_sign($data, $signature, $pkeyid);
// libère les clés de la mémoire
openssl_free_key($pkeyid);
?>
```

Voir aussi **openssl_verify()**.

openssl_verify (PHP 4 >= 4.0.4)

Vérifie une signature

```
int openssl_verify (string data, string signature, resource pub_key_id)
```

openssl_verify() retourne 1 si la signature est correcte, 0 si la signature est incorrecte, et -1 en cas d'erreur.

openssl_verify() vérifie que la signature *signature* est correcte pour les données *data*, et avecd la clé publique *pub_key_id*. Cette clé doit être la clé publique correspondant à la clé privée utilisée lors de la signature.

Exemple 1. Exemple avec openssl_verify()

```
<?php
// On suppose que $data et $signature contiennent les données à signer et
// la signature
// lecture de la clé publique depuis le certificat
$fp = fopen("/src/openssl-0.9.6/demos/sign/cert.pem", "r");
$cert = fread($fp, 8192);
fclose($fp);
$pubkeyid = openssl_get_publickey($cert);
// indique si la signature est correcte
$ok = openssl_verify($data, $signature, $pubkeyid);
if ($ok == 1)
    echo "Signature valide";
elseif ($ok == 0)
    echo "Signature erronnée";
else
    echo "Erreur de vérification de la signature";
// libère les clés de la mémoire
openssl_free_key($pubkeyid);
?>
```

Voir aussi **openssl_sign()**.

openssl_pkcs7_decrypt (PHP 4 >= 4.0.6)

Déchiffre un message S/MIME

```
bool openssl_pkcs7_decrypt (string infilename, string outfilename, mixed recipcert, mixed
recipkey)
```

openssl_pkcs7_decrypt() déchiffre le message S/MIME contenu dans le fichier *infilename*, en utilisant le certificat et la clé privée spécifiés par *recipcert* et *recipkey*. Le message déchiffré sera écrit dans le fichier *outfilename*.

Les paramètres et le type de retour de cette fonction risquent d'evoluer d'ici à la prochaine version de PHP.

Exemple 1. Exemple avec openssl_pkcs7_decrypt()

```
<?php
// $cert et $key contiennent vos certificats et clés privés
// On suppose aussi que le message vous est destiné
$infilename = "encrypted.msg";  // Le message chiffré
$outfilename = "decrypted.msg"; // Assurez vous de bien pouvoir écrire dans ce fichier
if (openssl_pkcs7_decrypt($infilename, $outfilename, $cert, $key))
    echo "déchiffré!";
else
    echo "impossible de déchiffrer!";
```

```
?>
```

Note : Ces constantes ont été ajoutées en PHP 4.0.6.

openssl_pkcs7_encrypt (PHP 4 >= 4.0.6)

Chiffre un message S/MIME

```
bool openssl_pkcs7_encrypt (string infilename, string outfilename, mixed recipcerts, array
headers [, long flags])
```

openssl_pkcs7_encrypt() prend le contenu du fichier *infilename* et le chiffre en utilisant un chiffrement RC2 à 40-bit, de manière à ce que le message ne puisse être lu que par le possesseur de *recipcerts*, qui peut être un certificat X.509, ou un tableau de certificats X.509. *headers* est un tableau d'entête qui sera ajouté en tête de message, une fois que les données auront été chiffrées. *flags* peut être utilisé pour spécifier des options qui affecteront le chiffrement (voir les constantes PKCS7). *headers* peut être un tableau associatif, dont les clés sont les noms d'entête, ou bien un tableau indexé dont chaque ligne contient une ligne d'entête complète.

Les paramètres et le type de retour de cette fonction risquent d'evoluer d'ici à la prochaine version de PHP.

Exemple 1. Exemple avec openssl_pkcs7_encrypt()

```
<?php
// le message que vous souhaitez chiffrer et envoyer à votre agent secret
// en mission commandée, appelé "nighthawk". Vous avez son certificat
// dans le fichir "nighthawk.pem"
$data = <<«EOD
Nighthawk,
Top secret, uniquement vous votre lecture!
L'ennemi approche! Rendez vous au café à 8h30,
pour votre faux passeport.
HQ
EOD;
// sauvez le message dans un fichier
$fp = fopen("msg.txt", "w");
fwrite($fp, $data);
fclose($fp);
// chiffrez le
if (openssl_pkcs7_encrypt("msg.txt", "enc.txt", "nighthawk.pem",
    array("To" => "nighthawk@agent.com", // keyed syntax
          "From: HQ <hq@cia.com>", // indexed syntax
          "Subject" => "Eyes only")))
{
    // message chiffré : envoyez le!
    exec(ini_get("sendmail_path") . " < enc.txt");
}
?>
```

Note : Ces constantes ont été ajoutées en PHP 4.0.6.

openssl_pkcs7_sign (PHP 4 >= 4.0.6)

signe un message S/MIME

```
bool openssl_pkcs7_sign (string infilename, string outfilename, mixed signcert, mixed
privkey, array headers [, long flags [, string extracertsfilename]])
```

openssl_pkcs7_sign() prend le contenu du fichier *infilename* et le signe en utilisant le certificat et la clé privée contenus dans les arguments *signcert* et *privkey*.

headers est un tableau d'entête qui sera ajouté au données chiffrées (voir la fonction **openssl_pkcs7_encrypt()** pour plus de détails sur le format du paramètre).

flags sert à modifier le message final. Voyez les constantes PKCS7. Par défaut, la valeur est : PKCS7_DETACHED.

extracerts spécifi le nom du fichier contenant un ensemble de certificat supplémentaires à inclure dans la signature, qui pourront aider le destinataire à vérifier les données que vous utilisez.

Les paramètres et le type de retour de cette fonction risquent d'evoluer d'ici à la prochaine version de PHP.

Exemple 1. openssl_pkcs7_sign() example

```
<?php
// le message que vous voulez signer, afin que le destinataire soit sÛr qu'il
// vient bien de vous
$data = <«EOD
Tu peux dépenser jusqu'à 10000 euros en note de frais.
Ton boss
HQ
EOD;
// sauvez le message dans un fichier
$fp = fopen("msg.txt", "w");
fwrite($fp, $data);
fclose($fp);
// chiffrez le
if (openssl_pkcs7_sign("msg.txt", "signed.txt", "mycert.pem",
    array("mycert.pem", "mypassphrase"),
    array("To" => "joes@sales.com", // keyed syntax
          "From: HQ <ceo@sales.com>", // indexed syntax
          "Subject" => "Eyes only"))
{
    // message signed - send it!
    exec(ini_get("sendmail_path") . " < signed.txt");
}
?>
```

Note : Ces constantes ont été ajoutées en PHP 4.0.6.

openssl_pkcs7_verify (PHP 4 >= 4.0.6)

Vérifie la signature d'un message S/MIME

```
bool openssl_pkcs7_verify (string filename, int flags [, string outfilename [, array cainfo
[, string extracerts]]])
```

openssl_pkcs7_verify() lit le message S/MIME contenu dans le fichier *filename* et examine la signature digitale. **openssl_pkcs7_verify()** retourne TRUE si la signature est vérifiée, et FALSE sinon (le message a été modifié, ou bien le

certificat de signature est invalide). **openssl_pkcs7_verify()** retourne -1 en cas d'erreur de vérification (la vérification s'est mal déroulée, aucune conclusion possible).

flags sert à modifier le message final. Voyez les constantes PKCS7. Par défaut, la valeur est : PKCS7_DETACHED.

Si le paramètre *outfilename* est spécifié, il doit être une chaîne contenant le nom d'un fichier qui contient le certificat du signataire, au format PEM.

Si le paramètre *cainfo* est spécifié, il doit contenir les informations sur les tiers de certificats de confiance utilisé lors de la vérification. Voyez vérification des certificats pour plus de détails.

Si le paramètre *extracerts* est spécifié, il doit représenter le nom d'un fichier contenant un ensemble de certificat utilisé comme certificats de peu de confiance.

Les paramètres et le type de retour de cette fonction risquent d'evoluer d'ici à la prochaine version de PHP.

Note : Ces constantes ont été ajoutées en PHP 4.0.6.

openssl_x509_checkpurpose (PHP 4 >= 4.0.6)

Vérifie l'usage d'un certificat

```
bool openssl_x509_checkpurpose (mixed x509cert, int purpose, array cainfo [, string untrustedfile])
```

openssl_x509_checkpurpose() TRUE si le certificat peut être utilisé pour un but particulier, FALSE s'il ne le peut pas, et -1 en cas d'erreur.

openssl_x509_checkpurpose() examine le certificat specifié par *x509cert*, pour voir s'il peut être utilisé pour une opération particulière *purpose*.

cainfo doit être un tableau de dossiers/fichiers de CA de confiance comme décrit dans la Vérification des certificats.

untrustedfile, si spécifié, est le nom d'un fichier au format PEM contenant les certificats qui pourront aider lors de la vérification du certificat, même si une confiance limitée doit leur être portée.

Les paramètres et le type de retour de cette fonction risquent d'evoluer d'ici à la prochaine version de PHP.

Tableau 1. Utilisations de openssl_x509_checkpurpose()

Constante	Description
X509_PURPOSE_SSL_CLIENT	Est ce que le certificat peut être utilisé avec le client d'une connexion SSL?
X509_PURPOSE_SSL_SERVER	Est ce que le certificat peut être utilisé avec le serveur d'une connexion SSL?
X509_PURPOSE_NS_SSL_SERVER	Est ce que le certificat peut être utilisé avec un serveur Netscape d'une connexion SSL?
X509_PURPOSE_SMIME_SIGN	Est ce que le certificat peut être utilisé pour signer des courrier à la norme S/MIME?
X509_PURPOSE_SMIME_ENCRYPT	Est-ce que le certificat peut être utilisé pour chiffrer un courrier au format S/MIME?
X509_PURPOSE_CRL_SIGN	Est-ce que le certificat peut être utilisé pour chiffrer une liste de revocation de certificat? (CRL)?
X509_PURPOSE_ANY	Est-ce que le certificat peut être utilisé pour n'importe lequel de ces cas?

Ces options ne sont pas des champs de bits : vous ne pouvez en passer qu'une seule à la fois.

Note : Ces constantes ont été ajoutées en PHP 4.0.6.

openssl_x509_free (PHP 4 >= 4.0.6)

Libère les ressources prises par un certificat

```
void openssl_x509_free (resource x509cert)
```

openssl_x509_free() libère les ressources prises par le certificat *x509cert*

Note : Ces constantes ont été ajoutées en PHP 4.0.6.

openssl_x509_parse (PHP 4 >= 4.0.6)

Analyse un certificat X509.

```
array openssl_x509_parse (mixed x509cert [, bool shortnames])
```

openssl_x509_parse() analyse le certificat X509 *x509cert*, et retourne les informations contenues dedans, y compris le sujet (subject), nom (name), émetteur (issuer name), dates de début et de fin (valid from date et valid to date), etc... *shortnames* contrôle l'indexation des données dans le tableau : si *shortnames* vaut TRUE (valeur par défaut), alors les champs seront indexés avec la forme courte des noms, sinon, les noms longs seront utilisés. (par exemple, CN est le nom court de commonName).

La structure des données retournées est (délibérément) non documentée, car elle se sujette à des changements probables.

Note : Ces constantes ont été ajoutées en PHP 4.0.6.

openssl_x509_read (PHP 4 >= 4.0.6)

Analyse un certificat X.509 et retourne une ressource

```
resource openssl_x509_read (mixed x509certdata)
```

openssl_x509_read() analyse le certificat *x509certdata* et retourne un identifiant de ressource.

Note : Ces constantes ont été ajoutées en PHP 4.0.6.

LX. Oracle

Ora_Bind (PHP 3, PHP 4 >= 4.0b1)

Lie une variable PHP à un paramètre Oracle.

```
int ora_bind (resource cursor, string PHP variable name, string SQL parameter name, int
length [, int type])
```

ora_bind() retourne TRUE si la liaison a pu se faire, et sinon FALSE. Les erreurs sont accessibles avec les fonctions **ora_error()** et **ora_errorcode()**.

Cette fonction lie une variable PHP avec un paramètre SQL. Le paramètre SQL doit être de la forme ":name". Avec l'option, vous pouvez choisir si le paramètre SQL est de type entrée/sortie (0, valeur par défaut), entrée seulement (1) ou sortie seulement (2). Comme dans PHP 3.0.1, vous pouvez respectivement utiliser les constantes ORA_BIND_INOUT, ORA_BIND_IN eet ORA_BIND_OUT plutôt que des nombres.

ora_bind() doit être appelée après la fonction **ora_parse()** et avant **ora_exec()**. Les valeurs d'entrées peuvent alors être fournies par assignation des variables PHP. Après la fonction **ora_exec()** les variables liées contiennent les valeurs de sortie, si elles sont disponibles. Par exemple :

```
<?php
ora_parse($curs, "declare tmp INTEGER; begin tmp := :in; :out := tmp; :x := 7.77; end;");
ora_bind($curs, "result", ":x", $len, 2);
ora_bind($curs, "input", ":in", 5, 1);
ora_bind($curs, "output", ":out", 5, 2);
$input = 765;
ora_exec($curs);
echo "Résultat: $result<BR>Sortie: $output<BR>Entrée: $input";
?>
```

Ora_Close (PHP 3, PHP 4 >= 4.0b1)

Ferme un pointeur Oracle.

```
int ora_close (resource cursor)
```

ora_close() retourne TRUE si la fermeture a bien eu lieu, et FALSE sinon. Les erreurs sont accessibles avec les fonctions **ora_error()** et **ora_errorcode()**.

ora_close() termine les pointeurs ouverts avec la fonction **ora_open()**.

Ora_ColumnName (PHP 3, PHP 4 >= 4.0b1)

Retourne le nom de la colonne de résultat.

```
string ora_columnname (resource cursor, int column)
```

ora_columnname() retourne le nom du champs *column* du pointeur *cursor*. Le nom retourné sera en majuscule.

Ora_ColumnSize (PHP 3, PHP 4 >= 4.0b1)

Lit la taille d'une colonne

```
int ora_columnsize (resource cursor, int column)
```

ora_columnsize() retourne la taille de la colonne *column* dans le résultat *cursor*.

Ora_ColumnType (PHP 3, PHP 4 >= 4.0b1)

Retourne le type de la colonne de résultat.

```
string ora_columntype (resource cursor, int column)
```

ora_columntype() retourne le type de la colonne *column* du résultat *cursor*. Le type retourné prendra une des valeurs suivantes :

```
"VARCHAR2"
"VARCHAR"
"CHAR"
"NUMBER"
"LONG"
"LONG RAW"
"ROWID"
"DATE"
"CURSOR"
```

Ora_Commit (PHP 3, PHP 4 >= 4.0b1)

Valide une transaction Oracle.

```
int ora_commit (resource conn)
```

ora_commit() retourne TRUE si la validation a bien eu lieu, et FALSE sinon. Les erreurs sont accessibles avec les fonctions **ora_error()** et **ora_errorcode()**.

ora_commit() valide les transactions Oracle. Une transaction est définie par toutes les requêtes effectuées sur la connexion conn depuis la dernière validation ou annulation (avec auto-validation inactivée) ou depuis l'établissement de la connexion.

Ora_CommitOff (PHP 3, PHP 4 >= 4.0b1)

Inactive la validation automatique.

```
int ora_commitoff (resource conn)
```

ora_commitoff() retourne TRUE si la désactivation a bien eu lieu, et FALSE sinon. Les erreurs sont accessibles avec les fonctions **ora_error()** et **ora_errorcode()**.

ora_commitoff() inactive la validation automatique après chaque **ora_exec()**.

Ora_CommitOn (PHP 3, PHP 4 >= 4.0b1)

Active la validation automatique.

```
int ora_commiton (resource conn)
```

ora_commiton() active la validation automatique après chaque **ora_exec()**.

ora_commiton() retourne TRUE si l'activation a bien eu lieu, et FALSE sinon. Les erreurs sont accessibles avec les fonctions **ora_error()** et **ora_errorcode()**.

Ora_Do (PHP 3, PHP 4 >= 4.0b1)

Analyse, exécute et lit une requête

```
int ora_do (resource conn, string query)
```

ora_do() est une combinaison de **ora_parse()**, **ora_exec()** et **ora_fetch()**. Elle va analyser la requête, l'éxécuter et lire la première ligne du résultat.

ora_do() retourne TRUE en cas de succès, et FALSE sinon. Le détails sur les erreurs est accessible avec les fonctions **ora_error()** et **ora_errorcode()**.

Voir aussi **ora_parse()**,**ora_exec()**, et **ora_fetch()**.

Ora_Error (PHP 3, PHP 4 >= 4.0b1)

Retourne le message d'erreur Oracle.

```
string ora_error (resource cursor_or_connection)
```

ora_error() retourne un messages d'erreur de la forme *XXX-NNNNN* avec *XXX* qui est l'origine de l'erreur, et *NNNNN* qui identifie le message d'erreur.

Note : Le support des connexions a été ajouté dans PHP 3.0.4.

Avec les versions UNIX d'Oracle, vous pouvez avoir des messages d'erreur tels que : `$ oerr ora 00001 00001, 00000, "unique constraint (%s.%s) violated" // *Cause: An update or insert statement attempted to insert a duplicate key // For Trusted ORACLE configured in DBMS MAC mode, you may see // this message if a duplicate entry exists at a different level. // *Action: Either remove the unique restriction or do not insert the key .`

Ora_ErrorCode (PHP 3, PHP 4 >= 4.0b1)

Retourne le code d'erreur Oracle.

```
int ora_errorcode (resource cursor_or_connection)
```

ora_errorcode() retourne le code d'erreur numérique de la dernière commande exécuté sur la connexion ou le pointeur fourni en paramètre.

Note : Les identifiants de connexion ne sont acceptés qu'à partir de la version 3.0.4.

Ora_Exec (PHP 3, PHP 4 >= 4.0b1)

Exécute une commande analysée sur un pointeur Oracle.

```
bool ora_exec (resource cursor)
```

ora_exec() retourne TRUE en cas de succès, et FALSE en cas d'erreur. L'erreur générée sera alors accessible avec les fonctions **ora_error()** et **ora_errorcode()**.

Ora_Fetch (PHP 3, PHP 4 >= 4.0b1)

Retourne une ligne de résultat.

```
int ora_fetch (resource cursor)
```

ora_fetch() retourne TRUE (une ligne a été lue) ou FALSE (plus de lignes à lire ou erreur). Si une erreur survient, sa valeur sera disponible dans les fonctions **ora_error()** et **ora_errorcode()**.

Lit une ligne de données sur le pointeur cursor.

Voir aussi **ora_parse()**,**ora_exec()**, et **ora_do()**.

Ora_Fetch_Into (PHP 3, PHP 4 >= 4.0b1)

Lit une ligne dans un tableau

```
int ora_fetch_into (resource cursor, array result, int [flags])
```

ora_fetch_into() lit la ligne courante du résultat *`cursor`* dans le tableau *`result`*.

Exemple 1. Exemple ora_fetch_into()

```
<?php
array($results);
ora_fetch_into($cursor, &$results);
echo $results[0];
echo $results[1];
?>
```

Notez que vous devez passer le tableau par référence;

Voir aussi **ora_parse()**,**ora_exec()**, **ora_fetch()**, et **ora_do()**.

Ora_GetColumn (PHP 3, PHP 4 >= 4.0b1)

Retourne une donnée d'une ligne lue.

```
mixed ora_getcolumn (resource cursor, mixed column)
```

ora_getcolumn() retourne la valeur de la colonne. Si une erreur survient, FALSE est retourné et **ora_errorcode()** aura une valeur non nulle. Notez, qu'un test à FALSE, avec cette fonction peut être TRUE, même sans erreur : en effet, la fonction peut retourner des valeurs telles que (résultat NULL, chaînes vides, nombre 0, la chaîne "0").

Ora_Logoff (PHP 3, PHP 4 >= 4.0b1)

Ferme une connexion Oracle.

```
int ora_logoff (resource connection)
```

ora_logoff() retourne TRUE si la fermeture a bien eu lieu, et FALSE sinon. Les erreurs sont accessibles avec les fonctions **ora_error()** et **ora_errorcode()**.

ora_logoff() déconnecte l'utilisateur, et se déconnecte.

Voir aussi **ora_logon()**.

Ora_Logon (PHP 3, PHP 4 >= 4.0b1)

Ouvre une connexion Oracle.

```
resource ora_logon (string user, string password)
```

ora_logon() établit une connexion entre PHP et un serveur Oracle avec les noms d'utilisateur user et le mot de passe password.

Les connexions peut être faîtes avec SQL*Net en fournissant le nom TNS de la manière suivante :

```
<?php
$conn = ora_logon("user@TNSNAME", "pass");
?>
```

Si vous avez des données qui ne sont pas ASCII, vous devriez vérifier que la variable NLS_LANG a été correctement configuré dans votre environnement. Pour les modules de serveur, vous devrez la configurer dans l'environnement d'exécution du serveur avant de le lancer.

ora_logon() retourne un index de connexion, en cas de succès, ou FALSE en cas d'échec. Les erreurs sont accessibles avec les fonctions **ora_error()** et **ora_errorcode()**.

Ora_pLogon (PHP 3, PHP 4 >= 4.0b1)

Ouvre une connexion persistante à Oracle

```
resource ora_plogon (string user, string password)
```

ora_plogon() établit une connexion persistante à un serveur Oracle, avec l'utilisateur *user* et le mot de passe *password*.

Voir aussi **ora_logon()**.

Ora_Numcols (PHP 3, PHP 4 >= 4.0b1)

Retourne le nombre de colonnes

```
int ora_numcols (resource cursor_ind)
```

ora_numcols() retourne le nombre de colonne dans le résultat *cursor_ind*. Cette valeur n'est significative qu'après une requête parse/exec/fetch.

Voir aussi **ora_parse()**,**ora_exec()**, **ora_fetch()**, et **ora_do()**.

Ora_Numrows (PHP 3, PHP 4 >= 4.0b1)

Retourne le nombre de colonnes

```
int ora_numrows (resource cursor_ind)
```

ora_numrows() retourne le nombre de colonnes dans le résultat *cursor_ind*.

Ora_Open (PHP 3, PHP 4 >= 4.0b1)

Ouvre un pointeur Oracle.

```
resource ora_open (resource connection)
```

ora_open() ouvre un pointeur Oracle sur la connexion.

ora_open() retourne TRUE si la fermeture a bien eu lieu, et FALSE sinon. Les erreurs sont accessibles avec les fonctions **ora_error()** et **ora_errorcode()**.

Ora_Parse (PHP 3, PHP 4 >= 4.0b1)

Analyse une requête SQL.

```
int ora_parse (resource cursor_ind, string sql_statement, int defer)
```

ora_parse() analyse une requête SQL ou un bloc PL/SQL et l'associe avec le pointeur cursor_ind. Retourne 0 en cas de succès, et -1 en cas d'erreur.

ora_parse() retourne 0 en cas de succès, et -1 en cas d'erreur.

Voir aussi **ora_exec()**, **ora_fetch()** et **ora_do()**.

Ora_Rollback (PHP 3, PHP 4 >= 4.0b1)

Annule une transaction.

```
int ora_rollback (resource connection)
```

ora_rollback() annule une transaction Oracle. (Voir **ora_commit()** pour la définition d'une transaction).

ora_rollback() retourne TRUE si la fermeture a bien eu lieu, et FALSE sinon. Les erreurs sont accessibles avec les fonctions **ora_error()** et **ora_errorcode()**.

LXI. Ovrimos SQL

Ovrimos SQL Server est une base de données relationnelle client/serveur et transactionelle, combinée avec des fonctionnalités web, et des transactions rapides.

Ovrimos SQL Server est disponible à www.ovrimos.com (http://www.ovrimos.com/). Pour activer le support ovrimos de PHP, il suffit de compiler PHP avec l'option '--with-ovrimos' du script de configuration. Vous devrez aussi installer la librairie sqlcli disponbile avec la distribution Ovrimos SQL Server.

Exemple 1. Connection au serveur Ovrimos SQL Server et selectionn d'une table système

```
<?php
$conn = ovrimos_connect ("server.domain.com", "8001", "admin", "password");
if ($conn != 0) {
    echo ("Connection établie!");
    $res = ovrimos_exec ($conn, "select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Requête effectuée!";
        ovrimos_result_all ($res);
        ovrimos_free_result ($res);
    }
    ovrimos_close($conn);
}
?>
```

Cet exemple effectue une connexion réussie.

ovrimos_connect (PHP 4 >= 4.0.3)

Connexion à un serveur

```
int ovrimos_connect (string host, string db, string user, string password)
```

ovrimos_connect() sert à se connecter à un serveur Ovrimos.

ovrimos_connect() retourne un identifiant de connexion, supérieur à 0, ou 0 en cas d'échec.*host* est l'adresse IP de l'hôte Ovrimos, et *db* est soit le nom d'une base de donnes, soit une chaîne contenant le numéro de port.

Exemple 1. Exemple avec ovrimos_connect()

```
<?php
$conn = ovrimos_connect ("server.domain.com", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection établie!";
    $res=ovrimos_exec ($conn, "select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Requête effectuée!";
        ovrimos_result_all ($res);
        ovrimos_free_result ($res);
    }
    ovrimos_close ($conn);
}
?>
```

L'exemple ci dessus montre comment se connecter à une base de donnée et afficher le contenu d'une table.

ovrimos_close (PHP 4 >= 4.0.3)

Ferme une connexion

```
void ovrimos_close (int connection)
```

ovrimos_close() sert à ferme une connexion à un serveur Ovrimos.

ovrimos_close() ferme la connexion au serveur Ovrimos. Toutes les transactions non validées sont annulées.

ovrimos_close_all (PHP 4 >= 4.0.3)

Ferme toutes les connexions aux serveur ovrimos

```
void ovrimos_close_all (void)
```

ovrimos_close_all() sert à ferme toutes les connexions.

ovrimos_close_all() ferme toute les connexions à Ovrimos. Toutes les transactions non validées sont annulées.

ovrimos_longreadlen (PHP 4 >= 4.0.3)

Indique la taille des données à lire dans une colonne de grande taille

```
int ovrimos_longreadlen (int result_id, int length)
```

ovrimos_longreadlen() sert à lire la taille des données qui sera lues lors de l'accès une colonne de grande taille.

ovrimos_longreadlen() indique le nombre d'octets qui seront lus dans une colonne de grande taille (long varchar et long varbinary). Par défaut, 0. Indépendemment du fait que **ovrimos_longreadlen()** requiert *result_id*, actuellement **ovrimos_longreadlen()** affecte ce paramètre pour tous les résultats. Retourne vrai.

ovrimos_prepare (PHP 4 >= 4.0.3)

Prépare une requête SQL

```
int ovrimos_prepare (int connection_id, string query)
```

ovrimos_prepare() sert à préparer une requête SQL.

ovrimos_prepare() prépare une requête SQL et retourne un identifiant de résultat result_id (ou FALSE en cas d'échec).

Exemple 1. Connexion à un serveur Ovrimos SQL Server et préparation d'une requête

```
<?php
$conn=ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn!=0) {
    echo "Connection établie!";
    $res=ovrimos_prepare ($conn, "select table_id, table_name
                                  from sys.tables where table_id=1");
    if ($res != 0) {
        echo "Préparation faite!";
        if (ovrimos_execute ($res)) {
            echo "Exécution réussie!\n";
            ovrimos_result_all ($res);
        } else {
            echo "Exécution manquée!";
        }
        ovrimos_free_result ($res);
    } else {
        echo "Préparation manquée!\n";
    }
    ovrimos_close ($conn);
}
?>
```

Cet exemple montre comment se connecter à un serveur Ovrimos SQL Server, comment préparer une requête SQL et l'exécuter.

ovrimos_execute (PHP 4 >= 4.0.3)

Exécute une requête préparée

```
int ovrimos_execute (int result_id [, array parameters_array])
```

ovrimos_execute() sert à exécuter une requête SQL.

ovrimos_execute() exécute une requête préparée. Retourne TRUE ou FALSE. Si la requête préparée contient des paramètres (des points d'interrogations dans la requête), un nombre correct de paramètre doit être passé dans le tableau *parameters_array*. Notez que **ovrimos_execute()** ne suit pas les conventions PHP qui placent les noms des paramètres entre crochets. L'auteur n'a pas pu s'y faire.

ovrimos_cursor (PHP 4 >= 4.0.3)

Retourne le nom du curseur

```
int ovrimos_cursor (int result_id)
```

ovrimos_cursor() sert à lire le nom du curseur

ovrimos_cursor() retourne le nom du curseur. Pratique, lorsqu'on veut faire des modifications ou des effacements avec des curseurs déjà positionnés.

ovrimos_exec (PHP 4 >= 4.0.3)

Exécute une requÍte SQL

```
int ovrimos_exec (int connection_id, string query)
```

ovrimos_exec() sert à exécuter une requête SQL.

ovrimos_exec() exécute une requête SQL (selection ou modification), et retourne un identifiant de résultat result_id (ou bien FALSE, en cas d'échec). Evidemment, la requête SQL ne doit pas contenir de paramètres.

ovrimos_fetch_into (PHP 4 >= 4.0.3)

Lit une ligne dans un résultat

```
int ovrimos_fetch_into (int result_id, array result_array [, string how [, int
rownumber]])
```

ovrimos_fetch_into() lit une ligne dans un résultat SQL.

ovrimos_fetch_into() lit une ligne dans le résultat *result_id*, qui doit être passé en référence. La ligne qui sera lue est déterminée par les deux paramètres *how* et *rownumber*. *how* peut prendre les valeurs de 'Next' (suivant, valeur par défaut), 'Prev' (précédent), 'First' (premier), 'Last' (dernier), 'Absolute' (position absolue). La casse de *how* n'est pas prise en compte. *rownumber* est optionne, sauf dans le cas d"Absolute'. Retourne TRUE ou FALSE.

Exemple 1. Lit un exemple

```
<?php
$conn=ovrimos_connect ("neptune", "8001", "admin", "password");
if ($conn!=0) {
    echo "Connection établie!";
    $res=ovrimos_exec ($conn,"SELECT table_id, table_name FROM sys.tables");
    if ($res != 0) {
        echo "Requête effectuée!";
        if (ovrimos_fetch_into ($res, &$row)) {
            list ($table_id, $table_name) = $row;
            echo "table_id=".$table_id.", table_name=".$table_name."\n";
            if (ovrimos_fetch_into ($res, &$row)) {
                list ($table_id, $table_name) = $row;
                echo "table_id=".$table_id.", table_name=".$table_name."\n";
            } else {
                echo "Next: erreur\n";
            }
        } else {
            echo "First: erreur\n";
        }
        ovrimos_free_result ($res);
```

```
    }
    ovrimos_close ($conn);
}
?>
```

Cet exemple lis une ligne.

ovrimos_fetch_row (PHP 4 >= 4.0.3)

Lit une ligne dans un résultat

```
int ovrimos_fetch_row (int result_id [, int how [, int row_number]])
```

ovrimos_fetch_row() lit une ligne dans un résultat SQL.

ovrimos_fetch_row() lit une ligne dans un résultat. Les colonnes doivent être lues par un autre appel. Retourne TRUE ou FALSE.

Exemple 1. Exemple de lecture de ligne

```
<?php
$conn = ovrimos_connect ("remote.host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection établie!";
    $res=ovrimos_exec ($conn, "select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Requête effectuée!";
        if (ovrimos_fetch_row ($res, "First")) {
            $table_id = ovrimos_result ($res, 1);
            $table_name = ovrimos_result ($res, 2);
            echo "table_id=".$table_id.", table_name=".$table_name."\n";
            if (ovrimos_fetch_row ($res, "Next")) {
                $table_id = ovrimos_result ($res, "table_id");
                $table_name = ovrimos_result ($res, "table_name");
                echo "table_id=".$table_id.", table_name=".$table_name."\n";
            } else {
                echo "Next: erreur\n";
            }
        } else {
            echo "First: erreur\n";
        }
        ovrimos_free_result ($res);
    }
    ovrimos_close ($conn);
}
?>
```

Cet exemple lit une ligne et l’affiche.

ovrimos_result (PHP 4 >= 4.0.3)

Lit le contenu d’une colonne

```
int ovrimos_result (int result_id, mixed field)
```

ovrimos_result() sert à lire le contenu d’une colonne.

ovrimos_result() lit le contenu de la colonne *`field`* dans le résultat *`result_id`*. *`field`* peut être le nom de la colonne (une chaîne) ou bien le numéro de la colonne (la première colonne est alors 1).

ovrimos_result_all (PHP 4 >= 4.0.3)

Affiche un résultat sous forme de table HTML

```
int ovrimos_result_all (int result_id [, string format])
```

ovrimos_result_all() sert à afficher tout le résultat d'une requête.

ovrimos_result_all() affiche le résultat de la requête représentée par *`result_id`*. Retourne TRUE ou FALSE.

Exemple 1. Prépare une requête, l'éxécute, et affiche le résultat

```
<?php
$conn = ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection établie!";
    $res = ovrimos_prepare ($conn, "select table_id, table_name
                                    from sys.tables where table_id = 7");
    if ($res != 0) {
        echo "Préparation faite!";
        if (ovrimos_execute ($res, array(3))) {
            echo "Exécution réussie!\n";
            ovrimos_result_all ($res);
        } else {
            echo "Exécution manquée!";
        }
        ovrimos_free_result ($res);
    } else {
        echo "Préparation manquée!\n";
    }
    ovrimos_close ($conn);
}
?>
```

Cet exemple exécute une requête SQL et affiche le résultat sous forme d'une table HTML.

Exemple 2. ovrimos_result_all() avec meta-information

```
<?php
$conn = ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection établie!";
    $res = ovrimos_exec ($conn, "select table_id, table_name
                                 from sys.tables where table_id = 1")
    if ($res != 0) {
        echo "Requête effectuée! cursor=".ovrimos_cursor ($res)."\n";
        $colnb = ovrimos_num_fields ($res);
        echo "Output columns=".$colnb."\n";
        for ($i=1; $i<=$colnb; $i++) {
            $name = ovrimos_field_name ($res, $i);
            $type = ovrimos_field_type ($res, $i);
            $len = ovrimos_field_len ($res, $i);
            echo "Colonne ".$i." nom=".$name." type=".$type." longueur=".$len."\n";
        }
        ovrimos_result_all ($res);
        ovrimos_free_result ($res);
    }
    ovrimos_close ($conn);
```

```
}
?>
```

Exemple 3. Exemple avec ovrimos_result_all()

```
<?php
$conn = ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection établie!";
    $res = ovrimos_exec ($conn, "update test set i=5");
    if ($res != 0) {
        echo "Requête effectuée!";
        echo ovrimos_num_rows ($res)." lignes affectées\n";
        ovrimos_free_result ($res);
    }
    ovrimos_close ($conn);
}
?>
```

ovrimos_num_rows (PHP 4 >= 4.0.3)

Retourne le nombre de lignes affectées par une modification

```
int ovrimos_num_rows (int result_id)
```

ovrimos_num_rows() retourne le nombre de lignes affectées par une modification

ovrimos_num_fields (PHP 4 >= 4.0.3)

Retourne le nombre de colonnes

```
int ovrimos_num_fields (int result_id)
```

ovrimos_num_fields() indique le nombre de colonnes du résultat *result_id*.

ovrimos_field_name (PHP 4 >= 4.0.3)

Retourne le nom d’une colonne

```
int ovrimos_field_name (int result_id, int field_number)
```

ovrimos_field_name() sert à obtenir le nom d’une colonne.

ovrimos_field_name() retourne le nom d’une colonne à partir de son numéro de colonne *field_number*, (la première colonne est à 1).

ovrimos_field_type (PHP 4 >= 4.0.3)

Retourne le type numérique d'une colonne

```
int ovrimos_field_type (int result_id, int field_number)
```

ovrimos_field_type() sert à connaitre le type numérique d'une colonne.

ovrimos_field_type() retourne le type numérique d'une colonne, identifiée par son numéro *field_number* dans le résultat *field_number*.

ovrimos_field_len (PHP 4 >= 4.0.3)

Retourne la taille d'une colonne

```
int ovrimos_field_len (int result_id, int field_number)
```

ovrimos_field_len() sert à connaître la taille d'une colonne.

ovrimos_field_len() retourne la taille de la colonne *field_number*, dans le résultat *field_number*.

ovrimos_field_num (PHP 4 >= 4.0.3)

Retourne le numéro de colonne

```
int ovrimos_field_num (int result_id, string field_name)
```

ovrimos_field_num() sert à connaître le nuémro de colonne, à partir de son nom.

ovrimos_field_num() retourne le numéro de la colonne *field_name* (la numérotation commence à 1), dans *result_id*.

ovrimos_free_result (PHP 4 >= 4.0.3)

Libère les ressources utilisées par un résultat

```
int ovrimos_free_result (int result_id)
```

ovrimos_free_result() sert à effacer un résultat.

ovrimos_free_result() libère toutes les ressources prises par le résultat *result_id*. Retourne TRUE.

ovrimos_commit (PHP 4 >= 4.0.3)

Valide une transaction

```
int ovrimos_commit (int connection_id)
```

ovrimos_commit() sert à exécuter une transaction.

ovrimos_commit() exécute la transaction préparée sur la connexion *connection_id*.

ovrimos_rollback (PHP 4 >= 4.0.3)

Annule une transaction

```
int ovrimos_rollback (int connection_id)
```

ovrimos_rollback() sert à annuler une transaction.

ovrimos_rollback() annule la transaction préparée sur la connexion *connection_id*.

LXII. Entrées/sorties

Les fonctions d'entrée/sorties vous permettent de contrôler quand les données sont envoyées par le script. Cela peut être utile dans certaines situations, notamment si vous devez envoyer des entêtes au navigateur après avoir envoyé des données. Ces fonctions n'affectent pas les entêtes envoyés par la fonction **header()** ou les cookies envoyés par **setcookie()**. Seules les fonctions telles que **echo()** et les données entre blocs PHP sont affectés.

Exemple 1. Exemple de gestion des sorties

```
<?php
ob_start();
echo "Bonjour\n";
setcookie ("nom_du_cookie", "valeur_du_cookie");
ob_end_flush();
?>
```

Dans l'exemple ci-dessus, la fonction **echo()** est stockée dans un buffer jusqu'à l'appel de la fonction **ob_end_flush()**. Dans le même temps, l'appel à **setcookie()** a réussi à créer un cookie, sans générer d'erreur. (D'habitude, vous devez envoyer les entêtes avant les données).

Voir aussi **header()** et **setcookie()**.

flush (PHP 3, PHP 4 >= 4.0b1)

Vide les buffers de sortie.

```
void flush (void)
```

flush() vide les buffers de sortie de PHP et tous ceux que PHP utilisait (CGI, un serveur web, etc.).

> **Note : flush()** n'a aucun effet sur la bufferisation de votre serveur web ou du navigateur.
>
> De nombreux serveur, essentiellement sous Windows, continueront à bufferiser l'affichage de votre script jusqu'à ce qu'il soit terminé, avant de transmettre les résultats à l'internaute.
>
> Même le navigateur peut mettre des informations en cache avant de les afficher. Par exemple, Netscape écrit les textes dasn un cache, jusqu'à ce qu'il ai recu une fin de ligne, ou une balise ouvrante. Il n'affichera pas les tables avant d'avoir recu la balise fermante </table>.

ob_start (PHP 4 >= 4.0b1)

Enclenche la bufferisation de sortie

```
void ob_start (string [output_callback])
```

ob_start() démarre la bufferisation de sortie. Tant qu'elle est enclenchée, aucune donnée, hormis les entêtes, n'est envoyée au navigateur, mais temporairement mis en buffer.

Le contenu de ce buffer peut être copié dans une chaîne avec la fonction **ob_get_contents()**. Pour afficher le contenu de ce buffer, utilisez **ob_end_flush()**. Au contraire, **ob_end_clean()** effacera le contenu de ce buffer.

Une fonction optionnelle de callback peut être spécifiée en troisième argument. **ob_start()** prend une chaîne comme paramètre, et retourne une chaîne. Elle sera appelée par **ob_end_flush()** ou lorsque le buffer sera envoyé au navigateur à la fin du script et recevra le contenu du buffer de sortie. Lorsque la fonction *output_callback* est appelée, elle doit retourner un nouveau contenu pour le buffer de sortie : celui-ci sera envoyé au navigateur.

> **Note :** En PHP 4.0.4, **ob_gzhandler()** a été introduit pour faciliter l'envoi de fichier compressé avec gz aux navigateurs web qui supportent les pages compressées. **ob_gzhandler()** détermine le type d'encodage accepté par un navigateur, et retourne le contenu le plus adéquat.

Les buffers de sortie sont gérés par pile, c'est à dire que vous pouvez appeler plusieurs fois **ob_start()** simultanément. Assurez-vous que vous appelez **ob_end_flush()** suffisamment souvent. Si plusieurs fonctions de callback sont actives, les contenus seront filtrés séquentiellement, dans l'ordre d'emboîtement.

Exemple 1. Exemple de callback avec fonction utilisateur

```
<?php
function callback($buffer) {
  // remplace toutes les pommes par des oranges
  return (ereg_replace("pommes de terre", "carottes", $buffer));
}
ob_start("callback");
?>
<html>
<body>
<p>C'est comme comparer des carottes et des pommes de terre.
</body>
</html>
<?php
 ob_end_flush();
?>
```

va afficher :

```
<html>
<body>
<p>C’est comme comparer des carottes et des carottes.
</body>
</html>
```

Voir aussi **ob_get_contents()**, **ob_end_flush()**, **ob_end_clean()**, **ob_implicit_flush()** et **ob_gzhandler()**.

ob_gzhandler (PHP 4 >= 4.0.4)

Fonction de callback pour la compression automatique des buffers

```
string ob_gzhandler (string buffer)
```

ob_gzhandler() est destinée à être utilisée comme fonction de callback par **ob_start()** pour faciliter l’envoi de données compressées aux navigateurs qui supportent les pages compressées. Avant que **ob_gzhandler()** envoie les données compressées, il détermine les types d’encodage qui sont supportés par le navigateur ("gzip", "deflate" ou aucun) et retourne le contenu des buffers de manière appropriée. Tous les navigateurs sont traités, car c’est aux navigateurs d’envoyer un entête indiquant les types de pages supportés.

Exemple 1. Exemple d’envoi de page compressée avec ob_gzhandler()

```
<?php
ob_start("ob_gzhandler");
?>
<html>
<body>
<p>Ceci devrait être une page compressée.
</html>
</body>
?>
```

Voir aussi **ob_start()** et **ob_end_flush()**.

ob_get_contents (PHP 4 >= 4.0b1)

Retourne le contenu du buffer de sortie

```
string ob_get_contents (void)
```

ob_get_contents() retourne le contenu du buffer de sortie si la bufferisation est activée, ou FALSE sinon.

Voir aussi **ob_start()** et **ob_get_length()**.

ob_get_length (PHP 4 >= 4.0.2)

Retourne la longueur du contenu du buffer de sortie

```
string ob_get_length (void)
```

ob_get_length() retourne la longueur du contenu du buffer de sortie si la bufferisation est activée, et FALSE sinon.

Voir aussi **ob_start()** et **ob_get_contents()**.

ob_end_flush (PHP 4 >= 4.0b1)

Envoie les données du buffer de sortie, et éteind la bufferisation de sortie

```
void ob_end_flush ()
```

ob_end_flush() envoie le contenu du buffer de sortie (s'il existe) et éteind la bufferisation de sortie. Si vous voulez continuer à manipuler la valeur du buffer, vous pouvez appeler **ob_get_contents()** avant **ob_end_flush()** car le contenu du buffer est détruit après un appel à **ob_end_flush()**.

Voir aussi **ob_start()**, **ob_get_contents()** et **ob_end_clean()**.

ob_end_clean (PHP 4 >= 4.0b1)

Détruit les données du buffer de sortie, et éteind la bufferisation de sortie

```
void ob_end_clean ()
```

ob_end_clean() détruit les données du buffer de sortie, et éteind la bufferisation.

Voir aussi **ob_start()** et **ob_end_flush()**.

ob_implicit_flush (PHP 4 >= 4.0b4)

Active/désactive l'envoi implicite

```
void ob_implicit_flush ([int flag])
```

ob_implicit_flush() active/désactive l'envoi implicite (si *flag* est fourni. Par défaut, il est activé). L'envoi implicite signifie que toute fonction qui envoie des données au navigateur veront leurs données envoyées immédiatement (la fonction **flush()** est appelée automatiquement).

Une fois que l'envoi implicite est desactivé, le buffer de sortie ne sera envoyé qu'au moment de l'appel de **ob_end_flush()**.

Voir aussi **flush()**, **ob_start()** et **ob_end_flush()**.

LXIII. PDF

Vous disposez de fonctions PDF en PHP pour créer des fichiers PDF, pour peu que vous ayez la bibliothèque PDF de Thomas Merz (disponible à : http://www.pdflib.com/pdflib/index.html (site anglais)). Vous aurez aussi besoin des librairies JPEG library (ftp://ftp.uu.net/graphics/jpeg/), the TIFF library (http://www.libtiff.org/), pour compiler cette librairie. Ces deux librairies posent pas mal de problèmes lors de la configuration. Suivez attentivement les messages d'erreur.

Reportez vous à l'excellente documentation de PDFLib, disponible avec la distribution de PDFLib. C'est une introduction très pratique des possibilités de PDFLib et elle contient la liste la plus complète et les descriptions les plus à jours des fonctions.

Toutes les fonctions de PDFLib se retrouvent dans PHP sous le même nom. De même, les paramètres sont identiques. Vous devez connaître les concepts de base de PDF ou de Postscript pour utiliser efficacement ce module. Toutes les longueurs et coordonnées sont mesurées en points Postscript points. Il y a généralement 72 points PostScript par pouce, mais cela dépend en fait de la résolution d'affichage.

Il y a un autre module PHP pour créer des document PDF, basé sur la bibliothèque FastIO's (http://www.fastio.com/)'s ClibPDF. Les API sont légèrement différentes. Reportez-vous à la section fonctions ClipPDF pour plus de détails.

Le module PDF introduit un nouveau type de variables. C'est *`pdfdoc`* : c'est un pointeur sur un document PDF et toutes les fonctions l'utilise comme premier paramètre.

Confusion entre les vieilles versions de PDFLib

Depuis le début du support de PDF sous PHP, (commençant avec la version PDFLib 0.6), il y a eu des milliers de modifications dans les API de PDFLib. La plus part de ces modifications ont été suivies par PHP, et parfois même au prix de modifications des API PHP. Depuis la version 3.x, ces API semblent s'être stabilisées, et PHP 4 a adoptée cette version comme le minimum nécessaire pour supporter PDF. En conséquence de quoi, un grand nombre de fonction vont disparaître, ou être remplacée. Le support de PDFLib 0.6 est complétement abandonné. La liste suivante indique quelles sont les fonctions obsolètes en PHP 4.02, et qui devraient être remplacées par de nouvelles versions.

Tableau 1. Fonctions obsolètes et leur remplacements

Anciennes fonctions	Nouvelles fonctions
pdf_put_image()	Désormais inutile
pdf_execute_image()	Désormais inutile
pdf_get_annotation()	**pdf_get_bookmark()** avec les mêmes paramètres.
pdf_get_font()	**pdf_get_value()** avec `"font"` comme second paramètre.
pdf_get_fontsize()	**pdf_get_value()** avec `"fontsize"` comme second paramètre.
pdf_get_fontname()	**pdf_get_parameter()** avec `"fontname"` comme second paramètre.
pdf_set_info_creator()	**pdf_set_info()** avec `"Creator"` comme second paramètre.
pdf_set_info_title()	**pdf_set_info()** avec `"Title"` comme second paramètre.
pdf_set_info_subject()	**pdf_set_info()** avec `"Subject"` comme second paramètre.
pdf_set_info_author()	**pdf_set_info()** avec `"Author"` comme second paramètre.
pdf_set_info_keywords()	**pdf_set_info()** avec `"Keywords"` comme second paramètre.
pdf_set_leading()	**pdf_set_value()** avec `"leading"` comme second paramètre.
pdf_set_text_rendering()	**pdf_set_value()** avec `"textrendering"` comme second paramètre.
pdf_set_text_rise()	**pdf_set_value()** avec `"textrise"` comme second paramètre.
pdf_set_horiz_scaling()	**pdf_set_value()** avec `"horizscaling"` comme second paramètre.
pdf_set_text_matrix()	Désormais inutile

Anciennes fonctions	Nouvelles fonctions
pdf_set_char_spacing()	**pdf_set_value()** avec `"charspacing"` comme second paramètre.
pdf_set_word_spacing()	**pdf_set_value()** avec `"wordspacing"` comme second paramètre.
pdf_set_transition()	**pdf_set_parameter()** avec `"transition"` comme second paramètre.
pdf_open()	**pdf_new()** suivi d'un appel à **pdf_open_file()**
pdf_set_font()	**pdf_findfont()** suivi d'un appel à **pdf_setfont()**
pdf_set_duration()	**pdf_set_value()** avec `"duration"` comme second paramètre.
pdf_open_gif()	**pdf_open_image_file()** avec `"gif"` comme second paramètre.
pdf_open_jpeg()	**pdf_open_image_file()** avec `"jpeg"` comme second paramètre.
pdf_open_tiff()	**pdf_open_image_file()** avec `"tiff"` comme second paramètre.
pdf_open_png()	**pdf_open_image_file()** avec `"png"` comme second paramètre.
pdf_get_imagewidth()	**pdf_get_value()** avec `"imagewidth"` comme second paramètre et l'image comme troisième.
pdf_get_imageheight()	**pdf_get_value()** avec `"imageheight"` comme second paramètre et l'image comme troisième.

Conseils pour installer PDFLib 3.x

Depuis la version 3.0 de PDFLib vous pouvez configurer cette librairie avec l'option `--enable-shared-pdflib`.

Choix de la version de PDFlib

Avec toutes les versions de PHP 4, ultérieure au 9 mars 2000, vous devez utiliser PDFlib 3.0 ou plus récent.

PHP 3, d'un autre coté, ne doit pas être utilisé avec une version plus récente que la 2.01. Depuis la version 1.61 du source `php3/functions/pdf.c` (php 3.19), il est possible d'utiliser la version PDFlib 3.0 ou plus récent.

Installation des anciennes versions de PDFLib

Si vous utilisez PDFLib 2.01 vérifiez comment votre librairie a été installée. Il doit y avoir un fichier (ou un lien) vers `libpdf.so`. La version 2.01 ne fait que créer une librairie avec le nom `libpdf2.01.so` qui ne peut être trouvé lors de la compilation du programme de configuration. Vous devez créer vous même ce lien symbolique de `libpdf.so` vers `libpdf2.01.so`.

La version 2.20 de PDFLib a introduit de nombreuses modifications dans ses API, ainsi que le support des polices chinoises et japonaises. Cela impliquent malheureusement des modifications dans le module PDF de PHP 4 (mais pas de PHP 3). Si vous utilisez PDFLib 2.20, gérer correctement votre mémoire. Jusqu'à la version 3.0, PDFLib peut se révéler très instable. Le paramètre d'encodage **pdf_set_font()** est devenu une chaîne. Cela signifique notamment qu'il faut remplacer 4 par 'winansi'.

Si vous utilisez PDFLib 2.30, pdf_set_text_matrix() a disparu. Elle n'est plus supporté. En général, il est recommandé de consulter les notes de version de la PDFLib pour lister toutes les modifications.

A partir du 9 mars 2000, PHP 4 ne supporte plus que la version 3.0 et plus récente de PDFLib. PHP 3, par contre, ne doit pas être utilisé avec des versions plus récentes que la 2.01.

Exemples

La plus part des fonctions sont simples d'emploi. Le plus difficile est probablement de créer un fichier PDF simple. L'exemple suivant devrait vous mettre sur les rails. Il crée un fichier `test.pdf` d'une page. La page contient du texte "Times Roman outlined", de taille de 30pt. Le texte est aussi souligné.

Exemple 1. Création d'un document PDF avec PDFLib

```
<?php
$pdf = pdf_new();
pdf_open_file($pdf, "test.pdf");
pdf_set_info($pdf, "Author", "Uwe Steinmann");
pdf_set_info($pdf, "Title", "Test for PHP wrapper of PDFLib 2.0");
pdf_set_info($pdf, "Creator", "See Author");
pdf_set_info($pdf, "Subject", "Testing");
pdf_begin_page($pdf, 595, 842);
pdf_add_outline($pdf, "Page 1");
pdf_set_font($pdf, "Times-Roman", 30, "host");
pdf_set_value($pdf, "textrendering", 1);
pdf_show_xy($pdf, "Times Roman outlined", 50, 750);
pdf_moveto($pdf, 50, 740);
pdf_lineto($pdf, 330, 740);
pdf_stroke($pdf);
pdf_end_page($pdf);
pdf_close($pdf);
pdf_delete($pdf);
echo "<A HREF=getpdf.php>finished</A>";
?>
```

Ce script `getpdf.php` retourne simplement le document PDF.

```
<?php
$fp = fopen("test.pdf", "r");
header("Content-type: application/pdf");
fpassthru($fp);
fclose($fp);
?>
```

La distribution PDFLib contient un exemple plus complexe, qui crée des pages plus élaborées, avec une horloge. Cet exemple a été converti en script PHP (vous retrouverez cet exemple dans le module clibpdf). Il utilise les possibilités de création de fichier en mémoire, sans fichier temporaire.

Exemple 2. Exemple pdfclock issue de la distribution PDFLib

```
<?php
$radius = 200;
$margin = 20;
$pagecount = 10;
$pdf = pdf_new();
if (!pdf_open_file($pdf, "")) {
    print error;
    exit;
};
pdf_set_parameter($pdf, "warning", "true");
pdf_set_info($pdf, "Creator", "pdf_clock.php");
pdf_set_info($pdf, "Author", "Uwe Steinmann");
```

```
pdf_set_info($pdf, "Title", "Analog Clock");
while($pagecount-- > 0) {
    pdf_begin_page($pdf, 2 * ($radius + $margin), 2 * ($radius + $margin));
    pdf_set_parameter($pdf, "transition", "wipe");
    pdf_set_value($pdf, "duration", 0.5);
    pdf_translate($pdf, $radius + $margin, $radius + $margin);
    pdf_save($pdf);
    pdf_setrgbcolor($pdf, 0.0, 0.0, 1.0);
    /* indicateurs de minutes */
    pdf_setlinewidth($pdf, 2.0);
    for ($alpha = 0; $alpha < 360; $alpha += 6) {
        pdf_rotate($pdf, 6.0);
        pdf_moveto($pdf, $radius, 0.0);
        pdf_lineto($pdf, $radius-$margin/3, 0.0);
        pdf_stroke($pdf);
    }
    pdf_restore($pdf);
    pdf_save($pdf);
    /* indicateurs de 5 minutes */
    pdf_setlinewidth($pdf, 3.0);
    for ($alpha = 0; $alpha < 360; $alpha += 30) {
        pdf_rotate($pdf, 30.0);
        pdf_moveto($pdf, $radius, 0.0);
        pdf_lineto($pdf, $radius-$margin, 0.0);
        pdf_stroke($pdf);
    }
    $ltime = getdate();
    /* aiguille des heures */
    pdf_save($pdf);
    pdf_rotate($pdf,-(($ltime['minutes']/60.0)+$ltime['hours']-3.0)*30.0);
    pdf_moveto($pdf, -$radius/10, -$radius/20);
    pdf_lineto($pdf, $radius/2, 0.0);
    pdf_lineto($pdf, -$radius/10, $radius/20);
    pdf_closepath($pdf);
    pdf_fill($pdf);
    pdf_restore($pdf);
    /* aiguille des minutes */
    pdf_save($pdf);
    pdf_rotate($pdf,-(($ltime['seconds']/60.0)+$ltime['minutes']-15.0)*6.0);
    pdf_moveto($pdf, -$radius/10, -$radius/20);
    pdf_lineto($pdf, $radius * 0.8, 0.0);
    pdf_lineto($pdf, -$radius/10, $radius/20);
    pdf_closepath($pdf);
    pdf_fill($pdf);
    pdf_restore($pdf);
    /* aiguille des secondes */
    pdf_setrgbcolor($pdf, 1.0, 0.0, 0.0);
    pdf_setlinewidth($pdf, 2);
    pdf_save($pdf);
    pdf_rotate($pdf, -(($ltime['seconds'] - 15.0) * 6.0));
    pdf_moveto($pdf, -$radius/5, 0.0);
    pdf_lineto($pdf, $radius, 0.0);
    pdf_stroke($pdf);
    pdf_restore($pdf);
    /* petit cercle au centre */
    pdf_circle($pdf, 0, 0, $radius/30);
    pdf_fill($pdf);
    pdf_restore($pdf);
    pdf_end_page($pdf);
    # pour voir une différence
    sleep(1);
}
pdf_close($pdf);
$buf = pdf_get_buffer($pdf);
$len = strlen($buf);
header("Content-type: application/pdf");
```

```
header("Content-Length: $len");
header("Content-Disposition: inline; filename=foo.pdf");
print $buf;
pdf_delete($pdf);
?>
```

pdf_add_annotation (PHP 3>= 3.0.12, PHP 4)

Obsolète: Ajoute une annotation

pdf_add_outline() est remplacé par **pdf_add_note()**

Voir aussi **pdf_add_note()**.

pdf_add_bookmark (PHP 4 >= 4.0.1)

Ajoute un signet à la page courante

```
int pdf_add_bookmark (resource pdf_object, string text [, int parent [, int open]])
```

pdf_add_bookmark() ajoute un signet sous le père *parent*, ou un signet général, si *parent* vaut 0.
pdf_add_bookmark() retourne un descripteur de signet, qui peut être utilisé comme père d'un autre signet. Si *open* vaut 1, les signets fils seront ouverts. Ils seront fermés sinon.

pdf_add_launchlink (PHP 4 >= 4.0.5)

Ajoute une annotation éxécutable dans la page courante

```
int pdf_add_launchlink (resource pdf_object, double llx, double lly, double urx, double
ury, string filename)
```

pdf_add_launchlink() ajoute une annotation exécutable (le fichier de destination peut etre n'importe quel fichier).

pdf_add_locallink (PHP 4 >= 4.0.5)

Ajoute un lien sur une annotation dans la page courante

```
int pdf_add_locallink (resource pdf_object, double llx, double lly, double urx, double
ury, int page, string dest)
```

Add a link annotation to a target within the current PDF file.

pdf_add_note (PHP 4 >= 4.0.5)

Ajoute une note d'annotation dans la page courante

```
int pdf_add_note (resource pdf_object, double llx, double lly, double urx, double ury,
string contents, string title, string icon, int open)
```

pdf_add_note() ajoute une note d'annotation. *icon* peut prendre une des valeurs suivantes : "comment" (commentaire), "insert" (insertion), "note" (note), "paragraph" (paragraphe), "newparagraph" (nouveau paragraphe), "key" (cle), ou "help" (aide).

pdf_add_outline (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Obsolète: Ajoute un signet dans la page courante

Obsolète.

Voir aussi **pdf_add_bookmark()**.

pdf_add_pdflink (PHP 3>= 3.0.12, PHP 4)

Ajoute un lien sur un fichier dans la page courante

```
int pdf_add_pdflink (resource pdf_object, double llx, double lly, double urx, double ury,
string filename, int page, string dest)
```

pdf_add_pdflink() ajoute un lien vers un fichier PDF, sous forme d'annotation.

pdf_add_weblink (PHP 3>= 3.0.12, PHP 4)

Ajoute un lien hypertexte dans la page courante

```
int pdf_add_weblink (resource pdf_object, double llx, double lly, double urx, double ury,
string url)
```

pdf_add_weblink() ajoute un lien hypertexte vers une URL sur le web.

pdf_arc (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Dessine un arc.

```
void pdf_arc (resource pdf_object, double x-coor, double y-coor, double radius, double
start, double end)
```

pdf_arc() dessine un arc de cercle, de centre (*x-coor*, *y-coor*) et de rayon *radius*, en commencant à l'angle *start* et finissant à l'angle *end*.

Voir aussi **pdf_circle()**, **pdf_stroke()**.

pdf_attach_file (PHP 4 >= 4.0.5)

Attache un fichier à la page courante

```
int pdf_attach_file (resource pdf_object, double llx, double lly, double urx, double ury,
string filename, string description, string author, string mimetype, string icon)
```

pdf_attach_file() attache un fichier à la page courante. *icon* peut prendre l'une des valeurs suivantes : "graph" (image), "paperclip" (texte), "pushpin" (??? NdTraducteur : mailez moi!) ou "tag" (étiquette).

pdf_begin_page (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Commence une nouvelle page.

```
void pdf_begin_page (resource pdf_object, double width, double height)
```

pdf_begin_page() commence une nouvelle page avec la taille *height* et la largueur *width*. Afin de créer un nouveau document valide, vous devez appeler cette fonction et **pdf_end_page()** au moins une fois.

Voir aussi **pdf_end_page()**.

pdf_circle (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Dessine un cercle.

```
void pdf_circle (resource pdf_object, double x-coor, double y-coor, double radius)
```

pdf_circle() dessine un cercle de centre (*x-coor*, *y-coor*), et de rayon *radius*.

Voir aussi **pdf_arc()**, **pdf_stroke()**.

pdf_clip (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Aligne sur le chemin courant.

```
void pdf_clip (resource pdf_object)
```

pdf_clip() aligne tous les dessins sur le chemin courant.

pdf_close (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Ferme un document PDF.

```
void pdf_close (resource pdf_object)
```

pdf_close() ferme un document PDF.

Voir aussi **pdf_open()**, **fclose()**.

pdf_closepath (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Ferme et clos le chemin.

```
void pdf_closepath (resource pdf_object)
```

pdf_closepath() ferme et clos le chemin courant. Cela signifie qu'une ligne va être ajoutée entre le point courant et le premier point du chemin. De nombreuses fonctions telles que **pdf_moveto()**, **pdf_circle()** et **pdf_rect()** démarre un nouveau chemin.

pdf_closepath_fill_stroke (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Remplis, dessine et ferme le chemin courant.

```
void pdf_closepath_fill_stroke (resource pdf_object)
```

pdf_closepath_fill_stroke() clos le chemin, le remplis avec la couleur courante, et dessine le chemin.

Voir aussi **pdf_closepath()**, **pdf_stroke()**, **pdf_fill()**, **pdf_setgray_fill()**, **pdf_setgray()**, **pdf_setrgbcolor_fill()**, **pdf_setrgbcolor()**.

pdf_closepath_stroke (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Ferme le chemin et dessine le long du chemin.

```
void pdf_closepath_stroke (resource pdf_object)
```

pdf_closepath_stroke() est une combinaison de **pdf_closepath()** et **pdf_stroke()**. Elle ferme aussi le chemin.

Voir aussi **pdf_closepath()**, **pdf_stroke()**.

pdf_close_image (PHP 3>= 3.0.7, PHP 4)

Ferme une image

```
void pdf_close_image (resource image)
```

pdf_close_image() ferme une image qui a été ouverte par **pdf_open_gif()** ou **pdf_open_jpeg()**.

Voir aussi **pdf_open_jpeg()**, **pdf_open_gif()**, **pdf_open_tiff()** et **pdf_open_memory_image()**.

pdf_concat (PHP 4 >= 4.0.5)

Concatène une matrice avec CTM

```
void pdf_concat (resource pdf_object, double a, double b, double c, double d, double e,
double f)
```

pdf_concat() concatène une matrice avec CTM.

pdf_continue_text (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Affiche un texte sur une nouvelle ligne.

```
void pdf_continue_text (resource pdf_object, string text)
```

pdf_continue_text() affiche le texte *text* sur une nouvelle ligne. La distance entre les lignes peut être choisie avec **pdf_set_leading()**.

Voir aussi **pdf_show_xy()**, **pdf_set_leading()** et **pdf_set_text_pos()**.

pdf_curveto (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Dessine une courbe.

```
void pdf_curveto (resource pdf_object, double x1, double y1, double x2, double y2, double x3, double y3)
```

pdf_curveto() dessine une courbe de Bézier entre le point courant et le point (*x3*, *y3*) en utilisant les points de contrôle (*x1*, *y1*) et (*x2*, *y2*).

Voir aussi **pdf_moveto()**, **pdf_lineto()** et **pdf_stroke()**.

pdf_delete (PHP 4 >= 4.0.5)

Efface un objet PDF

```
void pdf_delete (resource pdf_object)
```

pdf_delete() efface un objet PDF et libère les ressources.

pdf_end_page (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Termine une page.

```
void pdf_end_page (resource pdf_object)
```

pdf_end_page() termine une page. Une fois qu'une page a été fermée, elle ne peut pas être modifiée.

Voir aussi **pdf_begin_page()**.

pdf_endpath (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Ferme le chemin courant

```
void pdf_endpath (resource pdf_object)
```

pdf_endpath() ferme le chemin courant mais ne le clôt pas.

Voir aussi **pdf_closepath()**.

pdf_fill (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Remplis le chemin courant.

```
void pdf_fill (resource pdf_object)
```

pdf_fill() remplis l'intérieur du chemin courant avec la couleur courante.

Voir aussi **pdf_closepath()**, **pdf_stroke()**, **pdf_setgray_fill()**, **pdf_setgray()**, **pdf_setrgbcolor_fill()**, **pdf_setrgbcolor()**.

pdf_fill_stroke (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Remplis et dessine le chemin courant.

```
void pdf_fill_stroke (resource pdf_object)
```

pdf_fill_stroke() remplis l'intérieur du chemin courant avec la couleur courante, puis dessine le chemin courant.

Voir aussi **pdf_closepath()**, **pdf_stroke()**, **pdf_fill()**, **pdf_setgray_fill()**, **pdf_setgray()**, **pdf_setrgbcolor_fill()**, **pdf_setrgbcolor()**.

pdf_findfont (PHP 4 >= 4.0.5)

Prépare une police pour utilisation ultérieure

```
void pdf_findfont (resource pdf_object, string fontname, string encoding, int embed)
```

pdf_findfont() prépare une police pour utilisation ultérieure avec **pdf_setfont()**. Les dimensions seront chargées, et lorsque c'est possible, le fichier de police sera vérifié, mais pas utilisé. *encoding* peut prendre l'une des valeurs suivantes : "builtin" (intégrée), "macroman", "winansi", "host", et nom d'encodage utilisateur, ou encore nom de CMap.

pdf_get_buffer (PHP 4 >= 4.0.5)

Lit un buffer contenant des données PDF

```
string pdf_get_buffer (resource pdf_object)
```

pdf_get_buffer() lit le contenu du buffer *pdf_object*. Le résultat doit être utilisé par le client avant d'appeler toute autre fonction PDFLib.

pdf_get_font (PHP 4 >= 4.0b4)

Obsolète : gestion de police

Obsolète.

Voir aussi **pdf_get_value()**.

pdf_get_fontname (PHP 4 >= 4.0b4)

Obsolète : gestion de nom de police

Obsolète.

Voir aussi **pdf_get_parameter()**.

pdf_get_fontsize (PHP 4 >= 4.0b4)

Obsolète : gestion de taille de police

Obsolète.

Voir aussi **pdf_get_value()**.

pdf_get_image_height (PHP 3>= 3.0.12, PHP 4)

Retourne la hauteur d'une image

```
string pdf_get_image_height (resource pdf_object, resource image)
```

pdf_get_image_height() retourne la hauteur de l'image *image* en pixels.

Voir aussi **pdf_open_image_file()**, **pdf_open_memory_image()** et **pdf_get_image_width()**.

pdf_get_image_width (PHP 3>= 3.0.12, PHP 4)

Retourne la largeur d'une image

```
string pdf_get_image_width (resource pdf_object, resource image)
```

pdf_get_image_width() retourne la largeur de l'image *image*, en pixels.

Voir aussi **pdf_open_image_file()**, **pdf_open_memory_image()** et **pdf_get_image_height()**.

pdf_get_parameter (PHP 4 >= 4.0.1)

Lit la valeur d'un paramètre PDFLib chaîne

```
string pdf_get_parameter (resource pdf_object, string key [, double modifier])
```

pdf_get_parameter() lit le contenu de certains paramètres PDFLib, au format chaîne de caractères.

pdf_get_value (PHP 4 >= 4.0.1)

Lit la valeur d'un paramètre PDFLib numérique

```
double pdf_get_value (resource pdf_object, string key [, double modifier])
```

pdf_get_value() lit le contenu de certains paramètres PDFLib, au format numérique.

pdf_lineto (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Dessine une ligne.

```
void pdf_lineto (resource pdf_object, double x-coor, double y-coor)
```

pdf_lineto() dessine une ligne entre le point courant et le point de coordonnées (*x-coor*, *y-coor*).

Voir aussi **pdf_moveto()**, **pdf_curveto()** et **pdf_stroke()**.

pdf_moveto (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Déplace le point courant.

```
void pdf_moveto (resource pdf_object, double x-coor, double y-coor)
```

pdf_moveto() déplace le point courant à la position (*x-coor*, *y-coor*).

pdf_new (PHP 4 >= 4.0.5)

Crée un nouvel objet PDF

```
resource pdf_new (void)
```

pdf_new() crée un nouvel objet PDF, avec gestion des erreurs et de la mémoire par défaut.

pdf_open (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Obsolète: Ouvre un nouvel objet PDF

pdf_open() est obsolète. Utilisez **pdf_new()** puis **pdf_open_file()**.

Voir aussi **pdf_new()** et **pdf_open_file()**.

pdf_open_CCITT (PHP 4 >= 4.0.5)

Ouvre une nouvelle image à partir de données CCITT

```
int pdf_open_ccitt (resource pdf_object, string filename, int width, int height, int
BitReverse, int k, int Blackls1)
```

pdf_open_ccitt() ouvre une image CCITT.

pdf_open_file (PHP 4 >= 4.0.5)

Ouvre un nouvel objet PDF

```
int pdf_open_file (resource pdf_object [, string filename])
```

pdf_open_file() crée un nouvel objet PDF à partir du fichier *filename*. Si *filename* est vide, le fichier PDF sera généré en mémoire. Le résultat devrat être lu avec la fonction **pdf_get_buffer()** function.

L'exemple suivant montre comment créer un fichier PDF en mémoire, et l'envoyer correctement au navigateur.

Exemple 1. Création d'un fichier PDF en mémoire

```
<?php
$pdf = pdf_new();
pdf_open_file($pdf);
pdf_begin_page($pdf, 595, 842);
pdf_set_font($pdf, "Times-Roman", 30, "host");
pdf_set_value($pdf, "textrendering", 1);
pdf_show_xy($pdf, "Un document PDF créé en memoire!", 50, 750);
```

```
pdf_end_page($pdf);
pdf_close($pdf);
$data = pdf_get_buffer($pdf);
header("Content-type: application/pdf");
header("Content-disposition: inline; filename=test.pdf");
header("Content-length: " . strlen($data));
echo $data;
?>
```

pdf_open_gif (PHP 3>= 3.0.7, PHP 4)

Obsolète: Ouvre une image GIF

Obsolète.

Voir aussi **pdf_open_image()**,

pdf_open_image (PHP 4 >= 4.0.5)

Fonction générique pour les images

```
int pdf_open_image (int PDF-document, string imagetype, string source, string data, long
length, int width, int height, int components, int bpc, string params)
```

pdf_open_image() ouvre des fichiers de divers formats d'images. *imagetype* peut prendre l'une des valeurs suivantes : "jpeg", "ccitt", "raw". *source* peut prendre l'une des valeurs suivantes : "memory" (mémoire), "fileref" (pointeur de fichier), "url". *length* ne sert que pour le type "raw"; *params* ne sert qur pour le type "ccitt".

pdf_open_image_file (PHP 3 CVS only, PHP 4 >= 4.0RC2)

Lit une image depuis un fichier

```
int pdf_open_image_file (resource pdf_object, string imagetype, string filename [, string
stringparam [, string intparam]])
```

pdf_open_image_file() ouvre une image dans le fichier *filename*. *imagetype* peut prendre une des valeurs suivantes : "jpeg", "tiff", "gif", et "png". *stringparam* peut prendre l'une des valeurs suivantes : "", "mask", "masked", ou "page". *intparam* peut valoir 0, l'id de l'image du masque appliqué, ou la page.

pdf_open_png (PHP 4 >= 4.0RC2)

Obsolète: Ouvre une image PNG

Obsolète.

Voir aussi **pdf_open_image()**,

pdf_open_jpeg (PHP 3>= 3.0.7, PHP 4)

Obsolète: Ouvre une image JPEG

Obsolète.

Voir aussi **pdf_open_image()**.

pdf_open_tiff (PHP 4 >= 4.0b4)

Obsolète: Ouvre une image TIFF

```
int pdf_open_tiff (int PDF-document, string filename)
```

Obsolète.

Voir aussi **pdf_open_image()**.

pdf_place_image (PHP 3>= 3.0.7, PHP 4)

Place une image dans la page.

```
void pdf_place_image (resource pdf_object, resource image, double x-coor, double y-coor, double scale)
```

pdf_place_image() place l'image *image* dans la page courante, à la position (*x-coor*, *x-coor*). L'image peut changer d'échelle simultanémement.

pdf_rect (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Dessine un rectangle.

```
void pdf_rect (resource pdf_object, double x-coor, double y-coor, double width, double height)
```

pdf_rect() dessine un rectangle un rectangle de coin inférieur gauche de coordonnées (*x-coor*, *y-coor*). Sa longueur vaut *width*. Et sa largeur *height*.

Voir aussi **pdf_stroke()**.

pdf_restore (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Restaure un environnement sauvé.

```
void pdf_restore (resource pdf_object)
```

pdf_restore() restaure un environnement sauvé par **pdf_save()**. Cela fonctionne de manière identique à la commande Postscript grestore. Très pratique lorsque vous vous faire des translations ou des rotations sans affecter les autres objets.

Exemple 1. Sauver et restaurer un environnement PDF

```
<?php
pdf_save($pdf);
// tout un lot de rotations, translations, transformations...
pdf_restore($pdf)
?>
```

Voir aussi **pdf_save()**.

pdf_rotate (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Choisi la rotation.

```
void pdf_rotate (resource pdf_object, double angle)
```

pdf_rotate() modifie la rotation de *angle* degré.

pdf_save (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Enregistre l'environnement courant.

```
void pdf_save (resource pdf_object)
```

pdf_save() enregistre l'environnement courant. Le fonctionnement est identique à la commande postscript gsave. Très pratique si vous voulez faire une translation ou une rotation d'un objet, sans affecter les autres. **pdf_save()** sera toujours suivi d'un **pdf_restore()**.

Voir aussi **pdf_restore()**.

pdf_scale (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Modifie l'échelle.

```
void pdf_scale (resource pdf_object, double x-scale, double y-scale)
```

pdf_scale() modifie l'échelle dans les deux directions. L'exemple suivant multiplie l'échelle par 72. La ligne suivante sera dessinée sur un pouce (2.54 cm) de large.

Exemple 1. Mise à l'échelle

```
<?php pdf_scale($pdf, 72.0, 72.0);
pdf_lineto($pdf, 1, 1);
pdf_stroke($pdf);
?>
```

pdf_setdash (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Modifie les caractères de remplissage.

```
void pdf_setdash (resource pdf_object, double white, double black)
```

pdf_setdash() modifie les caractères de remplissage, et affecte *white* comme caractère invisible, et *black* comme caractère de remplissage. Si les deux sont à zéros, une ligne continue est affichée.

pdf_setflat (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Modifie la platitude (flatness).

```
void pdf_setflat (resource pdf_object, double value)
```

pdf_setflat() modifie la platitude, et lui affecte la valeur *value* entre 0 et 100.

pdf_setfont (PHP 4 >= 4.0.5)

Modifie la police courante

```
void pdf_setfont (resource pdf_object, int font, double size)
```

pdf_setfont() remplace la police courante par *font*, à la taille *size*. *font* est créé par **pdf_findfont()**.

pdf_setgray (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Modifie la couleur grise comme couleur de remplissage et de dessin.

```
void pdf_setgray (resource pdf_object, double gray value)
```

pdf_setgray() modifie la couleur grise comme couleur de remplissage et de dessin.

Voir aussi **pdf_setrgbcolor_stroke()** et **pdf_setrgbcolor_fill()**.

pdf_setgray_fill (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Modifie la couleur grise comme couleur de remplissage.

```
void pdf_setgray_fill (resource pdf_object, double gray value)
```

pdf_setgray_fill() modifie la couleur grise comme couleur de remplissage.

Voir aussi **pdf_setrgbcolor_fill()**.

pdf_setgray_stroke (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Modifie la couleur de dessin à un niveau de gris.

```
void pdf_setgray_stroke (resource pdf_object, double gray value)
```

pdf_setgray_stroke() modifie la couleur de dessin à un niveau de gris.
Voir aussi **pdf_setrgbcolor_stroke()**.

pdf_setlinecap (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Modifie le paramètre linecap.

```
void pdf_setlinecap (resource pdf_object, int value)
```

pdf_setlinecap() affecte au paramètre "linecap" la valeur *value*, entre 0 et 2.

pdf_setlinejoin (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Modifie le paramètre linejoin.

```
void pdf_setlinejoin (resource pdf_object, long value)
```

pdf_setlinejoin() modifie le paramètre "linejoin", et lui affecte la valeur *value*, entre 0 et 2.

pdf_setlinewidth (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Modifie la largeur de ligne.

```
void pdf_setlinewidth (resource pdf_object, double width)
```

pdf_setlinewidth() affecte à largeur de ligne la valeur *width*.

pdf_setmiterlimit (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Modifie la "miter limit".

```
void pdf_setmiterlimit (resource pdf_object, double value)
```

pdf_setmiterlimit() modifie la "miter limit" et lui affecte la valeur *value*, supérieure à 1.

pdf_setpolydash (PHP 4 >= 4.0.5)

Modifie les pointillés compliqués

```
void pdf_setpolydash (resource pdf_object, array dasharray)
```

pdf_setpolydash() modifie les pointillés complexes, définit par le tableau *dasharray*.

pdf_setrgbcolor (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Modifie la couleur rgb comme couleur de dessin et de remplissage.

```
void pdf_setrgbcolor (resource pdf_object, double red value, double green value, double
blue value)
```

pdf_setrgbcolor_stroke() modifie la couleur RGB comme couleur de remplissage.

Voir aussi **pdf_setrgbcolor_stroke()** et **pdf_setrgbcolor_fill()**.

pdf_setrgbcolor_fill (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Modifie la couleur rgb comme couleur de remplissage.

```
void pdf_setrgbcolor_fill (resource pdf_object, double red value, double green value,
double blue value)
```

pdf_setrgbcolor_fill() choisi la couleur RGB comme couleur de remplissage.

Voir aussi **pdf_setrgbcolor_fill()**.

pdf_setrgbcolor_stroke (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Modifie la couleur rgb comme couleur de dessin.

```
void pdf_setrgbcolor_stroke (resource pdf_object, double red value, double green value,
double blue value)
```

pdf_setrgbcolor_stroke() choisi la couleur RGB comme couleur de dessin.

Voir aussi **pdf_setrgbcolor_fill()**.

pdf_set_border_color (PHP 3>= 3.0.12, PHP 4)

Modifie la couleur des liens et annotations

```
void pdf_set_border_color (resource pdf_object, double red, double green, double blue)
```

pdf_set_border_color() modifie la couleur des bords de liens et d'annotation. Les trois composants *red*, *green*, *blue* représentent une couleur RGB (rouge, vert, bleu) et leur valeur doivent être comprise entre 0 et 1.

Voir aussi **pdf_set_border_style()** et **pdf_set_border_dash()**.

pdf_set_border_dash (PHP 4 >= 4.0.1)

Modifie les pointillés des liens et annotations

```
void pdf_set_border_dash (resource pdf_object, double black, double white)
```

pdf_set_border_dash() modifie la longueur des pointillés (si le style de bord d'une annotation est en pointillés). *black* représente la taille des traits noirs, et *white* celle des espaces blancs.

Voir aussi **pdf_set_border_style()** et **pdf_set_border_color()**.

pdf_set_border_style (PHP 3>= 3.0.12, PHP 4)

Modifie le bord des liens et annotations

```
void pdf_set_border_style (resource pdf_object, string style, double width)
```

pdf_set_border_style() modifie le style des bords de liens et d'annotation. *style* peut valoir 'solid' (trait plain) ou 'dashed'(pointillé).

Voir aussi **pdf_set_border_color()**, **pdf_set_border_dash()**.

pdf_set_char_spacing (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Fixe l'espacement des caractères.

```
void pdf_set_char_spacing (resource pdf_object, double space)
```

pdf_set_char_spacing() modifie l'espacement des caractères.

Voir aussi **pdf_set_word_spacing()** et **pdf_set_leading()**.

pdf_set_duration (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Choisi la durée de transition entre deux pages.

```
void pdf_set_duration (resource pdf_object, double duration)
```

pdf_set_duration() choisi la durée de transition, en secondes, entre deux pages.

pdf_set_font (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Sélectionne la police et sa taille.

```
void pdf_set_font (resource pdf_object, string font name, double size, string encoding [,
int embed])
```

pdf_set_font() sélectionne la police, sa taille et son encodage. Il vous faudra fournir des fichiers Adobe Font Metrics (afm) comme police, dans le dossier de police (par défaut ./fonts). Si vous utilisez PDFLib 0.6, vous devrez fournir des fichiers Adobe Font Métric (afm-files) pour les polices, dans le chemin de police (par défaut, ./fonts). Si vous utilisez php versin 3 ou une version plus ancienne que la version 2.20 de PDFLib, le quatrième paramètre *encoding* peut prendre les valeurs suivantes : 0 = builtin, 1 = pdfdoc, 2 = macroman, 3 = macexpert, 4 = winansi. Un encodage plus grand que 4 et inférieur à 0 sera transformé en 'winansi'. 'winansi' est souvent un bon choix. Si vous utilisez PHP version 4 et une version plus ancienne que la version 2.20 de PDFLib le quatrième paramètre *encoding* est une chaîne : 'builtin', 'pdfdoc', 'macroman', 'macexpert', 'winansi'. Si le dernier paramètre est à 1, la police est intégrée dans le document. Sinon, elle ne le sera pas. Incorporer une police dans un document est un bonne idée si la police n'est pas répandue, ou si vous ne pouvez pas vous assurez que le la personne qui regardera votre document peut accéder à cette police.

Note : pdf_set_font() doit être appelée après **pdf_begin_page()** pour créer un document PDF valide.

Note : Si vous référencez une police dans un fichier `.upr`, assurez-vous que le nom du fichier `.afm` et celui de la police sont bien les mêmes. Sinon, la police sera aggrandie plusieur fois (Merci à Paul Haddon pour cette info).

pdf_set_horiz_scaling (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Fixe l'echelle horizontale du texte.

```
void pdf_set_horiz_scaling (resource pdf_object, double scale)
```

pdf_set_horiz_scaling() fixe l'échelle horizontale du texte, à *`scale`* en pourcentage.

pdf_set_info (PHP 4 >= 4.0.1)

Remplis les entêtes du document

```
void pdf_set_info (resource pdf_object, string fieldname, string value)
```

pdf_set_info() modifie un champs d'entête d'un document PDF. Les valeurs possibles pour *`fieldname`* sont : 'Subject' (sujet), 'Title'(titre), 'Creator'(créateur), 'Author'(auteur), 'Keywords'(mots-clé) et un autre nom, défini par l'utilisateur. **pdf_set_info()** peut être appelée avant la création d'une page.

Exemple 1. Préparer l'entête d'un document PDF

```
<?php
$fd = fopen("test.pdf", "w");
$pdfdoc = pdf_open($fd);
pdf_set_info($pdfdoc, "Author", "Uwe Steinmann");
pdf_set_info($pdfdoc, "Creator", "Uwe Steinmann");
pdf_set_info($pdfdoc, "Title", "Testing Info Fields");
pdf_set_info($pdfdoc, "Subject", "Test");
pdf_set_info($pdfdoc, "Keywords", "Test, Fields");
pdf_set_info($pdfdoc, "CustomField", "What ever makes sense");
pdf_begin_page($pdfdoc, 595, 842);
pdf_end_page($pdfdoc);
pdf_close($pdfdoc);
?>
```

Note : pdf_set_info() remplace pdf_set_info_keywords(), pdf_set_info_title(), pdf_set_info_subject(), pdf_set_info_creator() et pdf_set_info_subject().

pdf_set_leading (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Obsolète : Modifie la distance entre les lignes du texte

Obsolète.

Voir aussi **pdf_set_value()**.

pdf_set_parameter (PHP 4 >= 4.0RC1)

Modifie certains paramètres.

```
void pdf_set_parameter (resource pdf_object, string name, string value)
```

pdf_set_parameter() modifie certaines valeurs de pdglib. *value* est de type chaîne.

Voir aussi **pdf_get_value()**, **pdf_set_value()** et **pdf_get_parameter()**.

pdf_set_text_pos (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Fixe la position du texte.

```
void pdf_set_text_pos (resource pdf_object, double x-coor, double y-coor)
```

pdf_set_text_pos() modifie la position du texte qui sera utilisée lors du prochain **pdf_show()**.

Voir aussi **pdf_show()** et **pdf_show_xy()**.

pdf_set_text_rendering (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Détermine le rendu du texte.

```
void pdf_set_text_rendering (resource pdf_object, int mode)
```

pdf_set_text_rendering() détermine le rendu du texte. Les valeurs possibles pour *mode* sont 0=fill text (texte plein), 1=stroke text (???), 2=fill et stroke text (texte plein et stroke), 3=invisible, 4=texte plein, et ajouté au chemin, 5=stroke text, ajouté au chemin, 6=texte plein et stroke, ajouté au chemin, 7=ajouté au chemin.

pdf_set_text_matrix (PHP 3>= 3.0.6, 4.0b1 - 4.0b4 only)

Obsolète: Modifie la transition des pages

Voir **pdf_set_parameter()**.

pdf_set_value (PHP 4 >= 4.0.1)

Modifie certains paramètre numériques

```
void pdf_set_value (resource pdf_object, string name, double value)
```

pdf_set_value() modifie la valeur (numérique) du paramètre *name* de PDFLib.

Voir aussi **pdf_get_value()**, **pdf_get_parameter()** et **pdf_set_parameter()**.

pdf_set_word_spacing (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Fixe l'espacement des mots.

```
void pdf_set_word_spacing (resource pdf_object, double space)
```

pdf_set_word_spacing() modifie l'espacement des mots.

Voir aussi **pdf_set_char_spacing()** et **pdf_set_leading()**.

pdf_show (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Affiche un texte à la position courante.

```
void pdf_show (resource pdf_object, string text)
```

pdf_show() affiche le texte *text* avec la position courante, et avec la police courante.

Voir aussi **pdf_show_xy()**, **pdf_show_boxed()**, **pdf_set_text_pos()** et **pdf_set_font()**.

pdf_show_boxed (PHP 4 >= 4.0RC1)

Affiche un texte dans un rectangle.

```
int pdf_show_boxed (resource pdf_object, string text, double x-coor, double y-coor, double
width, double height, string mode [, string feature])
```

pdf_show_boxed() affiche le texte *text* dans un rectangle, dont le coin inférieur gauche est aux coordonnées (*x-coor*, *y-coor*). Les dimensions du rectangle sont *height* et *width*. Le paramètre *mode* indique le type de *text*. Si *width* et *height* sont à zéro, le mode *mode* peut être "left" (gauche), "right" (droite) ou "center" (centré). *width* ou *height* sont différents pouvant prendre les valeurs de "justify" (justification) ou "fulljustify" (justification complète).

Si le paramètre *feature* vaut "blind", le texte n'est pas affiché.

pdf_show_boxed() retourne le nombre de caractères qui n'ont pas pu être traités, car ils ne rentraient pas dans le rectangle.

Voir aussi **pdf_show()** et **pdf_show_xy()**.

pdf_show_xy (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Affiche un texte à une position donnée.

```
void pdf_show_xy (resource pdf_object, string text, double x-coor, double y-coor)
```

pdf_show_xy() affiche le texte *text* à la position donnée par les coordonnées (*x-coor*, *y-coor*).

Voir aussi **pdf_show()** et **pdf_show_boxed()**.

pdf_skew (PHP 4 >= 4.0RC1)

Modifie le système de coordonnées.

```
void pdf_skew (resource pdf_object, double alpha, double beta)
```

pdf_skew() modifie le système de coordonnées, en faisant une rotation d'angle *alpha* pour les (x) et d'angle *beta* pour les (y), en degrés. *alpha* et *beta* ne peuvent pas pendre les valeurs de 90 ou 270 degrés.

pdf_stringwidth (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Retourne la largeur du texte avec la police courante.

```
double pdf_stringwidth (resource pdf_object, string text)
```

pdf_stringwidth() retourne la largeur du texte *text* avec la police courante. Il faut qu'une police ait été choisie auparavant.

Voir aussi **pdf_set_font()**.

pdf_stroke (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Dessine le long du chemin.

```
void pdf_stroke (resource pdf_object)
```

pdf_stroke() dessine une ligne le long du chemin. Le chemin courant est la somme de toutes les lignes déssinées. Sans cette fonction, la ligne de chemin ne sera pas dessinée.

Voir aussi **pdf_closepath()**, **pdf_closepath_stroke()**.

pdf_translate (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Modifie l'origine du système de coordonnées.

```
void pdf_translate (resource pdf_object, double x-coor, double y-coor)
```

pdf_translate() place l'origine du système de coordonnées au point (*x-coor*, *y-coor*). L'exemple suivant trace une ligne de (0, 0) à (200, 200) par rapport aux coordonnées initiales. Il faut aussi désigner le point courant après **pdf_translate()** et avant de commencer à dessiner les objets.

Exemple 1. Translation

```
<?php pdf_moveto($pdf, 0, 0);
pdf_lineto($pdf, 100, 100);
pdf_stroke($pdf);
pdf_translate($pdf, 100, 100);
pdf_moveto($pdf, 0, 0);
pdf_lineto($pdf, 100, 100);
pdf_stroke($pdf);
?>
```

pdf_open_memory_image (PHP 3>= 3.0.10, PHP 4)

Ouvre une image créée par les fonctions images PHP.

```
resource pdf_open_memory_image (resource pdf_object, resource image)
```

pdf_open_memory_image() prend comme argument une image créée avec les fonctions PHP, et la rend disponible pour le document PDF. La fonction retourne un identifiant PDF d'image.

Exemple 1. Inclusion d'une image mémoire

```
<?php
$im = imagecreate(100, 100);
$col = Imagecolorallocate($im, 80, 45, 190);
ImageFill($im, 10, 10, $col);
$pim = pdf_open_memory_image($pdf, $im);
ImageDestroy($im);
pdf_place_image($pdf, $pim, 100, 100, 1);
pdf_close_image($pdf, $pim);
?>
```

Voir aussi **pdf_close_image()**, **pdf_open_jpeg()**, **pdf_open_gif()** et **pdf_place_image()**.

LXIV. Verisign Payflow Pro Paiement

Cette extension vous permet d'effectuer des transactions avec des cartes de crédits en utilisant les services Verisign Payment Services, anciennement connu sous le nom de Signio (http://www.verisign.com/payment/).

Ces fonctions sont utilisables dès que PHP a été compilé avec l'option `--with-pfpro[=DIR]`. Vous devez aussi utiliser le SDK approprié sur votre plate-forme : il est disponible l'interface du manager (https://testmanager.signio.com/Downloads/Downloads_secure.htm), une fois que vous vous êtes inscrit. Si vous avez l'intention d'utiliser cette extension sur un serveur web SSL ou avec d'autres composants SSL (tels que l'extension CURL et SSL) vous DEVEZ utiliser le SDK beta.

Une fois que vous avez téléchargé le SDK vous devez copier les fichiers depuis le dossier `lib` de la distribution. Copier le fichier d'entête `pfpro.h` dans `/usr/local/include` et la librairie `libpfpro.so` dans `/usr/local/lib`.

Lorsque vous utilisez ces fonctions, vous pouvez omettre d'appeler les fonctions **pfpro_init()** et **pfpro_cleanup()** : l'extension se chargera de le faire automatiquement. Cependant, elles sont toujours disponibles au cas où vous auriez un grand nombre de transaction à traiter, ou que vous souhaiteriez un contrôle plus fin de la librairie. Vous pouvez effectuer autant de transaction que vous le souhaitez avec **pfpro_process()** lors d'une connexion.

Ces fonctions ont été ajoutée en PHP 4.0.2.

> **Note :** Ces fonctions ne font que fournir un accès aux services Verisign Payment Services. Assurez vous bien de lire le "Payflow Pro Developers Guide" pour plus de détails sur les parmètres.

pfpro_init (PHP 4 >= 4.0.2)

Initialise la librairie Payflow Pro

```
void pfpro_init ()
```

pfpro_init() initialise la librairie Payflow Pro library. Vous pouvez omettre cet appel : dans ce cas, elle sera appelée automatiquement **pfpro_init()** avant la première transaction.

Voir aussi **pfpro_cleanup()**.

pfpro_cleanup (PHP 4 >= 4.0.2)

Eteind la librairie Payflow Pro

```
void pfpro_cleanup ()
```

pfpro_cleanup() sert à terminer proprement une session avec la librairie Payflow Pro library. Elle doit être appelé après toutes les transactions, et avant la fin du script. Cependant, vous pouvez omettre cet appel : dans ce cas, cette fonction sera automatiquement appelée à la fin du script.

Voir aussi **pfpro_init()**.

pfpro_process (PHP 4 >= 4.0.2)

Effectue une transaction avec Payflow Pro

```
array pfpro_process (array parameters [, string address [, int port [, int timeout [,
string proxy address [, int proxy port [, string proxy logon [, string proxy
password]]]]]]])
```

Retourne un tableau associatif, contenant la réponse.

pfpro_process() effectue une transaction avec Payflow Pro. Le premier paramètre est un tableau associatif contenant des paires clés/valeurs, qui seront encodées, puis passées au serveur.

Le second paramètre *address* indique quel hôte contacter. Il est optinnel. Par défaut, il vaut "test.signio.com" : vous devrez probablement le remplacer par "connect.signio.com" pour effectuer de vraies transactions.

Le troisième paramètre *port* spécifie le port de connexion. Par défaut, c'est 443, le port SSL standard.

Le quatrième paramètre *timeout* indique le temps de timeout à utiliser. Par défaut, c'est 30 secondes. Notez que ce timeout ne prend effet que lorsqu'une connexion a été établie avec un serveur : votre script peut potentiellement attendre indéfiniement un événement DNS ou un problème de réseau.

Le cinquième paramètre *proxy address* indique le nom du proxy SSL. Le sixième paramètre *proxy port* indique le port à utiliser sur ce proxy.

Les septième et huitième paramètres, *proxy logon* et *proxy password* indique le nom de compte et le mot de passe à utiliser sur le proxy.

Cette fonction retourne un tableau associatif.

Note : Lisez attentivement le "Payflow Pro Developers Guide" pour connaître les détails des autres paramètres.

Exemple 1. Exemple Payflow Pro

```
<?php
pfpro_init();
$transaction = array(USER => 'monlogin',
     PWD => 'mmotdepasse',
     TRXTYPE => 'S',
     TENDER => 'C',
     AMT => 1.50,
     ACCT => '4111111111111111',
     EXPDATE => '0904'
     );
$response = pfpro_process($transaction);
if (!$response) {
  die("Impossible d'établir un lien avec Verisign.\n");
}
echo "La réponse de Verisign était ".$response[RESULT];
echo ", c'est à dire : ".$response[RESPMSG]."\n";
echo "\nLa requête de transaction: ";
print_r($transaction);
echo "\nLa réponse: ";
print_r($response);
pfpro_cleanup();
?>
```

pfpro_process_raw (PHP 4 >= 4.0.2)

Envoie une transaction brute à Payflow Pro

```
string pfpro_process_raw (string parameters [, string address [, int port [, int timeout [,
string proxy address [, int proxy port [, string proxy logon [, string proxy
password]]]]]]])
```

Retourne une chaîne avec une réponse.

pfpro_process_raw() envoie une transaction brute au serveur Payflow Pro. Il est vivement recommandé d'utiliser **pfpro_process()** à la place, car les règles de codage sont non standard.

Le premier argumentest une chaîne contenant la transaction brute. Tous les autres paramètres sont les mêmes que ceux de **pfpro_process()**. La valeur de retour est une chaîne contenant la réponse brute.

> **Note :** Lisez attentivement le "Payflow Pro Developers Guide" pour connaître tous les détails des paramètres et leur règle d'encodage. Il est recommandé d'utiliser plutôt **pfpro_process()**.

Exemple 1. Exemple pfpro_process_raw()

```
<?php
pfpro_init();
$response = pfpro_process("USER=mylogin&PWD[5]=m&ndy&TRXTYPE=S&TENDER=C&AMT=1.50&ACCT=411111111111
if (!$response) {
  die("Impossible de contacter Verisign.\n");
}
echo "La réponse brute de Verisign est ".$response;
pfpro_cleanup();
?>
```

pfpro_version (PHP 4 >= 4.0.2)

Lit le numéro de version de Payflow Pro

```
string pfpro_version ()
```

pfpro_version() lit la version de la librairie Payflow Pro. Au moment de rédaction de la doc, c'était L211.

LXV. Options PHP et informations

assert (PHP 4 >= 4.0b4)

Vérifie si une assertion est fausse

```
int assert (string|boolean assertion)
```

assert() va vérifier l'assertion *assertion* et prendre la mesure appropriée si le résultat est FALSE.

Si *assertion* est donnée sous la forme d'une chaîne, elle sera évaluée comme un code PHP par la fonction **assert()**. Les avantages de ce type d'assertion sont d'être moins lourd si la vérification d'assertion est désactivée, et le contenu des messages lorsque l'assertion échoue.

Il est recommandé de n'utiliser les assertions que comme outil de débuggage. Vous pouvez les utiliser pour les vérifications d'usage : ces conditions doivent normalement être vraies, et indiquer une erreur de programmation si ce n'est pas le cas. Vous pouvez aussi vérifier la présence de certaines extensions ou limitations du système.

Les assertions ne doivent pas être utilisées pour faire des opérations de vérifications en production, comme par exemple des vérifications de valeur d'argument. En conditions normales, votre code doit être en état de fonctionner si la vérification d'assertion est désactivée.

Le comportement de **assert()** peut être configuré par **assert_options()** ou par .ini-settings.

assert-options (PHP 4 >= 4.0b4)

Fixe et lit différentes options d'assertions

```
mixed assert_options (int what [, mixed value])
```

Avec **assert_options()** vous pouvez modifier les diverses options de la fonction **assert()**, ou simplement connaître la configuration actuelle.

Tableau 1. options d'assert

Option	Paramètre d'ini	Valeur par défaut	Description
ASSERT_ACTIVE	assert.active	1	active l'évaluation de la fonction **assert()**
ASSERT_WARNING	assert.warning	1	génère une alerte PHP pour chaque assertion fausse
ASSERT_BAIL	assert.bail	0	termine l'exécution en cas d'assertion fausse
ASSERT_QUIET_EVAL	assert.quiet_eval	0	inactive le rapport d'erreur durant l'évaluation d'une assertion
ASSERT_CALLBACK	assert_callback	(null)	fonction utilisateur de traitement des assertions fausses

assert_options() retourne la valeur originale de l'option, ou bien FALSE en cas d'erreur.

extension_loaded (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Détermine si une extension est chargée ou non.

```
boolean extension_loaded (string name)
```

extension_loaded() retourne TRUE si l'extension *name* a été chargée. Vous pouvez voir les différents noms des extensions, en utilisant la fonction **phpinfo()**.

Voir aussi **phpinfo()**.

Note : Cette fonction a été ajoutée dans 3.0.10.

dl (PHP 3, PHP 4 >= 4.0b1)

Charge une extension PHP à la volée

```
int dl (string library)
```

dl() charge l'extension PHP *library* à la volée . Voir aussi la directive de configuration extension_dir.

getenv (PHP 3, PHP 4 >= 4.0b1)

Retourne la valeur de la variable d'environnement.

```
string getenv (string varname)
```

getenv() retourne la valeur de la variable d'environnement *varname*, ou FALSE en cas d'erreur.

```
<?php
  $ip = getenv("REMOTE_ADDR"); // retourne l'adresse IP de l'utilisateur
?>
```

Vous pouvez voir une liste complète des variables d'environnement en utilisant la fonction **phpinfo()**. Vous pouvez trouver la signification de chacune d'entre elles en consultant le site concernant CGI specification (http://hoohoo.ncsa.uiuc.edu/cgi/) (en anglais>, et particulièment la page concernant les variables d'environnement. (http://hoohoo.ncsa.uiuc.edu/cgi/env.html).

Voir aussi **putenv()**.

get_cfg_var (PHP 3, PHP 4 >= 4.0b1)

Retourne la valeur d'une option de PHP

```
string get_cfg_var (string varname)
```

get_cfg_var() retourne la valeur courante de l'option PHP *varname*, ou bien FALSE en cas d'erreur.

get_cfg_var() ne retourne pas les options qui ont été choisies lors de la compilation de PHP, ni ne lit dans le fichier de configuration d'Apache.

Pour vérifier si le système utilise le fichier de configuration, essayez de lire la valeur de cfg_file_path. Si cette valeur est disponible, alors le fichier de configuration est utilisé.

get_current_user (PHP 3, PHP 4 >= 4.0b1)

Retourne le nom du possesseur du script courant.

```
string get_current_user (void)
```

get_current_user() retourne le nom du possesseur du script courant.

Voir aussi **getmyuid()**, **getmypid()**, **getmyinode()** et **getlastmod()**.

get_magic_quotes_gpc (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Retourne la configuration actuelle de l'option magic_quotes_gpc.

```
long get_magic_quotes_gpc (void)
```

get_magic_quotes_gpc() retourne la configuration actuelle de l'option magic_quotes_gpc (0 pour l'option désactivée, 1 pour l'option activée).

Voir aussi **get_magic_quotes_runtime()** et **set_magic_quotes_runtime()**.

get_magic_quotes_runtime (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Retourne la configuration actuelle de l'option magic_quotes_runtime.

```
long get_magic_quotes_runtime (void)
```

get_magic_quotes_runtime() retourne la configuration actuelle de l'option magic_quotes_runtime. (0 pour option désactivée, 1 pour option activée).

Voir aussi **get_magic_quotes_gpc()** et **set_magic_quotes_runtime()**.

getlastmod (PHP 3, PHP 4 >= 4.0b1)

Retourne la date de dernière modification de la page.

```
int getlastmod (void)
```

getlastmod() retourne la date de dernière modification de la page. La valeur retournée est un marqueur de temps UNIX, utilisable comme paramètre avec la fonction **date()**. **getlastmod()** retourne `FALSE` en cas d'erreur.

Exemple 1. Exemple avec getlastmod()

```
<?php
// affiche 'Dernière modification: March 04 1998 20:43:59.'
  echo "Dernière modification: ".date( "F d Y H:i:s.", getlastmod() );
?>
```

Voir aussi **date()**, **getmyuid()**, **get_current_user()**, **getmyinode()** et **getmypid()**.

getmyinode (PHP 3, PHP 4 >= 4.0b1)

Retourne l'inode du script.

```
int getmyinode (void)
```

getmyinode() retourne l'inode du script, ou FALSE en cas d'erreur.

Voir aussi **getmyuid()**, **get_current_user()**, **getmypid()** et **getlastmod()**.

Note : getmyinode() est inopérante sur les systèmes Windows.

getmypid (PHP 3, PHP 4 >= 4.0b1)

Retourne le numéro de processus courant.

```
int getmypid (void)
```

getmypid() retourne le numéro de processus actuel ou FALSE en cas d'erreur.

Il est à noter que si vous utilisez PHP comme module Apache, il n'est pas garanti que deux invocations distinctes de la fonction donnent des résultats différents.

Avertissement

Les identifiants de processus ne sont pas uniques, et forment une source d'entropie faible. Nous recommandons de ne pas utiliser les pid pour assurer la sécurité d'un système.

Voir aussi **getmyuid()**, **get_current_user()**, **getmyinode()** et **getlastmod()**.

getmyuid (PHP 3, PHP 4 >= 4.0b1)

Retourne l'UID du propriétaire du script actuel.

```
int getmyuid (void)
```

getmyuid() retourne l'UID du propriétaire du script actuel ou FALSE en cas d'erreur.

Voir aussi **getmypid()**, **get_current_user()**, **getmyinode()** et **getlastmod()**.

getrusage (PHP 3>= 3.0.7, PHP 4)

Retourne le niveau d'utilisation des ressources.

```
array getrusage ([int who])
```

getrusage() est une interface à la fonction system getrusage(2). Elle retourne un tableau associatif contenant les informations renvoyées par cet appel système. Si "who is 1", getrusage sera appelé avec le paramètre RUSAGE_CHILDREN.

Toutes les valeurs du tableau sont accessibles en utilisant leur nom dans le tableau.

Exemple 1. Exemple getrusage

```
<?php
  $dat = getrusage();
  echo $dat["ru_nswap"];        // Taille de la mémoire swap
  echo $dat["ru_majflt"];       // Nombre de page mémoires utilisées
  echo $dat["ru_utime.tv_sec"]; // Temps utilisateur (en secondes)
  echo $dat["ru_utime.tv_usec"];// Temps utilisateur (en microsecondes)
?>
```

Consultez le manuel "man" pour plus de détails.

ini_alter (PHP 4 >= 4.0b1)

Change la valeur d'une option de configuration

```
string ini_alter (string varname, string newvalue)
```

ini_alter() change la valeur de l'option de configuration *varname* et lui donne la valeur de *newvalue*. **ini_alter()** retourne FALSE en cas d'échec, et la valeur précédente en cas de succès.

> **Note : ini_alter()** est un alias de **ini_set()**

Voir aussi **ini_get()**, **ini_restore()** et **ini_set()**

ini_get (PHP 4 >= 4.0b1)

Lit la valeur d'une option de configuration.

```
string ini_get (string varname)
```

ini_get() retourne la valeur de l'option de configuration *varname* en cas de succès, et FALSE.

Voir aussi **ini_alter()**, **ini_restore()** et **ini_set()**

ini_restore (PHP 4 >= 4.0b1)

Restaure la valeur de l'option de configuration

```
string ini_restore (string varname)
```

ini_restore() restaure la valeur originale de l'option de configuration *varname*.

Voir aussi **ini_alter()**, **ini_get()** et **ini_set()**

ini_set (PHP 4 >= 4.0RC1)

Change la valeur d'une option de configuration

```
string ini_set (string varname, string newvalue)
```

ini_set() change la valeur de l'option de configuration *varname* et lui donne la valeur de *newvalue*. **ini_set()** retourne FALSE en cas d'échec, et la valeur précédente en cas de succès. La valeur de l'option de configuration sera modifiée durant toute l'exécution du script et pour ce script spécifiquement. Elle reprendra sa valeur par défaut dès la fin du script.

Toutes les options disponibles ne peuvent pas être toutes modifiées avec **ini_set()**. Ci-dessous, vous trouverez une liste de toutes les options (disponibles en PHP 4.0.5-dev), et si elles peuvent être modifiées.

Tableau 1. Options de configuration

Nom	Par défaut	Modifiable
define_syslog_variables	"0"	PHP_INI_PERDIR|PHP_INI_SYSTEM
highlight.bg	HL_BG_COLOR	PHP_INI_ALL
highlight.comment	HL_COMMENT_COLOR	PHP_INI_ALL
highlight.default	HL_DEFAULT_COLOR	PHP_INI_ALL
highlight.html	HL_HTML_COLOR	PHP_INI_ALL
highlight.keyword	HL_KEYWORD_COLOR	PHP_INI_ALL
highlight.string	HL_STRING_COLOR	PHP_INI_ALL
allow_call_time_pass_reference	"1"	PHP_INI_SYSTEM|PHP_INI_PERDIR
asp_tags	"0"	PHP_INI_SYSTEM|PHP_INI_PERDIR
display_errors	"1"	PHP_INI_ALL
display_startup_errors	"0"	PHP_INI_ALL
enable_dl	"1"	PHP_INI_SYSTEM
error_append_string	NULL	PHP_INI_ALL
error_prepend_string	NULL	PHP_INI_ALL
expose_php	"1"	PHP_INI_SYSTEM
html_errors	"1"	PHP_INI_SYSTEM
ignore_user_abort	"0"	PHP_INI_ALL
implicit_flush	"0"	PHP_INI_PERDIR|PHP_INI_SYSTEM
log_errors	"0"	PHP_INI_ALL
magic_quotes_gpc	"1"	PHP_INI_ALL
magic_quotes_runtime	"0"	PHP_INI_ALL
magic_quotes_sybase	"0"	PHP_INI_ALL
output_buffering	"0"	PHP_INI_PERDIR|PHP_INI_SYSTEM
output_handler	NULL	PHP_INI_PERDIR|PHP_INI_SYSTEM
register_argc_argv	"1"	PHP_INI_ALL
register_globals	"1"	PHP_INI_ALL
safe_mode	"0"	PHP_INI_SYSTEM
short_open_tag	"1"	PHP_INI_SYSTEM|PHP_INI_PERDIR
sql.safe_mode	"0"	PHP_INI_SYSTEM
track_errors	"0"	PHP_INI_ALL
y2k_compliance	"0"	PHP_INI_ALL
arg_separator	"&"	PHP_INI_ALL
auto_append_file	NULL	PHP_INI_ALL
auto_prepend_file	NULL	PHP_INI_ALL
doc_root	NULL	PHP_INI_SYSTEM
default_charset	SAPI_DEFAULT_CHARSET	PHP_INI_ALL
default_mimetype	SAPI_DEFAULT_MIMETYPE	PHP_INI_ALL
error_log	NULL	PHP_INI_ALL

Nom	Par défaut	Modifiable
extension_dir	PHP_EXTENSION_DIR	PHP_INI_SYSTEM
gpc_order	"GPC"	PHP_INI_ALL
include_path	PHP_INCLUDE_PATH	PHP_INI_ALL
max_execution_time	"30"	PHP_INI_ALL
open_basedir	NULL	PHP_INI_SYSTEM
safe_mode_exec_dir	"1"	PHP_INI_SYSTEM
upload_max_filesize	"2M"	PHP_INI_ALL
file_uploads	"1"	PHP_INI_ALL
post_max_size	"8M"	PHP_INI_SYSTEM
upload_tmp_dir	NULL	PHP_INI_SYSTEM
user_dir	NULL	PHP_INI_SYSTEM
variables_order	NULL	PHP_INI_ALL
SMTP	"localhost"	PHP_INI_ALL
browscap	NULL	PHP_INI_SYSTEM
error_reporting	NULL	PHP_INI_ALL
memory_limit	"8M"	PHP_INI_ALL
precision	"14"	PHP_INI_ALL
sendmail_from	NULL	PHP_INI_ALL
sendmail_path	DEFAULT_SENDMAIL_PATH	PHP_INI_SYSTEM
disable_functions	""	PHP_INI_SYSTEM
allow_url_fopen	"1"	PHP_INI_ALL

Tableau 2. Définition des constantes PHP_INI_*

Constante	Valeur	Signification
PHP_INI_USER	1	La valeur peut être modifiée dans un script
PHP_INI_PERDIR	2	La valeur peut être modifiée dans le fichier `.htaccess`
PHP_INI_SYSTEM	4	La valeur peut être modifiée dans `php.ini` ou `httpd.conf`
PHP_INI_ALL	7	La valeur peut être modifiée n'importe où

Voir aussi **ini_alter()**, **ini_get()** et **ini_restore()**

phpcredits (PHP 4 >= 4.0b1)

Imprime les crédits de PHP.

```
void phpcredits (int flag)
```

phpcredits() affiche la liste des développeurs PHP, des modules, etc... Elle génère le code HTML approprié pour insérer les informations dans une page. Le paramètre *flag* indique les informations qui doivent être affichées. Par exemple, pour afficher les crédits généraux, vous pouvez utiliser le code suivant :

```
<?php
  phpcredits(CREDITS_GENERAL);
?>
```

Et pour afficher la liste des développeurs et du groupe de documentation dans une page séparée, vous utiliserez

```
<?php
  phpcredits(CREDITS_GROUP + CREDITS_DOCS + CREDITS_FULLPAGE);
?>
```

Si vous vous sentez l'envie de placer tous les crédits dans votre page, vous pouvez utiliser ceci :

```
<html>
  <head>
    <title>Ma page de crédits</title>
  </head>
  <body>
  <?php
    // Un peu de votre code
    phpcredits(CREDITS_ALL + CREDITS_FULLPAGE);
    // Un autre peu de votre code
  ?>
  </body>
</html>
```

Tableau 1. Paramètre prédéfinis de phpcredits()

Nom	Description
CREDITS_ALL	Tous les crédits, équivalent à : CREDITS_DOCS + CREDITS_GENERAL + CREDITS_GROUP + CREDITS_MODULES + CREDITS_FULLPAGE. La fonction génère alors une page HTML complète.
CREDITS_DOCS	Les crédits du groupe de documentation
CREDITS_FULLPAGE	En général, ce paramètre est utilisé avec d'autres constantes. Il indique que la page ainsi générée doit être une page HTML complète, avec toutes les balises nécessaires.
CREDITS_GENERAL	Crédits Généraux : conception et design du langage, auteurs de PHP 4.0, module SAPI.
CREDITS_GROUP	Une liste des développeurs principaux
CREDITS_MODULES	Une liste des extensions de PHP, et leurs auteurs
CREDITS_SAPI	Cette constante est définie, mais elle n'est toujours pas utilisée sous PHP 4.0.1pl2.

Voir aussi **phpinfo()**, **phpversion()** et **php_logo_guid()**.

phpinfo (PHP 3, PHP 4 >= 4.0b1)

Affiche de nombreuses informations sur le PHP.

```
int phpinfo ([int what])
```

phpinfo() affiche de nombreuses informations sur le PHP, concernant sa configuration courante : options de compilation, extensions, version, informations sur le serveur, et environnement (lorsque compilé comme module), environnement PHP, chemins, utilisateur, en-têtes HTTP, et licence GNU Public License.

Les affichages peuvent être personnalisés en passant une ou plusieurs valeurs parmi les suivantes, comme paramètre optionnel *what* :

- INFO_GENERAL
- INFO_CREDITS
- INFO_CONFIGURATION
- INFO_MODULES
- INFO_ENVIRONMENT
- INFO_VARIABLES
- INFO_LICENSE
- INFO_ALL

Voir aussi **phpversion()**, **phpcredits()** et **php_logo_guid()**

phpversion (PHP 3, PHP 4 >= 4.0b1)

Retourne le numéro de la version courante de PHP.

```
string phpversion (void)
```

phpversion() retourne le numéro de la version courante de PHP.

Exemple 1. Exemple avec phpversion()

```
<?php
// affiche le numéro de version courante du PHP.
  echo "PHP Version: ".phpversion();
?>
```

Voir aussi **phpinfo()**.

php_logo_guid (PHP 4 >= 4.0b4)

Retourne le logo

```
string php_logo_guid (void)
```

Note : Cette fonctionnalité a été ajoutée dans PHP 4 Beta 4.

php_sapi_name (PHP 4 >= 4.0.1)

Retourne le type d'interface utilisé entre le serveur web et PHP

```
string php_sapi_name (void)
```

php_sapi_name() retourne une chaîne en minuscule qui décrit le type d'interface utilisé en le serveur web et PHP (Server API, SAPI). En CGI PHP, cette chaîne est "CGI", en mod_php pour Apache, cette chaîne est "apache", etc...

Exemple 1. Exemple php_sapi_name()

```
<?php
$inter_type = php_sapi_name();
if ($inter_type == "cgi")
    print "Vous utilisez CGI PHP\n";
else
    print "Vous n'utilisez pas CGI PHP\n";
?>
```

php_uname (PHP 4 >= 4.0.2)

Retourne les informations sur le système d'exploitation

```
string php_uname (void)
```

php_uname() retourne les informations sur le système d'exploitation sur lequel tourne PHP.

Exemple 1. Exemple php_uname()

```
<?php
if (substr(php_uname(), 0, 7) == "Windows") {
    die("Désolé, ce script ne fonctionne pas sous Windows.\n");
}
?>
```

putenv (PHP 3, PHP 4 >= 4.0b1)

Fixe la valeur d'une variable d'environnement.

```
void putenv (string setting)
```

putenv() fixe la valeur d'une variable d'environnement. Cette valeur n'existera que durant la vie du script courant, et l'environnement initial sera restauré lorsque le script sera terminé.

Modifier la valeur de certaines variables système peut être un trou de sécurité considérable. La directive de configuration `safe_mode_allowed_env_vars` contient une liste de préfixes, séparés par des virgules. Lorsque le Safe Mode est actif, l'utilisateur ne peut que modifier les variables qui dont le nom commence par les préfixes fournis par cette directive. Par défaut, les utilisateurs ne peuvent modifier que les variables qui commencent par `PHP_` (i.e. `PHP_FOO=BAR`). Note: si cette directive est vide, PHP autorisera la modification de TOUTES les variables d'environnement!.

La directive de configuration `safe_mode_protected_env_vars` contient une liste de variables d'environnement, séparées par des virgules. Les utilisateurs ne pourront pas modifier ces variables avec la fonction **putenv()**. Ces variables seront protégées même si `safe_mode_allowed_env_vars` permet leur modification.

Exemple 1. Modification d'une variable d'environnement

```
<?php
putenv("UNIQID=$uniqid");
?>
```

Voir aussi **getenv()**.

set_magic_quotes_runtime (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Active/désactive l'option magic_quotes_runtime.

```
long set_magic_quotes_runtime (int new_setting)
```

set_magic_quotes_runtime() active/désactive l'option magic_quotes_runtime. (0 l'option est désactivée, 1 l'option est activée).

Voir aussi **get_magic_quotes_gpc()** et **get_magic_quotes_runtime()**.

set_time_limit (PHP 3, PHP 4 >= 4.0b1)

Fixe le temps maximum d'exécution d'un script.

```
void set_time_limit (int seconds)
```

set_time_limit() fixe le délai d'expiration d'un script, en secondes. Si cette limite est atteinte, le script s'interrompt, et renvoie une erreur fatale. La valeur par défaut est 30 secondes ou, si c'est le cas, la valeur de la directive max_execution_time définie dans le fichier de configuration. Si la valeur est zéro, il n'y a alors aucune limite imposée.

Lorsqu'elle est appelée, la fonction **set_time_limit()** remet le compteur de zéro. En d'autres termes, si la limite par défaut est à 30 secondes, et qu'après 25 secondes d'exécution du script l'appel set_time_limit(20) est fait, alors le script tournera pendant un total de 45 secondes avant de finir.

Notez que **set_time_limit()** n'a pas d'effet lorsque PHP fonctionne en mode safe mode. Il n'y a pas d'autre solution que de changer de mode, ou de modifier la durée maximale d'exécution dans le fichier de configuration.

zend_logo_guid (PHP 4 >= 4.0b4)

Retourne le logo de Zend

```
string zend_logo_guid (void)
```

Note : Cette fonctionnalité a été ajoutée en PHP 4 Beta 4.

get_defined_constants (PHP 4 CVS only)

Retourne la liste des constantes et leur valeur

```
array get_defined_constants ()
```

get_defined_constants() retourne les noms et valeurs des constantes déjà définies. Cela inclus les constantes créées par les extensions, et celles créées avec la fonction **define()**.

Par exemple :

```
<?php
print_r (get_defined_constants());
?>
```

affichera

```
Array
(
    [E_ERROR] => 1
    [E_WARNING] => 2
    [E_PARSE] => 4
    [E_NOTICE] => 8
    [E_CORE_ERROR] => 16
    [E_CORE_WARNING] => 32
    [E_COMPILE_ERROR] => 64
    [E_COMPILE_WARNING] => 128
    [E_USER_ERROR] => 256
    [E_USER_WARNING] => 512
    [E_USER_NOTICE] => 1024
    [E_ALL] => 2047
    [TRUE] => 1
)
```

Voir aussi **get_loaded_extensions()**.

get_loaded_extensions (PHP 4 >= 4.0b4)

Retourne la liste de tous les modules compilés et chargés

```
array get_loaded_extensions (void)
```

get_loaded_extensions() retourne un tableau contenant les noms de tous les modules compilés et chargés sur l'interpreteur PHP courant.

Par exemple, la ligne ci-dessous

```
<?php
print_r(get_loaded_extensions());
?>
```

affichera la liste suivante :

```
Array
(
    [0] => xml
```

```
    [1] => wddx
    [2] => standard
      [3] => session
      [4] => posix
      [5] => pgsql
      [6] => pcre
      [7] => gd
      [8] => ftp
      [9] => db
      [10] => Calendar
      [11] => bcmath
)
```

Voir aussi **get_extension_funcs()**.

get_extension_funcs (PHP 4 >= 4.0b4)

Liste les fonctions d'une extension

```
array get_extension_funcs (string module_name)
```

get_extension_funcs() retourne le nom des fonctions définies dans le module *module_name*.

Par exemple, les lignes suivantes :

```
<?php
print_r(get_extension_funcs("xml"));
print_r(get_extension_funcs("gd"));
?>
```

vont afficher la liste des fonctions disponibles avec les modules `xml` et `gd`.

Voir aussi **get_loaded_extensions()**.

get_required_files (PHP 4 >= 4.0RC2)

Retourne un tableau avec les noms des fichiers qui sont requis et inclus dans un script

```
array get_required_files (void)
```

get_required_files() retourne un tableau contenant les noms de tous les fichiers qui ont été chargés dans un script avec la fonction **require_once()** ou **include_once()**. Les index de ces tableaux sont les noms des fichiers utilisés dans les fonctions **require_once()** ou **include_once()**.

> **Note :** En PHP 4.0.1pl2, cette fonction supposait que `required_once` utilisait l'extension ".php" : les autres extensions ne fonctionnaient pas. Par ailleurs, dans cette version, le tableau retourné était un tableau associatif, et cette fonction n'était pas un alias de **get_included_files()**
>
> Depuis PHP 4.0.4, cette fonction est un alias de **get_included_files()**

Voir aussi **require_once()**, **include_once()** et **get_included_files()**

get_included_files (PHP 4 >= 4.0RC1)

Retourne un tableau avec les noms des fichiers qui sont inclus dans un script

```
array get_included_files (void)
```

get_included_files() retourne un tableau contenant les noms de tous les fichiers qui ont été ajoutés au script avec les fonctions **require_once()** ou **include_once()**.

L'exemple ci-dessous

Exemple 1. Affichage des fichiers inclus et requis

```
<?php
require_once("local.php");
require_once("../inc/global.php");
for ($i=1; $i<5; $i++)
  include "util".$i.".php";
  echo "Fichiers appelés avec required_once/included_once\n";
  print_r(get_required_files());
?>
```

va afficher :

```
Fichiers appelés avec required_once/included_once
Array
(
  [0] => local.php
  [1] => /full/path/to/inc/global.php
  [2] => util1.php
  [3] => util2.php
  [4] => util3.php
  [5] => util4.php
)
```

> **Note :** En PHP 4.0.1pl2, cette fonction supposait que `required_once` utilisait l'extension ".php" : les autres extensions ne fonctionnaient pas. Par ailleurs, dans cette version, le tableau retourné était un tableau associatif, et cette fonction n'était pas un alias de **get_included_files()**

Voir aussi **require_once()**, **include_once()** et **get_required_files()**.

zend_version (PHP 4 >= 4.0b4)

Lit la version courante du moteur Zend.

```
string zend_version (void)
```

zend_version() retourne une chaîne contenant le numéro de version du moteur d'analyse Zend, pour l'exécutable PHP courant.

Exemple 1. Exemple zend_version()

```
<?php
// affiche e.g. ’Version du moteur Zend: 1.0.4’
// ou bien quelque chose d’approchant si votre version de PHP date un peu
  echo "Version du moteur Zend: ".zend_version();
?>
```

Voir aussi **phpinfo()**, **phpcredits()**, **php_logo_guid()** et **phpversion()**.

LXVI. POSIX

Ce module contient une interface avec les documents au standard IEEE 1003.1 (POSIX.1), qui ne sont pas accessibles autrement. Par exemple, POSIX.1 définit les fonctions open(), read(), write() et close(), qui ont été traditionnellement les fonctions de PHP 3. Certains fonctionnalités spécifiques ne sont pas encore disponibles, bien que ce module tâche de remédier à cette situation avec ses fonctions.

posix_kill (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Envoie un signal à un process.

```
bool posix_kill (int pid, int sig)
```

posix_kill() envoie le signal *sig* au processus *pid*. **posix_kill()** retourne FALSE, s'il n'a pas pu envoyer le signal, et TRUE sinon.

Reportez vous à la page de manuel de kill(2) de votre système POSIX, qui contient plus de détails sur les identifiants négatifs de processus, les pid spéciaux 0 et -1, et le signal numéro 0.

posix_getpid (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Retourne l'identifiant du processus courant.

```
int posix_getpid (void )
```

posix_getpid() retourne l'identifiant du processus courant.

posix_getppid (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Retourne l'identifiant du processus parent.

```
int posix_getppid (void )
```

posix_getppid() retourne l'identifiant du processus parent du processus courant.

posix_getuid (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Retourne l'ID de l'utilisateur du processus courant.

```
int posix_getuid (void )
```

posix_getuid() retourne l'ID numérique de l'utilisateur du processus courant. Reportez vous à **posix_getpwuid()** pour accéder au nom d'utilisateur.

posix_geteuid (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Retourne l'UID effectif de l'utilisateur du processus courant.

```
int posix_geteuid (void )
```

posix_geteuid() retourne l'UID effectif de l'utilisateur du processus courant. Reportez vous à **posix_getpwuid()** pour obtenir le nom d'utilisateur.

posix_getgid (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Retourne l'UID du groupe du processus courant.

```
int posix_getgid (void )
```

posix_getgid() retourne l'UID du groupe du processus courant. Reportez vous à **posix_getgrgid()** pour accéder au nom du groupe.

posix_getegid (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Retourne l'ID effectif du groupe du processus courant.

```
int posix_getegid (void )
```

posix_getegid() retourne l'ID effectif du groupe du processus courant. Reportez vous à **posix_getgrgid()** pour transformer cette information en nom de groupe.

posix_setuid (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Fixe l'UID effective du processus courant.

```
bool posix_setuid (int uid)
```

posix_setuid() fixe l'UID effective de l'utilisateur du processus courant. Vous devez avoir les privilèges nécessaires (traditionnellement ceux du root) sur votre système pour faire ceci.

posix_setuid() retourne TRUE en cas de succès, FALSE sinon. Voir aussi **posix_setgid()**.

posix_setgid (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Fixe le GID effective du processus courant.

```
bool posix_setgid (int gid)
```

posix_setgid() fixe le GID effective du processus courant. Reportez vous à **posix_getgrgid()** pour transformer cette information en nom de groupe. L'ordre approprié est d'abord **posix_setgid()**, puis **posix_setuid()**.

posix_setgid() retourne TRUE en cas de succès, FALSE sinon.

posix_getgroups (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Retourne les identifiants du groupe du processus courant.

```
array posix_getgroups (void )
```

posix_getgroups() retourne un tableau contenant les identifiants du groupe du processus courant. Reportez vous à **posix_getgrgid()** pour pouvoir utiliser ces id.

posix_getlogin (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Retourne le nom de login.

```
string posix_getlogin (void )
```

posix_getlogin() retourne le nom de login de l'utilisateur qui possède le processus courant. Reportez vous à **posix_getpwnam()** pour obtenir plus d'informations sur cet utilisateur.

posix_getpgrp (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Retourne l'identifiant du groupe de processus.

```
int posix_getpgrp (void )
```

posix_getpgrp() retourne l'identifiant du groupe de processus du processus courant. Reportez vous à POSIX.1 et à getpgrp(2) dans le manuel de votre système POSIX pour plus d'informations sur les groupes de processus.

posix_setsid (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Fait du processus courant un chef de session.

```
int posix_setsid (void )
```

posix_setsid() fait du processus courant un chef de session. Reportez vous à POSIX.1 et setsid(2) dans le manuel de votre système POSIX pour plus d'informations sur le contrôle de tâche. **posix_setsid()** retourne un identifiant de session.

posix_setpgid (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Fixe l'identifiant de group de processus.

```
int posix_setpgid (int pid, int pgid)
```

posix_setpgid() ajoute le processus *pid* au groupe d'id *pgid*. Reportez vous à POSIX.1 et setsid(2) dans le manuel de votre système POSIX pour plus d'informations sur le contrôle de tâche. Retourne TRUE en cas de succès, et FALSE sinon.

posix_getpgid (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Retourne l'id du groupe de processus.

```
int posix_getpgid (int pid)
```

posix_getpgid() retourne l'id du groupe de processus pour le processus *pid*.

Ceci n'est pas une fonction POSIX, mais elle est répandu sur les systèmes BSD et System V. Si votre système ne supporte pas cette fonction, la fonction PHP retournera toujours FALSE.

posix_getsid (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Retourne le sid du processus.

```
int posix_getsid (int pid)
```

posix_getsid() retourne le sid du processus *pid*. Si *pid* est à 0, le sid retourné sera celui du processus courant.

Ceci n'est pas une fonction POSIX, mais elle est répandu sur les systèmes BSD et System V. Si votre système ne supporte pas cette fonction, la fonction PHP retournera toujours FALSE.

posix_uname (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Retourne le nom du système.

```
array posix_uname (void )
```

posix_uname() retourne un tableau associatif avec des informations sur le système. Les indices du tableau sont :

- sysname - nom du système d'exploitation (e.g. Linux)
- nodename - nom du système (e.g. valiant)
- release - édition du système d'exploitation (e.g. 2.2.10)
- version - version du système d'exploitation (e.g. #4 Tue Jul 20 17:01:36 MEST 1999)
- machine - architecture système (e.g. i586)

Posix impose que vous n'ayez pas d'a priori sur le format des chaînes, c'est à dire que vous ne devez pas vous attendre à avoir forcément 3 chiffres pour la version, par exemple.

posix_times (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Utilisation des ressources.

```
array posix_times (void )
```

posix_times() retourne un tableau avec les informations sur l'utilisation du CPU. Les indices sont :

- ticks - nombre de ticks depuis le dernier démarrage
- utime - temps utilisateur utilisé par le processus courant.
- stime - temps système utilisé par le processus courant.
- cutime - temps utilisateur utilisé par le processus courant et ses enfants.
- cstime - temps système utilisé par le processus courant et ses enfants.

posix_ctermid (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Retourne le chemin du terminal.

```
string posix_ctermid (void )
```

Encore à faire.

posix_ttyname (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Retourne le nom de device du terminal.

```
string posix_ttyname (int fd)
```

Encore à faire.

posix_isatty (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Détermine si un pointeur de fichier est un terminal interactif.

```
bool posix_isatty (int fd)
```

Encore à faire.

posix_getcwd (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Chemin du dossier courant.

```
string posix_getcwd (void )
```

Encore à faire très rapidement.

posix_mkfifo (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Crée un fichier fifo (first in, first out) (un pipe nommé).

```
bool posix_mkfifo (string pathname, int mode)
```

Encore à faire très rapidement.

posix_getgrnam (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Retourne des informations sur un groupe.

```
array posix_getgrnam (string name)
```

Encore à faire.

posix_getgrgid (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Retourne des informations sur un groupe.

```
array posix_getgrgid (int gid)
```

Encore à faire.

posix_getpwnam (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Retourne des informations sur un utilisateur.

```
array posix_getpwnam (string username)
```

posix_getpwnam() retourne un tableau associatif qui contient des informations à propos d'un utilisateur, identifié par son nom, passé en paramètre *username*.

Les éléments du tableau sont :

Tableau 1. Le tableau descriptif d'un utilisateur

Elément	Déscription
name	Le nom contient le nom de l'utilisateur. Généralement, c'est un nom court, de moins de 16 caractères, mais ce n'est pas son nom réel et complet. Cette valeur devrait correspondre au paramètre *username*, et donc, il est redondant.
passwd	Contient le mot de passe de l'utilisateur, encrypté. Souvent, dans les systèmes utilisant les mots de passe "fantômes", un astérisque est retourné.
uid	L'UID de l'utilisateur.
gid	L'ID du groupe de l'utilisateur. Utilisez la fonction **posix_getgrgid()** pour connaitre le nom du groupe, et ses membres.
gecos	GECOS est un terme obsolète qui fait référence aux données de finger, sur un système Honeywell. Le champs, cependant, a survécu, et son contenu a été formalisé par POSIX. Le champs contient une liste, séparée par des virgules, qui contient le nom complet de l'utilisateur, son téléphone professionnel, son numéro de téléphone bureau, et son numéro de téléphone personnel. Sur la plus part des sytèmes, seul le nom est disponible.
dir	Cet élément contient le chemin absolu jusqu'au dossier racine de l'utilisateur.
shell	Cet élément contient le chemin absolu jusqu'au dossier d'éxécution du shell de l'utilisateur.

posix_getpwuid (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Retourne des informations sur un utilisateur.

```
array posix_getpwuid (int uid)
```

posix_getpwuid() retourne un tableau associatif contenant des informations sur un utilisateur repéré par son UID, passé dans le paramètre *`uid`*.

Les éléments du tableau sont :

Tableau 1. Le tableau de description d'un utilisateur

Elément	Description
name	Le nom contient le nom de l'utilisateur. Généralement, c'est un nom court, de moins de 16 caractères, mais ce n'est pas son nom réel et complet.
passwd	Contient le mot de passe de l'utilisateur, encrypté. Souvent, dans les systèmes utilisant les mots de passes "fantômes", un astérisque est retourné.
uid	Cette valeur devrait correspondre au paramètre *`uid`*, et donc, il est redondant.
gid	L'ID du groupe de l'utilisateur. Utilisez la fonction **posix_getgrgid()** pour connaître le nom du groupe, et ses membres.
gecos	GECOS est un terme obsolète qui fait référence aux données de finger, sur un système Honeywell. Le champs, cependant, a survécu, et son contenu a été formalisé par POSIX. Le champs contient une liste, séparée par des virgules, qui contient le nom complet de l'utilisateur, son téléphone professionne, son numéro de bureau, et son numéro de téléphone personnel. Sur la plus part des sytèmes, seul le nom est disponible.
dir	Cet élément contient le chemin absolu jusqu'au dossier racine de l'utilisateur.
shell	Cet élément contient le chemin absolu jusqu'au dossier d'éxécution du shell de l'utilisateur.

posix_getrlimit (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Retourne les limites système.

```
array posix_getrlimit (void )
```

Encore à faire rapidement.

LXVII. PostgreSQL

Postgres, initialement développé au département de Science informatique, à UC Berkeley, mis en place la majorité des concepts des bases relationnelles, actuellement disponibles sur le marché. PostgreSQL accepte le langage SQL92/SQL3, assure l'intégrité transactionnelle, et l'extension de type. PostgreSQL est une évolution du code originale de Berkeley : il est Open Source et dans le domaine public.

PostgreSQL est disponible sans frais. La version actuelle est disponible à (en anglais) : www.PostgreSQL.org (http://www.postgresql.org/).

Depuis la version 6.3 (03/02/1998) PostgreSQL utilise les sockets UNIX, et une table est dédiée à ces nouvelles capacités. La socket est située dans le dossier `/tmp/.s.PGSQL.5432`. Cette option peut être activée avec '-i' passé au **postmaster** et cela s'interprète: "écoute sur les sockets TCP/IP et sur les sockets Unix".

Tableau 1. Postmaster et PHP

Postmaster	PHP	Statut
postmaster &	pg_connect("dbname=MonDbName");	OK
postmaster -i &	pg_connect("dbname=MonDbName");	OK
postmaster &	pg_connect("host=localhost dbname=MonDbName");	Unable to connect to PostgreSQL server: connectDB() failed: Impossible de se connecter au serveur PostgreSQL: connectDB() a échoué. Est ce que le postmaster fonctionne, et accepte les TCP/IP (option -i) sur le port '5432'?
postmaster -i &	pg_connect("host=localhost dbname=MonDbName");	OK

Il est possible de se connecter avec la commande suivante : **$conn = pg_Connect("host=monHote port=monPort tty=monTTY options=myOptions dbname=myDB user=myUser password=myPassword");**

L'ancienne syntaxe : **$conn = pg_connect("host", "port", "options", "tty", "dbname")** est obsolète.

Pour utiliser l'interface des grands objets (large object (lo) interface), il est nécessaire de les placer dans un bloc de transaction. Un bloc de transaction commence avec **begin** et si la transaction se termine avec un **commit** et **end**. Si la transaction échoue, elle doit être conclue par un **abort** et **rollback**.

Exemple 1. Utilisation des objets de grande taille (Large Objects)

```
<?php
$database = pg_connect("", "", "", "", "jacarta");
pg_exec($database, "begin");
    $oid = pg_locreate($database);
    echo "$oid\n";
    $handle = pg_loopen($database, $oid, "w");
    echo "$handle\n";
    pg_lowrite($handle, "gaga");
    pg_loclose($handle);
    pg_exec($database, "commit")
    pg_exec($database, "end")
?>
```

pg_Close (PHP 3, PHP 4 >= 4.0b1)

Termine une connexion PostgreSQL.

```
bool pg_close (resource connection)
```

pg_close() retourne FALSE si l'index de connexion n'est pas valable, et TRUE sinon. **pg_close()** ferme la connexion au serveur PostgreSQL associé à *connection*.

> **Note :** Il n'est généralement pas nécessaire de fermer une connexion non persistante, car elles sont automatiquement fermées à la fin d'un script.

pg_close() ne ferme pas les connexions persistantes ouvertes avec **pg_pconnect()**.

pg_cmdTuples (PHP 3, PHP 4 >= 4.0b1)

Retourne le nombre de tuples affectés.

```
int pg_cmdtuples (resource result_id)
```

pg_cmdtuples() retourne le nombre de tuples (instances) affectés par les requêtes INSERT, UPDATE, et DELETE. Si aucun tuple n'a été affecté, la fonction retournera 0.

Exemple 1. pg_cmdtuples

```
<?php
$result = pg_exec($conn, "INSERT INTO verlag VALUES ('Auteur')");
$cmdtuples = pg_cmdtuples($result);
echo $cmdtuples . " <- tuples modifiés.";
?>
```

pg_Connect (PHP 3, PHP 4 >= 4.0b1)

Ouvre une connexion.

```
resource pg_connect (string conn_string)
```

conn_string retourne un index de connexion en cas de succès, et FALSE sinon. Ouvre une connexion à un serveur PostgreSQL. Les arguments doivent être placé entre guillemets.

Exemple 1. Using pg_connect arguments

```
<?php
$dbconn = pg_connect("dbname=marie");
//connexion à une base de données nommée "marie"
$dbconn2 = pg_connect("host=localhost port=5432 dbname=marie");
//connexion à une base de données nommée "marie" sur l'hôte "localhost" sur le port "5432"
$dbconn3 = pg_Connect ("host=sheep port=5432 dbname=marie user=mouton password=baaaa");
//connection à une base de données nommée "marie" sur le serveur "mouton" avec
// un nom d'utilisateur et le mot de passe associé
?>
```

Les arguments disponbiles comptent notamment *dbname* *port*, *host*, *tty*, *options*, *user*, et *password*

pg_connect() retourne un index de connexion qui sera nécessaire aux autres fonctions PostgreSQL. Vous pouvez ouvrir plusieurs connexions simultanées.

L'ancienne syntaxe **$conn = pg_connect("host", "port", "options", "tty", "dbname")** est obsolète.

Voir aussi **pg_pconnect()**.

pg_DBname (PHP 3, PHP 4 >= 4.0b1)

Nom de la base de données.

```
string pg_dbname (resource connection)
```

pg_dbname() retourne le nom de la base de données PostgreSQL associée à l'index de connexion connection, ou FALSE si connection n'est pas valide.

pg_end_copy (PHP 4 >= 4.0.3)

Synchronise avec le serveur PostgreSQL

```
bool pg_end_copy ([resource connection])
```

pg_end_copy() synchronise le client PostgreSQL (ici PHP) avec le serveur, après une opération de copie. Il faut utiliser cette fonction, sous peine de recevoir une erreur "out of sync" (désynchronisé). Retourne TRUE en cas de succès, et FALSE sinon.

Pour plus de détails et un exemple voyez : **pg_put_line()**.

pg_ErrorMessage (PHP 3, PHP 4 >= 4.0b1)

Message d'erreur.

```
string pg_errormessage (resource connection)
```

pg_errormessage() retourne une chaîne contenant le dernier message d'erreur, ou FALSE en cas d'échec. Il sera impossible d'obtenir des détails sur l'erreur générée, en utilisant la fonction **pg_errormessage()** si une erreur est survenue dans la dernière action pour laquelle une connexion valide existe. **pg_errormessage()** retournera une chaîne contenant le message d'erreur généré par le serveur final.

pg_Exec (PHP 3, PHP 4 >= 4.0b1)

Exécute une requête.

```
resource pg_exec (resource connection, string query)
```

pg_exec() retourne un index de résultat, si la requête a été correctement exécutée, et FALSE en cas d'échec, ou si la connexion connection n'était pas un index de connexion valide. En cas d'erreur, le message d'erreur peut être obtenu grâce à la fonction **pg_errormessage()**, si l'index de connexion était valide. Envoie une requête à un serveur PostgreSQL identifié grâce à l'index de connexion. La réponse retournée par cette fonction est un index de résultat qui devra être utilisé pour accéder aux lignes de résultat, grâce à d'autres fonctions PostgreSQL.

Note : PHP/FI retournait 1 lorsque la requête n'attendait pas de données en réponse (insertion, modifcations, par exemple), et retournait un nombre plus grand que 1, même sur un select qui donnait un ensemble vide. Ce n'est plus le cas.

pg_Fetch_Array (PHP 3>= 3.0.1, PHP 4 >= 4.0b1)

Lit une ligne dans un tableau.

```
array pg_fetch_array (resource result, int row [, int result_type])
```

pg_fetch_array() retourne un tableau qui contient à la ligne demandée, dans le résultat identifiée par *result*, et FALSE , s'il ne reste plus de lignes.

pg_fetch_array() est une version évoluée de **pg_fetch_row()**. En plus de proposer un tableau à indice numérique, elle peut aussi enregistrer les données dans un tableau associatif, en utilisant les noms des champs comme clés.

L'argument optionnel *result_type* de **pg_fetch_array()** est une constante, qui peut prendre les valeurs suivantes : PGSQL_ASSOC, PGSQL_NUM, et PGSQL_BOTH.

Note : *result_type* a été ajoutée en PHP 4.0.

Il est important de noter que **pg_fetch_array()** n'est pas significativement plus lent que **pg_fetch_row()**, tandis qu'elle fournit un confort d'utilisation notable.

Pour plus de détails, reportez vous à **pg_fetch_row()**.

Exemple 1. PostgreSQL fetch array

```
<?php
$conn = pg_pconnect("dbname=publisher");
if (!$conn) {
    echo "Erreur de connexion.\n";
    exit;
}
$result = pg_exec($conn, "SELECT * FROM authors");
if (!$result) {
    echo "Erreur durant la requete.\n";
    exit;
}
$arr = pg_fetch_array($result, 0);
echo $arr[0] . " <- array\n";
$arr = pg_fetch_array($result, 1);
echo $arr["author"] . " <- array\n";
?>
```

pg_Fetch_Object (PHP 3>= 3.0.1, PHP 4 >= 4.0b1)

Lit une ligne dans un objet.

```
object pg_fetch_object (resource result, int row [, int result_type])
```

pg_fetch_object() retourne un objet dont les membres sont les champs de la ligne demandée, ou FALSE , si il n'y a plus de lignes.

pg_fetch_object() est similaire à **pg_fetch_array()**, avec une différence majeure : c'est un objet qui est retourné, au lieu d'un tableau. Par conséquent, cela signifie que vous ne pouvez accéder aux membres qu'avec leur nom, et non plus leur offset (les nombres ne sont pas autorisés comme nom de membre).

L'argument optionnel *result_type* de *result_type* est une constante qui peut prendre les valeurs suivantes : PGSQL_ASSOC, PGSQL_NUM, et PGSQL_BOTH.

> **Note :** *result_type* a été ajouté dans PHP 4.0.

Au niveau vitesse, **pg_fetch_object()** est aussi rapide que **pg_fetch_row()** et presque aussi rapide que **pg_fetch_row()** (la différence est non significative).

Voir aussi: **pg_fetch_array()** et **pg_fetch_row()**.

Exemple 1. Postgres fetch object

```
<?php
$database = "verlag";
$db_conn = pg_connect("host=localhost port=5432 dbname=$database");
if (!$db_conn):
?>
    <H1>Connexion impossible à la base postgres <?php echo $database ></H1> <?php
    exit;
endif;
$qu = pg_exec($db_conn, "SELECT * FROM verlag ORDER BY autor");
$row = 0; // postgres réclame un compteur de ligne, d'autres bases ne le font pas.
while ($data = pg_fetch_object($qu, $row)):
    echo $data->autor." (";
    echo $data->jahr ."): ";
    echo $data->titel."<BR>";
    $row++;
endwhile;
?>
<PRE><?php
$fields[] = array("autor", "Author");
$fields[] = array("jahr",  "  Year");
$fields[] = array("titel", " Title");
$row= 0; // Postgres réclame un compteur de ligne, d'autres bases ne le font pas.
while ($data = pg_fetch_object($qu, $row)):
    echo "--------\n";
    reset($fields);
    while (list(,$item) = each($fields)):
        echo $item[1].": ".$data->$item[0]."\n";
    endwhile;
    $row++;
endwhile;
echo "--------\n";
?>
</PRE>
<?php
pg_freeresult($qu);
pg_close($db_conn);
?>
```

pg_Fetch_Row (PHP 3>= 3.0.1, PHP 4 >= 4.0b1)

Lit une ligne dans un tableau.

```
array pg_fetch_row (resource result, int row)
```

pg_fetch_row() retourne un tableau qui contient les données de la ligne demandée, ou FALSE , si il ne reste plus de lignes.

pg_fetch_row() lit une ligne dans le résultat associé à l'index *result*. La ligne est retournée sous la forme d'un tableau. La ligne est retournée sous la forme d'un tableau, qui commence à l'index 0.

Les appels ultérieurs à **pg_fetch_row()** retourneront la ligne d'après, ou bien FALSE, lorsqu'il n'y aura plus de lignes.

Voir aussi: **pg_fetch_array()**, **pg_fetch_object()** et **pg_result()**.

Exemple 1. Postgres retourne une ligne

```
<?php
$conn = pg_pconnect("dbname=publisher");
if (!$conn) {
    echo "Une erreur est survenue.\n";
    exit;
}
$result = pg_exec($conn, "SELECT * FROM authors");
if (!$result) {
    echo "Une erreur est survenue.\n";
    exit;
}
$num = pg_numrows($result);
for ($i=0; $i<$num; $i++) {
  $r = pg_fetch_row($result, $i);
  for ($j=0; $j<count($r); $j++) {
    echo "$r[$j] ";
  }
  echo "<BR>";
}
?>
```

pg_FieldIsNull (PHP 3, PHP 4 >= 4.0b1)

Teste si un champs est à NULL.

```
int pg_fieldisnull (resource result_id, int row, mixed field)
```

pg_fieldisnull() teste si un champs est à NULL. **pg_fieldisnull()** retourne 0 si le champs n'est pas NULL. **pg_fieldisnull()** retourne 1 si le champs est à NULL. Le champs peut être identifié avec son nom ou son index numérique (commencant à 0).

pg_FieldName (PHP 3, PHP 4 >= 4.0b1)

Retourne le nom d'un champs.

```
string pg_fieldname (resource result_id, int field_number)
```

pg_fieldname() va retourne le nom du champs qui occupe la colonne numéro *field_number* dans le résultat *result_id*. La numérotation des champs commence à 0.

pg_FieldNum (PHP 3, PHP 4 >= 4.0b1)

Retourne le numéro d'une colonne.

```
int pg_fieldnum (resource result_id, string field_name)
```

pg_fieldnum() retourne le numéro de la colonne, dont le nom est *field_name*, dans le résultat *result_id*. La numérotation des champs commence à 0. Cette fonction retournera -1 en cas d'erreur.

pg_FieldPrtLen (PHP 3, PHP 4 >= 4.0b1)

Retourne la taille imprimée.

```
int pg_fieldprtlen (resource result_id, int row_number, string field_name)
```

pg_fieldprtlen() retourne la taille imprimée (nombre de caractères) d'une valeur donnée dans un résultat PostgreSQL. La numérotation des lignes commence à 0. Cette fonction retourne -1 en cas d'erreur.

pg_FieldSize (PHP 3, PHP 4 >= 4.0b1)

Retourne la taille interne de stockage d'un champs donné.

```
int pg_fieldsize (resource result_id, int field_number)
```

pg_fieldsize() retourne la taille interne de stockage d'un champs donné, en octets. Retourne -1 si la taille est variable. RetourneFALSEen cas d'erreur. La numérotation des colonnes commence à 0.

pg_FieldType (PHP 3, PHP 4 >= 4.0b1)

Retourne le type d'un champs donné par index.

```
string pg_fieldtype (resource result_id, int field_number)
```

pg_fieldtype() retourne une chaîne contenant le type du champs donné par son index field_number . La numérotation des champs commence à 0.

pg_FreeResult (PHP 3, PHP 4 >= 4.0b1)

Libère la mémoire

```
int pg_freeresult (resource result_id)
```

pg_freeresult() n'est vraiment utile que si vous risquez d'utiliser trop de mémoire durant votre script. La mémoire occupée par les résultats est automatiquement libérée à la fin du script. Mais, si vous êtes sÛr de ne pas avoir besoin du résultat ultérieurement, vous pouvez appeler **pg_freeresult()** avec l'index de résultat comme argument, et la mémoire sera libérée.

pg_GetLastOid (PHP 3, PHP 4 >= 4.0b1)

Retourne le dernier identifiant d'objet.

```
resource pg_getlastoid (resource result_id)
```

pg_getlastoid() sert à lire l' Oid assigné à un tuple inséré, si l'index de résultat a été obtenu avec la fonction **pg_exec()**, dont la requête était exclusivement SQL INSERT. Cette fonction retourne un entier positif si un Oid valide a été trouvé. Elle retournera -1 si une erreur est survenue, ou si la dernière commande n'était pas un INSERT.

pg_Host (PHP 3, PHP 4 >= 4.0b1)

Retourne le nom d'hôte.

```
string pg_host (resource connection_id)
```

pg_host() retourne le nom d'hôte associé à l'index de connexion PostgreSQL.

pg_loclose (PHP 3, PHP 4 >= 4.0b1)

Ferme un objet de grande taille.

```
void pg_loclose (resource fd)
```

pg_loclose() ferme un objet de type Inversion Large Object. *fd* est un descripteur de fichier, obtenu avec **pg_loopen()**.

pg_locreate (PHP 3, PHP 4 >= 4.0b1)

Crée un objet de grande taille.

```
resource pg_loimport (resource conn)
```

pg_locreate() crée un objet de type Inversion Large Object et retourne son Oid. *conn* doit être une connexion valide avec une base de données PostgreSQL. Les modes d'accès PostgreSQL INV_READ, INV_WRITE, et INV_ARCHIVE ne sont pas supportés : l'objet peut toujours être créé, avec des droits d'accès en lecture et écriture. Le mode INV_ARCHIVE a été supprimé des bases PostgreSQL (version 6.3 et ultérieur).

pg_loexport (PHP 4 >= 4.0.1)

Exporte un objet de grande taille vers un fichier

```
bool pg_loexport (resource oid, int file [, resource connection_id])
```

oid est un identifiant d'objet de grande taille qui sera exporté dans le fichier *filename*, qui spécifie son chemin. RetourneFALSEsi une erreur survient, et TRUE en cas de succès. N'oubliez pas que la manipulation d'un objet de grande taille dans PostgreSQL doit intervenir dans une transaction.

pg_loimport (PHP 4 >= 4.0.1)

Importe un objet de grande taille depuis un fichier

```
resource pg_loimport (int file [, resource connection_id])
```

filename est le chemin jusqu'à un fichier qui servira de source pour créer un objet de grande taille. La fonction retourneFALSEen cas d'erreur, et sinon un identifiant d'objet, créé directement à la bonne taille. N'oubliez pas que la manipulation d'un objet de grande taille dans PostgreSQL doit intervenir dans une transaction.

pg_loopen (PHP 3, PHP 4 >= 4.0b1)

Ouvre un objet de grande taille.

```
int pg_loopen (resource conn, resource objoid, string mode)
```

pg_loopen() ouvre un objet de type Inversion Large Object et retourne un descripteur de fichier pour cet objet. Le descripteur de fichier contient les informations de connexion. Ne refermez pas la connexion avant d'avoir fermé l'objet. *objoid* est un Oid valide de Large Object, et *mode* peut prendre es valeurs suivantes : "r", "w", ou "rw".

pg_loread (PHP 3, PHP 4 >= 4.0b1)

Lit un objet de grande taille.

```
string pg_loread (resource loid, int len)
```

pg_loread() lit au plus *len* octets d'un objet de grande taille, et retourne les données sous la forme d'une chaîne. *loid* est un identifiant valide d'objet de grande taille, et *len* indique la taille maximale de mémoire alloué à l'objet de grande taille.

pg_loreadall (PHP 3, PHP 4 >= 4.0b1)

Lit un objet de grande taille en totalité.

```
void pg_loreadall (resource fd)
```

pg_loreadall() lit un objet de grande taille en totalité et le passe directement au client, après les entêtes adéquats. Cette fonction est prévue pour transmettre des sons ou des images.

pg_lounlink (PHP 3, PHP 4 >= 4.0b1)

Efface un objet de grande taille

```
void pg_lounlink (resource conn, resource lobjid)
```

pg_lounlink() efface l' objet de grande taille dont l'identifiant est *lobjid*.

pg_lowrite (PHP 3, PHP 4 >= 4.0b1)

Ecrit un objet de grande taille

```
int pg_lowrite (resource fd, string buf)
```

pg_lowrite() écrit dans l'objet de grande taille autant de données possible, issues de la variable *buf* et retourne le nombre d'octets réellement écrits, ouFALSEen cas d'erreur. *fd* est un descripteut d'objet de grande taille, obtenu avec **pg_loopen()**.

pg_NumFields (PHP 3, PHP 4 >= 4.0b1)

Retourne le nombre de champs

```
int pg_numfields (resource result_id)
```

pg_numfields() retourne le nombre de champs ou (colonnes) d'un résultat PostgreSQL. L'argument doit être un identifiant de résultat valide retourné par **pg_exec()**. Cette fonction retournera -1 en cas d'erreur.

pg_NumRows (PHP 3, PHP 4 >= 4.0b1)

Retourne le nombre de lignes.

```
int pg_numrows (resource result_id)
```

pg_numrows() retourne le nombre de lignes d'un résultat PostgreSQL. L'argument doit être un identifiant de résultat valide retourné par **pg_exec()**. Cette fonction retournera -1 en cas d'erreur.

pg_Options (PHP 3, PHP 4 >= 4.0b1)

Retourne les options.

```
string pg_options (resource connection_id)
```

pg_options() retourne une chaîne contenant les options de la connexion PostgreSQL.

pg_pConnect (PHP 3, PHP 4 >= 4.0b1)

Etablit une connexion persistante.

```
int pg_pconnect (string conn_string)
```

pg_pconnect() retourne un index de connexion en cas de succès, ou FALSE en cas d'erreur. **pg_pconnect()** ouvre une connexion permanente à une base PostgreSQL. Les arguments doivent être insérés dans une chaîne à guillemets. Ils incluent : *host*, *port*, *tty*, *options*, *dbname*, *user* et *password*.

pg_pconnect() retourne un indentifiant de connexion qui sera utilisées par les autres fonctions PostgreSQL. Vous pouvez ouvrir plusieurs connexions en même temps.

L'ancienne syntaxe **$conn = pg_pconnect("host", "port", "options", "tty", "dbname")** est obsolète.

pg_Port (PHP 3, PHP 4 >= 4.0b1)

Retourne le numéro de port.

```
int pg_port (resource connection_id)
```

pg_port() retourne le numéro de port de la connexion identifiée *connection_id*.

pg_put_line (PHP 4 >= 4.0.3)

Envoie une chaîne au serveur PostgreSQL

```
bool pg_put_line ([resource connection_id, string data])
```

pg_put_line() envoie une chaîne (terminée par NULL) au serveur PostgreSQL. Ceci est pratique pour effectuer des insertions très rapides dans une table, initiée par une opération de copie PostgreSQL copy-operation. Le caractère final NULL est automatiquement ajouté. Retourne TRUE en cas de succès, et FALSE.

> **Note :** Notez que l'application doit explicitement ajouter les deux caractères "\." à la fin de la chaîne pour indiquer au serveur qu'elle a finit d'envoyer des données.

Voir aussi **pg_end_copy()**.

Exemple 1. Insertion à grande vitesse dans une table

```
<?php
    $conn = pg_pconnect("dbname=foo");
    pg_exec($conn, "create table bar (a int4, b char(16), d float8)");
    pg_exec($conn, "copy bar from stdin");
    pg_put_line($conn, "3\tBonjour le monde\t4.5\n");
    pg_put_line($conn, "4\tAu revoir le monde\t7.11\n");
    pg_put_line($conn, "\\.\n");
    pg_end_copy($conn);
?>
```

pg_Result (PHP 3, PHP 4 >= 4.0b1)

Retourne les valeurs d'un identifiant de résultat.

```
mixed pg_result (resource result_id, int row_number, mixed fieldname)
```

pg_result() retourne les valeurs d'un identifiant de résultat, produit par **pg_exec()**. Les arguments *row_number* et *fieldname* précisent la cellule qui sera retournée. La numérotation des lignes commence à 0. Au lieu d'utiliser le nom du champs, vous pouvez utiliser son index, sous la forme d'un nombre sans guillemets. La numérotation des champs commence à 0.

PostgreSQL dispose de nombreux types, et seuls, les types basiques sont supportés ici. Toutes les formes d'entier, booléen et Oid sont retournés sous la forme d'entiers. Toutes les formes de nombre à virgule flottante et types réels sont retournés sous la forme d'une valeur de type double. Tous les autres types, y compris les tableaux, sont retournés sous la forme de chaînes formatées, au format par défaut de PostgreSQL.

pg_set_client_encoding (PHP 3 CVS only, PHP 4 >= 4.0.3)

Choisi l'encodage du client

```
int pg_set_client_encoding ([resource connection, string encoding])
```

pg_set_client_encoding() fixe l'encodage du client. Elle retourne 0 en cas de succès, et -1 sinon.

encoding est l'encodage du client, et peut être SQL_ASCII, EUC_JP, EUC_CN, EUC_KR, EUC_TW, UNICODE, MULE_INTERNAL, LATINX (X=1...9), KOI8, WIN, ALT, SJIS, BIG5, WIN1250.

> **Note :** Cette fonction requiert PHP-4.0.2 ou plus récent et PostgreSQL-7.0 ou plus récent.
>
> Jadis, **pg_set_client_encoding()** s'appelait pg_setclientencoding().

Voir aussi **pg_client_encoding()**.

pg_client_encoding (PHP 3 CVS only, PHP 4 >= 4.0.3)

Lit l'encodage du client

```
string pg_client_encoding ([resource connection])
```

pg_client_encoding() retourne l'encodage du client. Elle retourne une des valeurs suivantes : SQL_ASCII, EUC_JP, EUC_CN, EUC_KR, EUC_TW, UNICODE, MULE_INTERNAL, LATINX (X=1...9), KOI8, WIN, ALT, SJIS, BIG5, WIN1250.

> **Note :** Cette fonction requiert PHP-4.0.2 ou plus récent et PostgreSQL-7.0 ou plus récent.
>
> Jadis, **pg_client_encoding()** s'appelait pg_clientencoding().

Voir aussi**pg_set_client_encoding()**.

pg_trace (PHP 4 >= 4.0.1)

Active le suivi d'une connexion PostgreSQL

```
bool pg_trace (string filename [, string mode [, resource connection]])
```

pg_trace() active le suivi des communications entre PHP et le serveur PostgreSQL. Cet historique sera enregistré dans un fichier. Pour comprendre ces lignes, il faut être familier avec le protocole de communication interne à PostgreSQL. Pour ceux qui le ne sont pas, elles peuvent être utiles pour suivre les requêtes et les erreurs : avec la commande **grep '^To backend' trace.log**, vous pourrez voir les requêtes réellement envoyées au serveur PostgreSQL.

filename et *mode* sont les mêmes arguments que pour la fonction **fopen()** (*mode* par défaut à 'w'), *connection* indique la connexion à suivre. Par défaut, c'est la dernière ouverte.

Retourne TRUE si *filename* a pu être ouvert en écriture, etFALSEsinon.

Voir aussi **fopen()** et **pg_untrace()**.

pg_tty (PHP 3, PHP 4 >= 4.0b1)

Retourne le nom de tty.

```
string pg_tty (resource connection_id)
```

pg_tty() retourne le nom de tty de la connexion associée à *connection_id*.

pg_untrace (PHP 4 >= 4.0.1)

Termine le suivi d'une connexion PostgreSQL

```
bool pg_untrace ([resource connection])
```

pg_untrace() termine le suivi d'une connexion PostgreSQL, initiée avec **pg_trace()**. *connection* indique la connexion à suivre. Par défaut, c'est la dernière ouverte.

pg_untrace() retourne toujours TRUE.

Voir aussi **pg_trace()**.

LXVIII. Exécution de programmes externes

Ces fonctions fournissent la possibilité de passer directement des commandes au système, mais aussi de protéger le système des commandes passées. Ces fonctions sont complétées par l’opérateur guillemets obliques.

escapeshellarg (PHP 4 >= 4.0.3)

Echappe une chaîne de caractères pour qu'elle soit utilisée en ligne de commande.

```
string escapeshellarg (string arg)
```

escapeshellarg() ajoute des guillemets simples autour des chaînes de caractères, et ajoute des guillemets puis échappe les guillemets simples de la chaîne. Cela permet de faire passer directement une chaîne comme argument shell, tout en assurant un maximum de sécurité. **escapeshellarg()** doit être utilisée pour traiter individuellement chacun des arguments à passer au shell. Les fonctions shell sont **exec()**, **system()** et les opérateurs guillemets obliques. Une utilisation typique est :

```
<?php
  system("ls ".escapeshellarg($dir));
?>
```

Voir aussi **exec()**, **popen()**, **system()** et les opérateurs guillemets obliques.

escapeshellcmd (PHP 3, PHP 4 >= 4.0b1)

Echappe les méta-caractères Shell.

```
string escapeshellcmd (string command)
```

escapeshellcmd() échappe tous les caractères de la chaîne *command* qui pourraient avoir une signification spéciale dans une commande shell. Cette fonction permet de s'assurer que la commande sera correctement passée à l'exécuteur de commande shell **exec()** et **system()**, ou encore à guillemets obliques. Généralement, cette fonction est utilisée comme ceci :

```
<?php
  system(escapeshellcmd($cmd));
?>
```

Voir aussi **exec()**, **popen()**, **system()**, et les opérateurs guillemets obliques.

exec (PHP 3, PHP 4 >= 4.0b1)

Exécute un programme externe.

```
string exec (string command, string [array] [, int return_var])
```

exec() éxecute la commande *command*, mais ne renvoie rien comme retour, hormis la dernière ligne du résultat de la commande. Pour exécuter une commande et obtenir le résultat sans aucun traitement, il faut utiliser la fonction **passthru()**.

Si l'argument *array* est présent, alors ce tableau sera rempli par les lignes retournées par la commande. Il faut noter que si ce tableau contient des éléments, **exec()** ajoutera les nouvelles lignes à la fin du tableau. Si vous ne voulez pars que les nouveaux éléments soient concaténés, utilisez la fonction **unset()** avec ce tableau avant de le passer à **exec()**.

Si l'argument *return_var* est présent en plus du tableau *array*, alors de statut de retour d'exécution sera inscrit dans cette variable.

Notez que si vous allez fournir des commandes qui proviennent d'un utilisateur, il est avisé d'utiliser la fonction **escapeshellcmd()** pour s'assurer que l'utilisateur n'essaie pas de profiter des caractères spéciaux pour tromper le système.

Voir aussi **system()**, **passthru()**, **popen()**, **escapeshellcmd()**, et les opérateurs guillemets obliques.

passthru (PHP 3, PHP 4 >= 4.0b1)

Exécute un programme externe et affiche le résultat brut.

```
void passthru (string command [, int return_var])
```

La fonction **passthru()** est similaire à la fonction **exec()** car les deux exécutent la commande *command*. Si l'argument *return_var* est présent, le code de statut de réponse UNIX y sera placé. Cette fonction doit être utilisée de préférence aux commandes **exec()** ou **system()** lorsque le résultat attendu est de type binaire, et doit être passé tel quel à un navigateur. Une utilisation classique de cette fonction est l'exécution de l'utilitaire pbmplus qui peut retourner une image. En fixant le résultat du contenu (Content-Type) à "image/gif" puis en appelant pbmplus pour obtenir une image gif, vous pouvez créer des scripts PHP qui retourne des images.

Voir aussi **exec()**, **system()**, **popen()**, **escapeshellcmd()**, et les opérateurs guillemets obliques.

system (PHP 3, PHP 4 >= 4.0b1)

Exécute un programme externe et affiche le résultat.

```
string system (string command [, int return_var])
```

system() est la version PHP de la fonction C qui exécute la commande *command* et retourne le résultat. Si une variable est fournie comme second argument, alors le code de statut de la commande y sera affecté.

Notez que si vous allez fournir des commandes qui proviennent d'un utilisateur, il est avisé d'utiliser la fonction **escapeshellcmd()** pour s'assurer que l'utilisateur n'essaie pas de profiter des caractères spéciaux pour tromper le système.

system() essaie automatiquement de vider les tampons du serveur web après chaque ligne de résultat PHP, lorsque ce dernier fonctionne comme un module.

system() retourne la dernière ligne du retour, en cas de succès, et FALSE en cas d'échec.

Si vous devez exécuter une commande et récupérer tout le résultat sans aucune intervention, utilisez la fonction **passthru()**.

Voir aussi **exec()**, **passthru()**, **popen()**, **escapeshellcmd()** et les opérateurs guillemets obliques.

LXIX. Printer functions

These functions are only available under Windows 9.x, ME, NT4 and 2000. They have been added in PHP 4 (4.0.4).

printer_open (PHP 4 >= 4.0.4)

Open connection to a printer

```
mixed printer_open ([string devicename])
```

This function tries to open a connection to the printer *devicename*, and returns a handle on success or FALSE on failure.

If no parameter was given it tries to open a connection to the default printer (if not specified in php.ini as printer.default_printer, php tries to detect it).

printer_open() also starts a device context.

Exemple 1. printer_open() example

```
$handle = printer_open("HP Deskjet 930c");
$handle = printer_open();
```

printer_abort (PHP 4 >= 4.0.6)

Deletes the printer's spool file

```
void printer_abort (resource handle)
```

This function deletes the printers spool file.

handle must be a valid handle to a printer.

Exemple 1. printer_abort() example

```
$handle = printer_open();
printer_abort($handle);
printer_close($handle);
```

printer_close (PHP 4 >= 4.0.4)

Close an open printer connection

```
void printer_close (resource handle)
```

This function closes the printer connection. **printer_close()** also closes the active device context.

handle must be a valid handle to a printer.

Exemple 1. printer_close() example

```
$handle = printer_open();
printer_close($handle);
```

printer_write (PHP 4 >= 4.0.4)

Write data to the printer

```
bool printer_write (resource handle, string content)
```

Writes *content* directly to the printer, and returns TRUE on success or FALSE if it failed.

handle must be a valid handle to a printer.

Exemple 1. printer_write() example

```
$handle = printer_open();
printer_write($handle, "Text to print");
printer_close($handle);
```

printer_list (PHP 4 >= 4.0.4)

Return an array of printers attached to the server

```
array printer_list (int enumtype [, string name [, int level]])
```

The function enumerates available printers and their capabilities. *level* sets the level of information request. Can be 1,2,4 or 5. *enumtype* must be one of the following predefined constants:

- *PRINTER_ENUM_LOCAL*: enumerates the locally installed printers.
- *PRINTER_ENUM_NAME*: enumerates the printer of *name*, can be a server, domain or print provider.
- *PRINTER_ENUM_SHARED*: this parameter can't be used alone, it has to be OR'ed with other parameters, i.e. PRINTER_ENUM_LOCAL to detect the locally shared printers.
- *PRINTER_ENUM_DEFAULT*: (Win9.x only) enumerates the default printer.
- *PRINTER_ENUM_CONNECTIONS*: (WinNT/2000 only) enumerates the printers to which the user has made connections.
- *PRINTER_ENUM_NETWORK*: (WinNT/2000 only) enumerates network printers in the computer's domain. Only valid if *level* is 1.
- *PRINTER_ENUM_REMOTE*: (WinNT/2000 only) enumerates network printers and print servers in the computer's domain. Only valid if *level* is 1.

Exemple 1. printer_list() example

```
/* detect locally shared printer */
var_dump( printer_list(PRINTER_ENUM_LOCAL | PRINTER_ENUM_SHARED) );
```

printer_set_option (PHP 4 >= 4.0.4)

Configure the printer connection

```
bool printer_set_option (resource handle, int option, mixed value)
```

The function sets the following options for the current connection: *handle* must be a valid handle to a printer. For *option* can be one of the following constants:

- *PRINTER_COPIES*: sets how many copies should be printed, *value* must be an integer.
- *PRINTER_MODE*: specifies the type of data (text, raw or emf), *value* must be a string.
- *PRINTER_TITLE*: specifies the name of the document, *value* must be a string.
- *PRINTER_ORIENTATION*: specifies the orientation of the paper, *value* can be either PRINTER_ORIENTATION_PORTRAIT or PRINTER_ORIENTATION_LANDSCAPE
- *PRINTER_RESOLUTION_Y*: specifies the y-resolution in DPI, *value* must be an integer.
- *PRINTER_RESOLUTION_X*: specifies the x-resolution in DPI, *value* must be an integer.
- *PRINTER_PAPER_FORMAT*: specifies the a predefined paper format, set *value* to PRINTER_FORMAT_CUSTOM if you want to specify a custom format with PRINTER_PAPER_WIDTH and PRINTER_PAPER_LENGTH. *value* can be one of the following constants.
  - *PRINTER_FORMAT_CUSTOM*: let's you specify a custom paper format.
  - *PRINTER_FORMAT_LETTER*: specifies standard letter format (8 1/2- by 11-inches).
  - *PRINTER_FORMAT_LETTER*: specifies standard legal format (8 1/2- by 14-inches).
  - *PRINTER_FORMAT_A3*: specifies standard A3 format (297- by 420-millimeters).
  - *PRINTER_FORMAT_A4*: specifies standard A4 format (210- by 297-millimeters).
  - *PRINTER_FORMAT_A5*: specifies standard A5 format (148- by 210-millimeters).
  - *PRINTER_FORMAT_B4*: specifies standard B4 format (250- by 354-millimeters).
  - *PRINTER_FORMAT_B5*: specifies standard B5 format (182- by 257-millimeter).
  - *PRINTER_FORMAT_FOLIO*: specifies standard FOLIO format (8 1/2- by 13-inch).

- *PRINTER_PAPER_LENGTH*: if PRINTER_PAPER_FORMAT is set to PRINTER_FORMAT_CUSTOM, PRINTER_PAPER_LENGTH specifies a custom paper length in mm, *value* must be an integer.
- *PRINTER_PAPER_WIDTH*: if PRINTER_PAPER_FORMAT is set to PRINTER_FORMAT_CUSTOM, PRINTER_PAPER_WIDTH specifies a custom paper width in mm, *value* must be an integer.
- *PRINTER_SCALE*: specifies the factor by which the printed output is to be scaled. the page size is scaled from the physical page size by a factor of scale/100. for example if you set the scale to 50, the output would be half of it's original size. *value* must be an integer.
- *PRINTER_BACKGROUND_COLOR*: specifies the background color for the actual device context, *value* must be a string containing the rgb information in hex format i.e. "005533".
- *PRINTER_TEXT_COLOR*: specifies the text color for the actual device context, *value* must be a string containing the rgb information in hex format i.e. "005533".
- *PRINTER_TEXT_ALIGN*: specifies the text alignment for the actual device context, *value* can be combined through OR'ing the following constants:
  - *PRINTER_TA_BASELINE*: text will be aligned at the base line.
  - *PRINTER_TA_BOTTOM*: text will be aligned at the bottom.
  - *PRINTER_TA_TOP*: text will be aligned at the top.
  - *PRINTER_TA_CENTER*: text will be aligned at the center.
  - *PRINTER_TA_LEFT*: text will be aligned at the left.
  - *PRINTER_TA_RIGHT*: text will be aligned at the right.

Exemple 1. printer_set_option() example

```
$handle = printer_open();
printer_set_option($handle, PRINTER_SCALE, 75);
printer_set_option($handle, PRINTER_TEXT_ALIGN, PRINTER_TA_LEFT);
printer_close($handle);
```

printer_get_option (PHP 4 >= 4.0.4)

Retrieve printer configuration data

```
mixed printer_get_option (resource handle, string option)
```

The function retrieves the configuration setting of *option*. *handle* must be a valid handle to a printer. Take a look at **printer_set_option()** for the settings that can be retrieved, additionally the following settings can be retrieved:

- *PRINTER_DEVICENAME* returns the devicename of the printer.
- *PRINTER_DRIVERVERSION* returns the printer driver version.

Exemple 1. printer_get_option() example

```
$handle = printer_open();
print printer_get_option($handle, PRINTER_DRIVERVERSION);
printer_close($handle);
```

printer_create_dc (PHP 4 >= 4.0.4)

Create a new device context

```
void printer_create_dc (resource handle)
```

The function creates a new device context. A device context is used to customize the graphic objects of the document. *handle* must be a valid handle to a printer.

Exemple 1. printer_create_dc() example

```
$handle = printer_open();
printer_start_doc($handle);
printer_start_page($handle);

printer_create_dc($handle);
/* do some stuff with the dc */
printer_set_option($handle, PRINTER_TEXT_COLOR, "333333");
printer_draw_text($handle, 1, 1, "text");
printer_delete_dc($handle);

/* create another dc */
printer_create_dc($handle);
printer_set_option($handle, PRINTER_TEXT_COLOR, "000000");
printer_draw_text($handle, 1, 1, "text");
/* do some stuff with the dc */

printer_delete_dc($handle);

printer_endpage($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_delete_dc (PHP 4 >= 4.0.4)

Delete a device context

```
bool printer_delete_dc (resource handle)
```

The function deletes the device context and returns TRUE on success, or FALSE if an error occurred. For an example see **printer_create_dc()**. *handle* must be a valid handle to a printer.

printer_start_doc (PHP 4 >= 4.0.4)

Start a new document

```
bool printer_start_doc (resource handle [, string document])
```

The function creates a new document in the printer spooler. A document can contain multiple pages, it's used to schedule the print job in the spooler. *handle* must be a valid handle to a printer. The optional parameter *document* can be used to set an alternative document name.

Exemple 1. printer_start_doc() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_end_doc (PHP 4 >= 4.0.4)

Close document

```
bool printer_end_doc  (resource handle)
```

Closes a new document in the printer spooler. The document is now ready for printing. For an example see **printer_start_doc()**. *handle* must be a valid handle to a printer.

printer_start_page (PHP 4 >= 4.0.4)

Start a new page

```
bool printer_start_page  (resource handle)
```

The function creates a new page in the active document. For an example see **printer_start_doc()**. *handle* must be a valid handle to a printer.

printer_end_page (PHP 4 >= 4.0.4)

Close active page

```
bool printer_end_page  (resource handle)
```

The function closes the active page in the active document. For an example see **printer_start_doc()**. *handle* must be a valid handle to a printer.

printer_create_pen (PHP 4 >= 4.0.4)

Create a new pen

```
mixed printer_create_pen  (int style, int width, string color)
```

The function creates a new pen and returns a handle to it. A pen is used to draw lines and curves. For an example see **printer_select_pen()**. *color* must be a color in RGB hex format, i.e. "000000" for black, *width* specifies the width of the pen whereas *style* must be one of the following constants:

- *PRINTER_PEN_SOLID*: creates a solid pen.
- *PRINTER_PEN_DASH*: creates a dashed pen.
- *PRINTER_PEN_DOT*: creates a dotted pen.
- *PRINTER_PEN_DASHDOT*: creates a pen with dashes and dots.
- *PRINTER_PEN_DASHDOTDOT*: creates a pen with dashes and double dots.
- *PRINTER_PEN_INVISIBLE*: creates an invisible pen.

printer_delete_pen (PHP 4 >= 4.0.4)

Delete a pen

```
bool printer_delete_pen  (resource handle)
```

The function deletes the selected pen. For an example see **printer_select_pen()**. It returns TRUE on success, or FALSE otherwise. *handle* must be a valid handle to a pen.

printer_select_pen (PHP 4 >= 4.0.4)

Select a pen

```
void printer_select_pen  (resource printer_handle, resource pen_handle)
```

The function selects a pen as the active drawing object of the actual device context. A pen is used to draw lines and curves. I.e. if you draw a single line the pen is used. If you draw an rectangle the pen is used to draw the borders, while the brush is used to fill the shape. If you haven't selected a pen before drawing shapes, the shape won't be outlined.
printer_handle must be a valid handle to a printer. *pen_handle* must be a valid handle to a pen.

Exemple 1. printer_select_pen() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 30, "2222FF");
printer_select_pen($handle, $pen);

printer_draw_line($handle, 1, 60, 500, 60);

printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_create_brush (PHP 4 >= 4.0.4)

Create a new brush

```
mixed printer_create_brush  (int style, string color)
```

The function creates a new brush and returns a handle to it. A brush is used to fill shapes. For an example see **printer_select_brush()**. *color* must be a color in RGB hex format, i.e. "000000" for black, *style* must be one of the following constants:

- *PRINTER_BRUSH_SOLID*: creates a brush with a solid color.
- *PRINTER_BRUSH_DIAGONAL*: creates a brush with a 45-degree upward left-to-right hatch (/).
- *PRINTER_BRUSH_CROSS*: creates a brush with a cross hatch (+).
- *PRINTER_BRUSH_DIAGCROSS*: creates a brush with a 45 cross hatch (x).
- *PRINTER_BRUSH_FDIAGONAL*: creates a brush with a 45-degree downward left-to-right hatch (\ ).
- *PRINTER_BRUSH_HORIZONTAL*: creates a brush with a horizontal hatch (-).
- *PRINTER_BRUSH_VERTICAL*: creates a brush with a vertical hatch (|).
- *PRINTER_BRUSH_CUSTOM*: creates a custom brush from an BMP file. The second parameter is used to specify the BMP instead of the RGB color code.

printer_delete_brush (PHP 4 >= 4.0.4)

Delete a brush

```
bool printer_delete_brush  (resource handle)
```

The function deletes the selected brush. For an example see **printer_select_brush()**. It returns TRUE on success, or FALSE otherwise. *handle* must be a valid handle to a brush.

printer_select_brush (PHP 4 >= 4.0.4)

Select a brush

```
void printer_select_brush  (resource printer_handle, resource brush_handle)
```

The function selects a brush as the active drawing object of the actual device context. A brush is used to fill shapes. If you draw an rectangle the brush is used to draw the shapes, while the pen is used to draw the border. If you haven't selected a brush before drawing shapes, the shape won't be filled. *printer_handle* must be a valid handle to a printer. *brush_handle* must be a valid handle to a brush.

Exemple 1. printer_select_brush() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);
$brush = printer_create_brush(PRINTER_BRUSH_CUSTOM, "c:\\brush.bmp");
printer_select_brush($handle, $brush);

printer_draw_rectangle($handle, 1,1,500,500);

printer_delete_brush($brush);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "000000");
printer_select_brush($handle, $brush);
printer_draw_rectangle($handle, 1,501,500,1001);
printer_delete_brush($brush);

printer_delete_pen($pen);


printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_create_font (PHP 4 >= 4.0.4)

Create a new font

```
mixed printer_create_font  (string face, int height, int width, int font_weight, bool
italic, bool underline, bool strikeout, int orientaton)
```

The function creates a new font and returns a handle to it. A font is used to draw text. For an example see **printer_select_font()**. *face* must be a string specifying the font face. *height* specifies the font height, and *width* the font width. The *font_weight* specifies the font weight (400 is normal), and can be one of the following predefined constants.

- *PRINTER_FW_THIN*: sets the font weight to thin (100).
- *PRINTER_FW_ULTRALIGHT*: sets the font weight to ultra light (200).
- *PRINTER_FW_LIGHT*: sets the font weight to light (300).
- *PRINTER_FW_NORMAL*: sets the font weight to normal (400).
- *PRINTER_FW_MEDIUM*: sets the font weight to medium (500).
- *PRINTER_FW_BOLD*: sets the font weight to bold (700).

- *PRINTER_FW_ULTRABOLD*: sets the font weight to ultra bold (800).
- *PRINTER_FW_HEAVY*: sets the font weight to heavy (900).

italic can be TRUE or FALSE, and sets whether the font should be italic.

underline can be TRUE or FALSE, and sets whether the font should be underlined.

strikeout can be TRUE or FALSE, and sets whether the font should be striked out.

orientation specifies a rotation. For an example see **printer_select_font()**.

printer_delete_font (PHP 4 >= 4.0.4)

Delete a font

```
bool printer_delete_font  (resource handle)
```

The function deletes the selected font. For an example see **printer_select_font()**. It returns TRUE on success, or FALSE otherwise. *handle* must be a valid handle to a font.

printer_select_font (PHP 4 >= 4.0.4)

Select a font

```
void printer_select_font  (resource printer_handle, resource font_handle)
```

The function selects a font to draw text. *printer_handle* must be a valid handle to a printer. *font_handle* must be a valid handle to a font.

Exemple 1. printer_select_font() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$font = printer_create_font("Arial", 148, 76, PRINTER_FW_MEDIUM, false, false, false, -50);
printer_select_font($handle, $font);
printer_draw_text($handle, "PHP is simply cool", 40, 40);
printer_delete_font($font);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_logical_fontheight (PHP 4 >= 4.0.4)

Get logical font height

```
int printer_logical_fontheight  (resource handle, int height)
```

The function calculates the logical font height of *height*. *handle* must be a valid handle to a printer.

Exemple 1. printer_logical_fontheight() example

```
$handle = printer_open();
print printer_logical_fontheight($handle, 72);
printer_close($handle);
```

printer_draw_roundrect (PHP 4 >= 4.0.4)

Draw a rectangle with rounded corners

```
void printer_draw_roundrect  (resource handle, int ul_x, int ul_y, int lr_x, int lr_y, int
width, int height)
```

The function simply draws a rectangle with rounded corners.

handle must be a valid handle to a printer.

ul_x is the upper left x coordinate of the rectangle.

ul_y is the upper left y coordinate of the rectangle.

lr_x is the lower right x coordinate of the rectangle.

lr_y is the lower right y coordinate of the rectangle.

width is the width of the ellipse.

height is the height of the ellipse.

Exemple 1. printer_draw_roundrect() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_roundrect($handle, 1, 1, 500, 500, 200, 200);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_rectangle (PHP 4 >= 4.0.4)

Draw a rectangle

```
void printer_draw_rectangle  (resource handle, int ul_x, int ul_y, int lr_x, int lr_y)
```

The function simply draws a rectangle.

handle must be a valid handle to a printer.

ul_x is the upper left x coordinate of the rectangle.

ul_y is the upper left y coordinate of the rectangle.

lr_x is the lower right x coordinate of the rectangle.

lr_y is the lower right y coordinate of the rectangle.

Exemple 1. printer_draw_rectangle() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_rectangle($handle, 1, 1, 500, 500);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_elipse (PHP 4 >= 4.0.4)

Draw an ellipse

```
void printer_draw_elipse  (resource handle, int ul_x, int ul_y, int lr_x, int lr_y)
```

The function simply draws an ellipse. *handle* must be a valid handle to a printer.

ul_x is the upper left x coordinate of the ellipse.

ul_y is the upper left y coordinate of the ellipse.

lr_x is the lower right x coordinate of the ellipse.

lr_y is the lower right y coordinate of the ellipse.

Exemple 1. printer_draw_elipse() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_elipse($handle, 1, 1, 500, 500);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_text (PHP 4 >= 4.0.4)

Draw text

```
void printer_draw_text  (resource printer_handle, string text, int x, int y)
```

The function simply draws *text* at position *x*, *y* using the selected font. *printer_handle* must be a valid handle to a printer.

Exemple 1. printer_draw_text() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$font = printer_create_font("Arial",72,48,400,false,false,false,0);
printer_select_font($handle, $font);
printer_draw_text($handle, "test", 10, 10);
printer_delete_font($font);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_line (PHP 4 >= 4.0.6)

Draw a line

```
void printer_draw_line (resource printer_handle, int from_x, int from_y, int to_x, int
to_y)
```

The function simply draws a line from position *from_x*, *from_y* to position *to_x*, *to_y* using the selected pen. *printer_handle* must be a valid handle to a printer.

Exemple 1. printer_draw_line() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 30, "000000");
printer_select_pen($handle, $pen);

printer_draw_line($handle, 1, 10, 1000, 10);
printer_draw_line($handle, 1, 60, 500, 60);

printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_chord (PHP 4 >= 4.0.6)

Draw a chord

```
void printer_draw_chord  (resource handle, int rec_x, int rec_y, int rec_x1, int rec_y1,
int rad_x, int rad_y, int rad_x1, int rad_y1)
```

The function simply draws an chord. *handle* must be a valid handle to a printer.

rec_x is the upper left x coordinate of the bounding rectangle.

rec_y is the upper left y coordinate of the bounding rectangle.

rec_x1 is the lower right x coordinate of the bounding rectangle.

rec_y1 is the lower right y coordinate of the bounding rectangle.

rad_x is x coordinate of the radial defining the beginning of the chord.

rad_y is y coordinate of the radial defining the beginning of the chord.

rad_x1 is x coordinate of the radial defining the end of the chord.

rad_y1 is y coordinate of the radial defining the end of the chord.

Exemple 1. printer_draw_chord() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_chord($handle, 1, 1, 500, 500, 1, 1, 500, 1);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_pie (PHP 4 >= 4.0.6)

Draw a pie

```
void printer_draw_pie (resource handle, int rec_x, int rec_y, int rec_x1, int rec_y1, int
rad1_x, int rad1_y, int rad2_x, int rad2_y)
```

The function simply draws an pie. *handle* must be a valid handle to a printer.

rec_x is the upper left x coordinate of the bounding rectangle.

rec_y is the upper left y coordinate of the bounding rectangle.

rec_x1 is the lower right x coordinate of the bounding rectangle.

rec_y1 is the lower right y coordinate of the bounding rectangle.

rad1_x is x coordinate of the first radial's ending.

rad1_y is y coordinate of the first radial's ending.

rad2_x is x coordinate of the second radial's ending.

rad2_y is y coordinate of the second radial's ending.

Exemple 1. printer_draw_chord() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_pie($handle, 1, 1, 500, 500, 1, 1, 500, 1);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_bmp (PHP 4 >= 4.0.6)

Draw a bmp

```
void printer_draw_bmp  (resource handle, string filename, int x, int y)
```

The function simply draws an bmp the bitmap *filename* at position *x*, *y*. *handle* must be a valid handle to a printer.

The function returns TRUE on success, or otherwise FALSE.

Exemple 1. printer_draw_bmp() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

printer_draw_bmp($handle, "c:\\image.bmp", 1, 1);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

LXX. Pspell

La librairie pspell vous permet de vérifier l'orthographe d'un mot, et suggérer des corrections.

Vous aurez besoin des librairies aspell et pspell, disponibles à http://aspell.sourceforge.net/ et http://pspell.sourceforge.net/ (respectivement). Il faut aussi ajouter l'otpion `--with-pspell[=dir]` lors de la compilation de PHP.

pspell_add_to_personal (PHP 4 >= 4.0.2)

Ajoute le mot au dictionnaire personnel.

```
int pspell_add_to_personal (resource dictionary_link, string word)
```

pspell_add_to_personal() ajoute un mot au dictionnaire personnel. Si vous utilisez **pspell_new_config()** avec **pspell_config_personal()** pour ouvrir le dictionnaire, vous pourrez sauver le dictionnaire personnel ultérieurement avec **pspell_save_wordlist()**. Notez bien que cette fonction ne fonctionnera pas avec les versions antérieures pspell .11.2 et aspell .32.5.

Exemple 1. Exemple avec pspell_add_to_personal()

```
<?php
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
$pspell_link = pspell_new_config ($pspell_config);
pspell_add_to_personal ($pspell_link, "Vlad");
pspell_save_wordlist ($pspell_link);
?>
```

pspell_add_to_session (PHP 4 >= 4.0.2)

Ajoute le mot au dictionnaire personnel de la session courante

```
int pspell_add_to_session (resource dictionary_link, string word)
```

pspell_add_to_session() ajoute un mot au dictionnaire personnel associé à la version courante. C’est une fonction similaire à **pspell_add_to_personal()**.

pspell_check (PHP 4 >= 4.0.2)

Vérifie un mot

```
boolean pspell_check (resource dictionary_link, string word)
```

pspell_check() vérifie l’orthographe d’un mot et retourne TRUE si l’orthographe est correcte, FALSE sinon.

Exemple 1. pspell_check()

```
<?php
$pspell_link = pspell_new ("french");
if (pspell_check ($pspell_link, "testt")) {
    echo "L’orthographe est exacte";
} else {
    echo "Désolé, mauvaise orthographe";
}
?>
```

pspell_clear_session (PHP 4 >= 4.0.2)

Remet à zéro la session courante.

```
int pspell_clear_session (resource dictionary_link)
```

pspell_clear_session() remet à zéro la session courante. Le dictionnaire personnel est vidé, et par exemple si vous tentez de l'enregistrer avec **pspell_save_wordlist()**, rien ne se passera.

Exemple 1. Exemple avec pspell_add_to_personal()

```
<?php
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
$pspell_link = pspell_new_config ($pspell_config);
pspell_add_to_personal ($pspell_link, "Vlad");
pspell_clear_session ($pspell_link);
pspell_save_wordlist ($pspell_link); //"Vlad" ne sera pas sauvé
?>
```

pspell_config_create (PHP 4 >= 4.0.2)

Crée une configuration utilisée pour ouvrir un dictionnaire

```
resource pspell_config_create (string language, string [spelling], string [jargon], string [encoding])
```

pspell_config_create() a une syntaxe similaire à **pspell_new()**. En fait, utiliser **pspell_config_create()** suivi immédiatement par **pspell_new_config()** produira exactement le même résultat. Cependant, après avoir créer une nouvelle configuration, vous pouvez aussi utiliser les fonctions pspell_config_* avant d'appeler **pspell_new_config()** pour tirer profit des fonctionnalités avancées.

Le paramètre de langage est le code de langue en deux lettres, défini dans la norme ISO 639, et deux lettres optionnelles ISO 3166, après un tiret ou un souligné (_).

Le paramètre d'orthographe *spelling* est nécessaire pour les langues qui ont plus d'une orthographe, comme l'anglais. Les valeurs reconnues sont alors 'american' (américain) , 'british' (anglais), et 'canadian' (canadien).

Le paramètre de jargon *jargon* contient des informations supplémentaires pour distinguer deux dictionnaires distincts pour la même langue et le même paramètre d'orthographe *spelling*.

Le paramètre d'encodage indique l'encodage attendu pour la réponse. Les valeurs valides sont : 'utf-8', 'iso8859-*', 'koi8-r', 'viscii', 'cp1252', 'machine unsigned 16', 'machine unsigned 32'. Ce paramètre n'a pas été testé de manière exhaustive, alors soyez prudent.

Le paramètre de mode est le mode de travail du vérificateur d'orthographe. Plusieurs modes sont disponibles :

- PSPELL_FAST - Mode rapide (moins de suggestions, plus de vitesse)
- PSPELL_NORMAL - Mode normal mode (plus de suggestions)
- PSPELL_BAD_SPELLERS - Mode lent (beaucoup plus de suggestions, moins de vitesse)

Pour plus d'informations et d'exemples, vérifiez le manuel pspell sur leur site web :http://pspell.sourceforge.net/.

Exemple 1. Exemple avec pspell_config_create()

```
<?php
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl ($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link = pspell_new_personal (pspell_config);
?>
```

pspell_config_ignore (PHP 4 >= 4.0.2)

Ignore les mots des moins de N caractères.

```
int pspell_config_ignore (resource dictionary_link, int n)
```

pspell_config_ignore() doit être utilisé avec une configuration avant d'appeler **pspell_new_config()**. Cette fonction permet au vérificateur d'ignorer les mots trop courts.

Exemple 1. Exemple avec pspell_config_ignore()

```
<?php
$pspell_config = pspell_config_create ("en");
pspell_config_ignore($pspell_config, 5);
$pspell_link = pspell_new_config($pspell_config);
pspell_check($pspell_link, "abcd"); // Ce mot ne provoquera pas d'erreur
?>
```

pspell_config_mode (PHP 4 >= 4.0.2)

Change le mode de suggestion

```
int pspell_config_mode (resource dictionary_link, int mode)
```

pspell_config_mode() doit être appelé avant **pspell_new_config()**. Cette fonction détermine le nombre de suggestions qui seront retournés par **pspell_suggest()**.

Le paramètre de mode est le mode de travail du vérificateur d'orthographe. Plusieurs modes sont disponibles :

- PSPELL_FAST - Mode rapide (moins de suggestions, plus de vitesse)
- PSPELL_NORMAL - Mode normal mode (plus de suggestions)
- PSPELL_BAD_SPELLERS - Mode lent (beaucoup plus de suggestions, moins de vitesse)

Exemple 1. pspell_config_mode()

```
<?php
$pspell_config = pspell_config_create ("en");
pspell_config_mode($pspell_config, PSPELL_FAST);
$pspell_link = pspell_new_config($pspell_config);
pspell_check($pspell_link, "thecat");
?>
```

pspell_config_personal (PHP 4 >= 4.0.2)

Choisit le fichier qui contient le dictionnaire personnel

```
int pspell_config_personal (resource dictionary_link, string file)
```

pspell_config_personal() doit être appelé dans une configuration avant d'appeler **pspell_new_config()**. Le dictionnaire personnel sera chargé est utilisé en plus du dictionnaire standard, une fois que vous aurez appelé **pspell_new_config()**. Si le fichier n'existe pas, il sera créé. Ce fichier sera aussi le fichier où **pspell_save_wordlist()** sauvera le dictionnaire personnel. Ce fichier devra donc être accessible en écriture par PHP. Notez que cette fonction ne fonctionne pas avec les versions antérieures à pspell .11.2 et aspell .32.5.

Exemple 1. Exemple avec pspell_config_personal()

```
<?php
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
$pspell_link = pspell_new_config ($pspell_config);
pspell_check ($pspell_link, "thecat");
?>
```

pspell_config_repl (PHP 4 >= 4.0.2)

Choisit le fichier qui contient les paires de remplacement.

```
int pspell_config_repl (resource dictionary_link, string file)
```

pspell_config_repl() doit être appelé dans une configuration avant d'appeler **pspell_new_config()**. Les paires de remplacement améliorent la qualité du vérificateur. Lorsqu'un mot est mal orthographié et qu'aucune suggestion valable n'est trouvée dans le dictionnaire, **pspell_store_replacement()** sera utilisé pour enregistrer une paire de remplacement et **pspell_save_wordlist()** pour sauver le dictionnaire avec les paires de remplacement. Ce fichier devra donc être accessible en écriture par PHP. Notez que cette fonction ne fonctionne pas avec les versions antérieures à pspell .11.2 et aspell .32.5.

Exemple 1. Exemple avec pspell_config_repl()

```
<?php
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl ($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link = pspell_new_config ($pspell_config);
pspell_check ($pspell_link, "thecat");
?>
```

pspell_config_runtogether (PHP 4 >= 4.0.2)

Considère deux mots accolés comme un composé.

```
int pspell_config_runtogether (resource dictionary_link, boolean flag)
```

pspell_config_runtogether() doit être appelé dans une configuration avant d'appeler **pspell_new_config()**. Cette fonction indique si deux mots accolés doivent être traités comme un composé valide, même si il devrait y avoir un espace entre ces deux mots. Modifier cette configuration n'affecte que les résultats retournés par **pspell_check()**; **pspell_suggest()** retournera toujours des suggestions.

Exemple 1. Exemple avec pspell_config_runtogether()

```
<?php
$pspell_config = pspell_config_create ("en");
pspell_config_runtogether ($pspell_config, TRUE);
$pspell_link = pspell_new_config ($pspell_config);
pspell_check ($pspell_link, "thecat");
?>
```

pspell_config_save_repl (PHP 4 >= 4.0.2)

Active la sauvegarde des paires de remplacement

```
int pspell_config_save_repl (resource dictionary_link, boolean flag)
```

pspell_config_save_repl() doit être appelé dans une configuration avant d'appeler **pspell_new_config()**. Elle détermine si **pspell_save_wordlist()** doit sauver les paires de remplacement avec le dictionnaire. Généralement, il n'y a pas besoin d'utiliser cette fonction car si **pspell_config_repl()** est utilisée, les paires de remplacement seront sauvées de toutes façons, et si ce n'est pas le cas, elles ne seront pas sauvées. Ce fichier devra donc être accessible en écriture par PHP. Notez que cette fonction ne fonctionne pas avec les versions antérieures à pspell .11.2 et aspell .32.5.

pspell_new (PHP 4 >= 4.0.2)

Charge un nouveau dictionnaire

```
resource pspell_new (string language, string [spelling], string [jargon], string
[encoding])
```

pspell_new() ouvre un nouveau dictionnaire et retourne un identifiant de dictionnaire, pour utiliser avec d'autres fonctions pspell.

Le paramèegrave;tre de langue *spelling* est constitué des deux lettres du codage de langue ISO 639, et du codage optionnel de pays ISO 3166, séparé par un '_'.

Ce paramèegrave;tre est nécessaire pour les langues qui ont plus d'une orthographe, comme l'anglais ou le français. Les valeurs reconnues sont "americain", "britannique", et "canadien".

Le paramèegrave;tre de jargon contient des informations supplémentaires pour distinguer deux listes de mots qui ont le même marquage de langue et d'orthographe.

Le paramèegrave;tre d'encodage est le type d'encodage des mots. Les valeurs valides sont 'utf-8', 'iso8859-*', 'koi8-r', 'viscii', 'cp1252', 'machine unsigned 16', 'machine unsigned 32'.

Le paramètre de mode est le mode de travail du vérificateur d'orthographe. Plusieurs modes sont disponibles :

- PSPELL_FAST - Mode rapide (moins de suggestions, plus de vitesse)
- PSPELL_NORMAL - Mode normal mode (plus de suggestions)
- PSPELL_BAD_SPELLERS - Mode lent (beaucoup plus de suggestions, moins de vitesse)
- PSPELL_RUN_TOGETHER - Considère que des mots accolés forment un composé autorisé. C'est à dire que "lechat" sera un composé valide. Cette option ne modifie que les résutlat retournés par **pspell_check()**; **pspell_suggest()** retournera toujours les mêmes suggestions.

Mode est un champs de bit, construits à partir des constantes listées ci dessus. Cependant, PSPELL_FAST, PSPELL_NORMAL et PSPELL_BAD_SPELLERS sont mutuellement exclusives : vous ne devez en utiliser qu'une seule en même temps.

Pour plus d'informations et d'exemples, reportez vous au site http://pspell.sourceforge.net/ (en anglais).

Exemple 1. pspell_new()

```
<?php
$pspell_link = pspell_new("en", "", "", "", (PSPELL_FAST|PSPELL_RUN_TOGETHER));
?>
```

pspell_new_config (PHP 4 >= 4.0.2)

Charge un nouveau dictionnaire

```
resource pspell_new_config (resource config)
```

pspell_new_config() ouvre un nouveau dictionnaire et charge les paramètrages spécifié dans la configuration *config*, créée avec **pspell_config_create()** et modifiée avec les fonctions pspell_config_*. Cette méthode vous donne le maximum de flexibilité, et dispose de toutes les fonctionnalités fournies par **pspell_new()** et **pspell_new_personal()**.

Le paramètre de configuration est celui qui a été retourné par **pspell_config_create()** lors de création de la configuration.

Exemple 1. pspell_new_config()

```
<?php
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl ($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link = pspell_new_personal (pspell_config);
?>
```

pspell_new_personal (PHP 4 >= 4.0.2)

Charge un nouveau dictionnaire avec un dictionnaire personnel

```
resource pspell_new_personal (string personal, string language, string [spelling], string
[jargon], string [encoding], int [mode])
```

pspell_new_personal() charge un nouveau dictionnaire avec un dictionnaire personnel, et retourne un identifiant de dictionnaire utilisé par d'autres fonctions pspells. Le dictionnaire peut être modifiée et sauvé avec **pspell_save_wordlist()**. Cependant, les paires de remplacement ne seront pas sauvées. Pour ce faire, vous devez créer une configuration qui utilise **pspell_config_create()**, et choisir le fichier de destination du dictionnaire personnel avec **pspell_config_personal()**,

choisir le fichier de paire de remplacement avec **pspell_config_repl()**, et ouvrir un nouveau dictionnaire avec **pspell_new_config()**.

Le paramètre *`personal`* spécifie le fichier où seront ajoutés les mots du dictionnaire personnel. Ce doit être un chemin absolu, qui commence par '/' car sinon, il sera relatif à $HOME, qui est "/root" sur la pluspart des systèmes, et probablement pas ce que vous souhaitez.

Le paramètre de langage est le code de langue en deux lettres, défini dans la norme ISO 639, et deux lettres optionnelles ISO 3166, après un tiret ou un souligné (_).

Le paramètre d'orthographe *`spelling`* est nécessaire pour les langues qui ont plus d'une orthographe, comme l'anglais. Les valeurs reconnues sont alors 'american' (américain) , 'british' (anglais), et 'canadian' (canadien).

Le paramètre de jargon *`jargon`* contient des informations supplémentaires pour distinguer deux dictionnaires distincts pour la même langue et le même paramètre d'orthographe *`spelling`*.

Le paramètre d'encodage indique l'encodage attendu pour la réponse. Les valeurs valides sont : 'utf-8', 'iso8859-*', 'koi8-r', 'viscii', 'cp1252', 'machine unsigned 16', 'machine unsigned 32'. Ce paramètre n'a pas été testé de manière exhaustive, alors soyez prudent.

Le paramètre de mode est le mode de travail du vérificateur d'orthographe. Plusieurs modes sont disponibles :

- PSPELL_FAST - Mode rapide (moins de suggestions, plus de vitesse)
- PSPELL_NORMAL - Mode normal mode (plus de suggestions)
- PSPELL_BAD_SPELLERS - Mode lent (beaucoup plus de suggestions, moins de vitesse)

Pour plus d'informations et d'exemples, vérifiez le manuel pspell sur leur site web :http://pspell.sourceforge.net/.

Exemple 1. Exemple avec pspell_new_personal()

```
<?php
$pspell_link = ps-
pell_new_personal ("/var/dictionaries/custom.pws", "en", "", "", "", PSPELL_FAST|PSPELL_RUN_TOGETH
?>
```

pspell_save_wordlist (PHP 4 >= 4.0.2)

Sauve le dictionnaire personnel dans un fichier.

```
int pspell_save_wordlist (resource dictionary_link)
```

pspell_save_wordlist() sauve le dictionnaire personnel de la session courante. Le dictionnaire doit avoir été ouvert avec **pspell_new_personal()**, et la localisation des fichiers doit avoir été spécifié avec **pspell_config_personal()** et (éventuellement) **pspell_config_repl()**. Notez que cette fonction n'est pas disponible avec les versions antérieures à pspell .11.2 et aspell .32.5.

Exemple 1. Exemple pspell_add_to_personal()

```
<?php
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/tmp/dicts/newdict");
$pspell_link = pspell_new_config ($pspell_config);
pspell_add_to_personal ($pspell_link, "Vlad");
pspell_save_wordlist ($pspell_link);
?>
```

pspell_store_replacement (PHP 4 >= 4.0.2)

Enregistre une paire de remplacement pour un mot

```
int pspell_store_replacement (resource dictionary_link, string misspelled, string correct)
```

pspell_store_replacement() enregistre une paire de remplacement pour un mot de façon à ce que cette suggestion soit retournée par **pspell_suggest()** plus tard. Pour pouvoir utiliser cette fonction, vous devez utiliser **pspell_new_personal()** pour ouvrir le dictionnaire. Pour pouvoir sauver tout le temps les paires de remplacement, vous devez utiliser **pspell_config_personal()** et **pspell_config_repl()** pour indiquer le lieu de sauvegarde des dictionnaires personnels, et **pspell_save_wordlist()** pour enregistrer les modifications sur le disque. Ce fichier devra donc être accessible en écriture par PHP. Notez que cette fonction ne fonctionne pas avec les versions antérieures à pspell .11.2 et aspell .32.5.

Exemple 1. Exemple avec pspell_store_replacement()

```
<?php
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl ($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link = pspell_new_config ($pspell_config);
pspell_store_replacement ($pspell_link, $misspelled, $correct);
pspell_save_wordlist ($pspell_link);
?>
```

pspell_suggest (PHP 4 >= 4.0.2)

Suggèegrave;re une orthographe

```
array pspell_suggest (resource dictionary_link, string word)
```

pspell_suggest() retourne un tableau de suggestions pour le mot *word*.

Exemple 1. pspell_suggest()

```
<?php
$pspell_link = pspell_new ("english");
if (!pspell_check ($pspell_link, "testt")){
    $suggestions = pspell_suggest ($pspell_link, "testt");
    for ($i=0; $i < count ($suggestions); $i++) {
        echo "Orthographes suggerées : " . $suggestions[$i] . "<br>";
    }
}
?>
```

LXXI. Readline (GNU)

Les fonctions **readline()** implémente une interface avec la librairie GNU Readline. Ces fonctions fournissent une ligne de commande éditable, un peu comme lorsque Bash vous permet d'utiliser les flèches de déplacement pour insérer un caractère ou passer en revue l'historique. A cause de l'interactivité de ces commande, elles ne seront que rarement utiles pour les applications Web, mais peuvent se révéler utiles lorsqu'un script est exécuté depuis une commande shell.

Le site du projet GNU Readline est http://cnswww.cns.cwru.edu/~chet/readline/rltop.html. Elle est entretenue par Chet Ramey, qui est aussi l'auteur de Bash.

readline (PHP 4 >= 4.0b4)

Lit une ligne

```
string readline ([string prompt])
```

readline() retourne une ligne entrée par l'utilisateur. Vous pouvez spécifier une chaîne de prompt. La ligne retournée est débarassée du caractère nouvelle ligne final. Vous devez ajouter cette ligne à l'historique vous-même, avec la fonction **readline_add_history()**.

Exemple 1. Exemple avec readline()

```
<?php
//Lit 3 commandes de l'utilisateur
for ($i=0; $i < 3; $i++) {
        $line = readline("Commande: ");
        readline_add_history($line);
}
//liste l'historique
print_r(readline_list_history());
//liste les variables
print_r(readline_info());
?>
```

readline_add_history (PHP 4 >= 4.0b4)

Ajoute une ligne à l'historique

```
void readline_add_history (string line)
```

readline_add_history() ajoute une ligne à l'historique.

readline_clear_history (PHP 4 >= 4.0b4)

Efface l'historique

```
boolean readline_clear_history (void )
```

readline_clear_history() efface tout l'historique.

readline_completion_function (PHP 4 >= 4.0b4)

Enregistre une fonction de complétion

```
boolean readline_completion_function  (string line)
```

readline_completion_function() enregistre une nouvelle fonction de complétion. Vous devez fournir le nom d'une fonction qui accepte un nom partiel de commande, et retourne une liste de fonctions complète possibles. C'est la même fonctionnalité que lorsque vous utilisez la touche de tabulation sous Bash.

readline_info (PHP 4 >= 4.0b4)

Lit/modifie diverses variables internes

```
mixed readline_info ([string varname [, string newvalue]])
```

Appelée sans paramètre, **readline_info()** retourne une tableau contenant les valeurs des paramètres de Readline. Les éléments seront indexés par les clés suivantes : done, end, erase_empty_line, library_version, line_buffer, mark, pending_input, point, prompt, readline_name, et terminal_name.

Appelée avec le paramètre *varname*, la valeur de cette variable sera retournée. Appelée avec deux paramètres, et la valeur de la variable *varname*, sera remplacée par *newvalue*.

readline_list_history (PHP 4 >= 4.0b4)

Liste l'historique

```
array readline_list_history (void )
```

readline_list_history() retourne un tableau avec la liste de toutes les lignes de commandes de l'historique. Les éléments sont indéxés numériquement, à partir de 0.

readline_read_history (PHP 4 >= 4.0b4)

Lit l'historique

```
boolean readline_read_history  (string filename)
```

readline_read_history() lit une ligne de l'historique.

readline_write_history (PHP 4 >= 4.0b4)

Ecrit dans l'historique

```
boolean readline_write_history  (string filename)
```

readline_write_history() écrit *filename* dans l'historique.

LXXII. Recode (GNU)

Ce module contient l'interface à la librairie GNU Recode library, version 3.5. Pour pouvoir utiliser ces fonctions, il faut que PHP ait été compilé avec l'option --with-recode. Pour cela, il faut que vous ayez la librairie GNU Recode 3.5 ou plus récent, installée sur votre système.

La librairie GNU Recode library convertit les fichiers ayant des jeux de caractères différents. Lorsque ce n'est pas possible, elle se débarasse des caractères illégaux, ou bien effectue une approximation. La librairie reconnait ou produit près de 150 jeux de caractères différents, et peut quasiment tous les convertir de l'un vers l'autre. La plus part des jeux de caractères de la RFC 1345 sont supportés.

recode_string (PHP 3>= 3.0.13, PHP 4 >= 4.0RC1)

Recode une chaîne en fonction de la requête.

```
string recode_string (string request, string string)
```

recode_string() recode la chaîne *string* en fonction de la requête *request*. **recode_string()** retourne FALSE, en cas d'échec, et TRUE sinon.

Une requête simple de recodage peut être "lat1..iso646-de". Reportez vous à la documentation GNU Recode de votre installation pour plus de détails sur les requêtes.

Exemple 1. Exemple simple avec recode_string()

```
<?php
print recode_string ("us..flat", "Le caractère suivant est diacritique : &aacute;");
?>
```

recode (PHP 4 >= 4.0RC1)

Recode une fonction grâce à une requête

```
string recode (string request, string string)
```

Note : recode() est un alias de **recode_string()**. Elle a été ajoutée en PHP 4.

recode_file (PHP 3>= 3.0.13, PHP 4 >= 4.0RC1)

Recode de fichier à fichier, en fonction de la requête.

```
boolean recode_file (string request, resource input, resource output)
```

recode_file() recode le fichier identifié par *input* dans le fichier identifié par *output* en fonction de la requête de recodage *request*. Retourne FALSE, en cas d'échec, et TRUE sinon.

recode_file() ne gère pas encore les fichiers distants (URLs). Les deux fichiers doivent faire référence à des fichiers locaux.

Exemple 1. Exemple simple avec recode_file()

```
<?php
$input = fopen('input.txt', 'r');
$output = fopen('output.txt', 'w');
recode_file("us..flat", $input, $output);
?>
```

LXXIII. Expressions régulières compatibles Perl

La syntaxe des masques utilisés dans ces fonctions ressemble fort à celle de Perl. Les expressions seront entourées de délimiteurs, slash (/), par exemple. N'importe quel caractère peut servir de délimiteur, tant qu'il n'est pas alpha-numérique ou n'est pas un antislash (\). Si un délimiteur doit être utilisé dans l'expression, il faudra l'échapper avec un antislash. Depuis PHP 4.0.4, vous pouvez utiliser les délimiteurs (), {}, [], et <>, comme en Perl.

Le délimiteur final peut être suivi d'options qui affecteront la recherche. Voir aussi options de recherche.

Exemple 1. Exemples de masques valides

- `/<\/\w+>/`
- `|(\d{3})-\d+|Sm`
- `/^(?i)php[34]/`
- `{^\s+(\s+)?$}`

Exemple 2. Exemples de masques invalides

- /href='(.*)' - délimiteur final manquant
- /\w+\s*\w+/J - option 'J' inconnue
- 1-\d3-\d3-\d4| - délimiteur initial manquant

Note : Les expressions régulières Perl sont disponibles depuis la PHP 4 et PHP 3.0.9.

Le support des expressions régulières est assuré par la librairie PCRE, qui est open source, et écrite par Philip Hazel. Elle est soumise au copyright de l'University of Cambridge, Angleterre. Elle est disponible à ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/.

preg_match (PHP 3>= 3.0.9, PHP 4 >= 4.0b1)

Expression régulière standard.

```
int preg_match (string pattern, string subject [, array matches])
```

preg_match() analyse *subject* pour trouver l'expression *pattern*.

Si *matches* est fourni, il sera rempli par les résultats de la recherche. $matches[0] contiendra le texte qui satisfait le masque complet, $matches[1] contiendra le texte qui satisfait la première parenthèse capturante, etc..

preg_match() retourne TRUE si la recherche réussit, et FALSE sinon (notamment en cas d'erreur).

Exemple 1. Extraction d'un numéro de page d'une chaîne.

```
<?php
if (preg_match("/page\s+#(\d+)/i", "Aller à la page numéro 9.", $parts))
    print "La page suivante est $parts[1]";
else
    print "Page introuvable.";
?>
```

Exemple 2. Trouve le mot "web"

```
<?php
// \b, dans le masque, indique une limite de mot, de façon à ce que le mot
// "web" uniquement soit repéré, et pas seulement des parties de mots comme
// dans "webbing" ou "cobweb"
if (preg_match ("/\bweb\b/i", "PHP est le meilleur langage de script du web.")) {
    print "Un mot a été trouvé.";
} else {
    print "Un mot n'a pas été trouvé.";
}
if (preg_match ("/\bweb\b/i", "PHP est le meilleur langage de script pour les weba-
gency.")) {
     print "Un mot a été trouvé.";
} else {
    print "Un mot n'a pas été trouvé.";
}
?>
```

Exemple 3. Lire un nom de domaine dans une URL

```
<?php
// repérer le nom de l'hôte dans l'URL
preg_match("/^(http:\/\/)?([^\/]+)/i",
"http://www.php.net/index.html", $matches);
$host = $matches[2];
// repérer les deux derniers segments du nom de l'hôte
preg_match("/[^\.\/]+\.[^\.\/]+$/",$host,$matches);
echo "Le nom de domaine est : ".$matches[0]."\n";
?>
```

Cet exemple va afficher : `Le nom de domaine est : php.net` Voir aussi **preg_match_all()**, **preg_replace()** et **preg_split()**.

preg_match_all (PHP 3>= 3.0.9, PHP 4 >= 4.0b1)

Expression régulière globale.

```
int preg_match_all (string pattern, string subject, array matches [, int order])
```

preg_match_all() analyse *subject* pour trouver l'expression *pattern* et met les résultats dans *matches*, dans l'ordre spécifié par *order*.

Après avoir trouvé un premier résultat, la recherche continue jusqu'à la fin de la chaîne.

order peut prendre une des deux valeurs suivantes :

PREG_PATTERN_ORDER

L'ordre est tel que $matches[0] est un tableau qui contient les résultats qui satisfont le masque complet, $matches[1] est un tableau qui contient les résultats qui satisfont la première parenthèse capturante, etc..

```
<?php
preg_match_all("|<[^>]+>(.*)</[^>]+>|U", "<b>example: </b><div align=left>a test</div>", $out,
print $out[0][0].", ".$out[0][1]."\n";
print $out[1][0].", ".$out[1][1]."\n";
?>
```

Cet exemple va afficher :

```
<b>exemple: </b>, <div align=left>ceci est un test</div>
exemple: , ceci est un test
```

Ainsi, $out[0] est un tableau qui contient les résultats qui satisfont le masque complet, et $out[1] est un tableau qui contient les balises entre > et <.

PREG_SET_ORDER

Les résultats sont classés de telle façon que $matches[0] contient la première série de résultat, $matches[1] contient la deuxième série de résultat, etc...

```
<?php
preg_match_all("|<[^>]+>(.*)</[^>]+>|U", "<b>exemple: </b><div align=left>un test</div>", $out,
print $out[0][0].", ".$out[0][1]."\n";
print $out[1][0].", ".$out[1][1]."\n";
?>
```

Cet exemple va afficher :

```
<b>exemple: </b>, exemple:
<div align=left>un test</div>, un test
```

Dans ce cas, $matches[0] est la première série de résultat, et $matches[0][0] contient le texte qui satisfait le masque complet, $matches[0][1] contient le texte de la première parenthèse capturante, etc... De même, $matches[1] contient le texte qui satisfait le masque complet, etc...

Si *order* est omis, PREG_PATTERN_ORDER est utilisé par défaut.

preg_match_all() retourne le nombre de résultat qui satisfont le masque complet, ou FALSE en cas d'échec ou d'erreur.

Exemple 1. Extraction de tous les numéros de téléphone d'un texte.

```
<?php
  preg_match_all("/\(?  (\d{3})?  \)?  (?(1)  [\-\s] ) \d{3}-\d{4}/x",
              "Appelez 555-1212 ou 1-800-555-1212", $phones);
?>
```

Exemple 2. Recherche les couples de balises HTML (gourmand)

```
<?php
// Cet exemple utilise les références arrières (\\2).
// Elles indiquent à l'analyseur qu'il doit trouver quelquechose qu'il
// a déjà repéré un peu plus tôt
// le nombre 2 indique que c'est le deuxième jeu de parenthèses
// capturante qui doit être utilisé (ici, ([\w]+)).
//  L'antislash est nécessaire ici, car la chaîne est entre guillemets doubles
$html = "<B>Texte en gras</B><a href=salut.html>clique moi</?>
preg_match_all ("/(<([\w]+)[?>]?>)(.*)(<\/\\?>)/", $html, $matches);
for ($i=0; $i< count($matches[0]); $i++) {
  echo "trouvé: ".$matches[0][$i]."\n";
  echo "partie 1: ".$matches[1][$i]."\n";
  echo "partie 2: ".$matches[3][$i]."\n";
  echo "partie 3: ".$matches[4][$i]."\n\n";
}
?>
```

Cet exemple va produire : `trouvé: <B>bold text</?> partie 1: <B> partie 2: Test en gras partie 3: </B> trouvé: <a href=salut.html>clique moi</?> partie 1: <a href=salut.html> partie 2: clique moi partie 3: </?>`

Voir aussi **preg_match()**, **preg_replace()** et **preg_split()**.

preg_replace (PHP 3>= 3.0.9, PHP 4 >= 4.0b1)

Rechercher et remplacer par expression régulière standard.

```
mixed preg_replace (mixed pattern, mixed replacement, mixed subject [, int limit])
```

preg_replace() analyse *subject* pour trouver l'expression *pattern* et remplace les résultats par *replacement*.

replacement peut contenir des références de la forme *n* ou, depuis PHP 4.0.4) *$n*. Cette dernière forme est recommandée. Ces références seront remplacées par le texte capturé par la *n*'-ième parenthèse capturante du masque. *n* peut prendre des valeurs de 0 à 99, et \\0 ou $0, correspondent au texte de qui satisfait le masque complet. Les parenthèses ouvrantes sont comptées de gauche à droite (en commençant à 1) pour déterminer le numéro de parenthèse capturante.

Si la recherche n'aboutit à aucun résultat, *subject* sera inchangé.

Tous les paramètres de **preg_replace()** peuvent être des tableaux.

Si *subject* est un tableau, alors l'opération sera appliquée à chacun des éléments du tableau, et le tableau sera retourné.

Si *pattern* et *replacement* sont des tableaux, alors **preg_replace()** prend une valeur de chaque tableau, et l'utilise pour faire la recherche et le remplacement. Si *replacement* à moins d'éléments que *pattern*, alors la chaîne vide est utilisé pour le reste des valeurs. Si *pattern* est un tableau, et que *replacement* est une chaîne, alors cette chaîne sera utilisée pour chaque valeur de *pattern*. Le contraire n'aurait pas de sens.

/e force **preg_replace()** à traiter *replacement* comme du code PHP une fois que les substitutions adéquates ont été faites. Conseil :assurez-vous que *replacement* est un code PHP valide, car sinon, PHP trouvera une erreur d'analyse (parse error) dans cette ligne.

`/F` indique que le paramètre *`replacement`* doit être considéré comme un nom de fonction. Cette fonction sera appelée, avec un tableau contenant les éléments trouvés comme arguments. La fonctoin doit retourner la chaine de remplacement. Cette option a été ajoutée en PHP 4.0.4.

Exemple 1. Remplacement de plusieurs valeurs

```
<?php
$patterns = array ("/(19|20)(\d{2})-(\d{1,2})-(\d{1,2})/",
                   "/^\s*{(\w+)}\s*=/");
$replace = array ("\\3/\\4/\\1\\2", "$\\1 =");
print preg_replace ($patterns, $replace, "{startDate} = 1999-5-27");
?>
```

Cet exemple va afficher : `$startDate = 5/27/1999`

Exemple 2. Utilisation de l'option /e

```
<?php
  preg_replace("/(<\/?)(\w+)([^>]*>/e", "'\\1'.strtoupper('\\2').'\\3'", $html_body);
?>
```

Cela va mettre en majuscule toutes les balises HTML du texte.

Exemple 3. Conversion HTML en texte

```
<?php
// $document contient un document HTML
// Ce script va effacer les balises HTML, les javascript
// et les espaces. Il remplace aussi quelques entités HTML
// courante en leur équivalent texte.
$search = array ("'<script[?>]*?>.*?</script>'si",  // Supprime le javascript
                "'<[\/\!]*?[^<?>]*?>'si",  // Supprime les balises HTML
                "'([\r\n])[\s]+'",  // Supprime les espaces
                "'&(quot|#34);'i",  // Supprime les entites HTML
                "'&(amp|#38);'i",
                "'&(lt|#60);'i",
                "'&(gt|#62);'i",
                "'&(nbsp|#160);'i",
                "'&(iexcl|#161);'i",
                "'&(cent|#162);'i",
                "'&(pound|#163);'i",
                "'&(copy|#169);'i",
                "'&#(\d+);'e");  // Evaluation comme PHP
$replace = array ("",
                  "",
                  "\\1",
                  "\"",
                  "&",
                  "<",
                  ">",
                  " ",
                  chr(161),
                  chr(162),
                  chr(163),
                  chr(169),
                  "chr(\\1)");
$text = preg_replace ($search, $replace, $document);
?>
```

Note : Le paramètre *`limit`* a été ajouté à partir de PHP 4.0.1pl2.

Voir aussi **preg_match()**, **preg_match_all()** et **preg_split()**.

preg_replace_callback (PHP 4 >= 4.0.5)

Rechercher/remplacer avec fonction de callback

```
mixed preg_replace_callback (mixed pattern, mixed callback, mixed subject [, int limit])
```

Le comportement de **preg_replace()** est presque identique à celui de **preg_replace()**, hormis le fait qu'à la place du paramètre *replacement*, il faut spécifier une fonction de callback *callback* qui sera appelée, avec les éléments trouvés en arguments. Cette fonction retourne alors la chaîne de remplacement.

preg_replace_callback() a été ajoutée en PHP 4.0.5.

Voir aussi **preg_replace()**.

preg_split (PHP 3>= 3.0.9, PHP 4 >= 4.0b1)

Eclatement d'une chaîne par expression régulière.

```
array preg_split (string pattern, string subject [, int limit [, int flags]])
```

preg_split() retourne un tableau contenant les sous-chaînes de *subject*, séparées par les chaînes qui vérifient *pattern*.

Si *limit* est spécifié, alors seules les *limit* premières sous-chaînes sont retournées et si *limit* vaut -1, cela signifie en fait "sans limite", ce qui est utile pour passer le paramètre *flags*.

flags peut être la combinaison des options suivantes (combinées avec l'opérateur |):

PREG_SPLIT_NO_EMPTY

Si cette option est activée, seules les sous-chaînes non vides seront retournées par **preg_split()**.

PREG_SPLIT_DELIM_CAPTURE

Si cette option est activée, les expressions entre parenthèses entre les délimiteurs de masques seront aussi capturées et retournées. Cette option a été ajoutée en PHP 4.0.5.

Note : Le paramètre *flags* a été ajouté en PHP Beta 3.

Exemple 1. Eclatement d'une chaîne de recherche.

```
<?php
// scinde la phrase grâce aux virgules et espacements
// ce qui inclus les " ", \r, \t, \n et \f
$keywords = preg_split ("/[\s,]+/", "langage hypertexte, programmation");
?>
```

Exemple 2. Scinder une chaîne en caractères

```
<?php
$str = 'string';
$chars = preg_split('//', $str, -1, PREG_SPLIT_NO_EMPTY);
print_r($chars);
?>
```

Voir aussi **explode()**, **spliti()**, **split()**, **implode()**, **preg_match()**, **preg_match_all()** et **preg_replace()**.

preg_quote (PHP 3>= 3.0.9, PHP 4 >= 4.0b1)

Echappement des caractères spéciaux des expressions régulières.

```
string preg_quote (string str [, string delimiter])
```

preg_quote() ajoute un antislash devant tous les caractères de la chaîne *str*. Cela est très utile si vous avez une chaîne qui va servir de masque, mais qui est générée durant l'exécution.

Si l'argument optionnel *delimiter* est fourni, il sera aussi échappé. Ceci est pratique pour échapper le délimiteur requis par les fonctions PCRE. Le slash / est le délimiteur le plus répandu.

Les caractères spéciaux qui seront échappés :

```
. \\ + * ? [ ^ ] $ ( ) { } = ! < > | :
```

Exemple 1. Protège des caractères spéciaux

```
<?php
$keywords = "$40 pour un g3/400";
$keywords = preg_quote ($keywords, "/");
echo $keywords; // retourne \$40 pour un g3\/400
?>
```

Exemple 2. Mise en italique d'un mot dans un texte

```
<?php
// Dans cet exemple, preg_quote($word) sert à éviter que les astérisques
// prennent une valeur particulière dans l'exepression régulière.
$textbody = "Ce livre est *très* difficile à trouver.";
$word = "*très*";
$textbody = preg_replace ("/".preg_quote($word)."/",
                          "<B>".$word."</B>",
                          $textbody);
?>
```

preg_grep (PHP 4 >= 4.0b1)

Retourne un tableau avec les résultat de la recherche.

```
array preg_grep (string pattern, array input)
```

preg_grep() retourne un tableau qui contient les éléments de *input* qui satisfont le masque *pattern*.

Depuis PHP 4.0.4, le tableau retourné par **preg_grep()** est indexé en utilisant les clés issues du tableau *input*. Si ces clés sont inutiles, utilisez la fonction **array_values()** sur le tableau retourné par **preg_grep()** pour obtenir le comportement traditionnel.

Exemple 1. Exemple avec preg_grep()

```
<?php
// recherche les nombres à virgule flottante
preg_grep("/^(\d+)?\.\d+$/", $array);
?>
```

options de recherche (unknown)

Options disponibles pour les expressions régulières.

Les options de PCRE sont listées ci-dessous. Les noms entre parenthèses sont les noms internes à PCRE.

i (PCRE_CASELESS)

Effectue une recherche insensible à la casse.

m (PCRE_MULTILINE)

Par défaut, PCRE traite la chaîne sujet comme une seule ligne (même si cette chaîne contient des retours chariot). Le méta-caractère "début de ligne" (^) ne sera valable qu'une seule fois, au début de la ligne, et le méta caractère "fin de ligne " ($) ne sera valable qu'à la fin de la chaîne, ou avant le retour chariot final (à moins que l'option D ne soit activée). C'est le même fonctionnement qu'en Perl.

Lorsque cette option est activée, " début de ligne " et " fin de ligne " correspondront alors aux caractères suivant et précédent immédiatement un caractère de nouvelle ligne, en plus du début et de la fin de la chaîne. C'est le même fonctionnement que l'option Perl /m. S'il n'y a pas de caractère de nouvelle ligne "\n" dans la chaîne sujet, ou s'il n'y a aucune occurrence de ^ ou $ dans le masque, cette option ne sert à rien.

s (PCRE_DOTALL)

Avec cette option, le méta caractère point (.) remplace n'importe quel caractère, y compris les nouvelles lignes. Sans cette option, le caractère point ne remplace pas les nouvelles lignes. Cette option est équivalente à l'option Perl /s. Une classe de caractères négative telle que [^a] acceptera toujours les caractères de nouvelles lignes, indépendamment de cette option.

x (PCRE_EXTENDED)

Avec cette option, les caractères d'espacement sont ignorés, sauf lorsqu'ils sont échappés, ou à l'intérieur d'une classe de caractères, et tous les caractères entre # non échappés et en dehors d'une classe de caractères, et le prochain caractère de nouvelle ligne sont ignorés. C'est l'équivalent Perl de l'option /x : elle permet l'ajout de commentaires dans les masques compliqués. Notez bien, cependant, que cela ne s'appliquent qu'aux caractères de données. Les caractères d'espacement ne doivent jamais apparaître dans les séquences spéciales d'un masque, comme par exemple dans la séquence (?(qui introduit une parenthèse conditionnelle.

E

Avec cette option, **preg_replace()** effectue la substitution normale des références arrières dans la chaîne de remplacement, puis l'évalue comme un code PHP, et utilise le résultat pour remplacer la chaîne de recherche.

Seule **preg_replace()** utilise cette option. Elle est ignorée par les autres.

A (PCRE_ANCHORED)

Avec cette option, le masque est ancré de force, c'est-à-dire que le masque doit s'appliquer entre le début et la fin de la chaîne sujet pour être considéré comme trouvé. Il est possible de réaliser le même effet en ajoutant les méta-caractères adéquats, ce qui est la seule manière de le faire en Perl.

E (PCRE_DOLLAR_ENDONLY)

Avec cette option, le méta-caractère $ ne sera valable qu'à la fin de la chaîne sujet. Sans cette option, $ est aussi valable avant une nouvelle ligne, si cette dernière est le dernier caractère de la chaîne. Cette option est ignorée si l'option *m* est activée. Il n'y a pas d'équivalent en Perl.

S

Lorsqu'un masque est utilisé plusieurs fois, cela vaut la peine de passer quelques instants de plus pour l'analyser et optimiser le code pour accélérer les traitements ultérieurs. Cette option force cette analyse plus poussée. Actuellement, cette analyse n'est utile que pour les masques non ancrés, qui ne commencent pas par un caractère fixe.

U (PCRE_UNGREEDY)

Cette option inverse la tendance à la gourmandise des expressions régulières. Vous pouvez aussi inverser cette tendance au coup par coup avec un ?. De même, si cette option est activée, le ? rendra gourmand une séquence. Cette option n'est pas compatible avec Perl. Elle peut aussi être mise dans le masque avec l'option ?U.

X (PCRE_EXTRA)

Cette option ajoute d'autres fonctionnalités incompatible avec le PCRE de Perl. Tous les antislash suivis d'une lettre qui n'aurait pas de signification particulière cause une erreur, permettant la réservation de ces combinaisons pour des ajouts fonctionnels ultérieurs. Par défaut, comme en Perl, les antislash suivis d'une lettre sans signification particulière sont traités comme des valeurs littérales. Actuellement, cette option ne déclenche pas d'autres fonctions.

u (PCRE_UTF8)

Cette option inactive les fonctionnalités additionnelles de PCRE qui ne sont pas compatibles avec Perl. Les chaînes sont traitées comme des chaînes UTF-8. Cette option est disponible en PHP 4.0.7 et plus récent.

syntaxe des masques (unknown)

Fonctionnement des expressions régulières.

La bibliothèque PCRE est un ensemble de fonctions qui implémentent la recherche par expressions régulières, en utilisant la même syntaxe et la même sémantique que le Perl 5, avec quelques nuances (voir ci-dessous). L'implémentation actuelle est celle de Perl 5.005.

Les différences avec le Perl 5.005 sont présentée ici :

1. Par défaut, un caractère d'espacement correspond à n'importe quel caractère que la fonction C isspace() reconnaît, bien qu'il soit possible de recompiler la bibliothèque PCRE avec d'autres tables de caractères. Normalement, isspace() retourne TRUE pour les espaces, les retours chariot, les nouvelles lignes, les formfeed, les tabulations verticales et horizontales. Le Perl 5 n'accepte plus la tabulation verticale comme caractère d'espacement. La séquence \v qui était dans la documentation Perl depuis longtemps n'a jamais été reconnue. Cependant, la tabulation verticale elle-même était reconnue comme un caractère d'espacement jusqu'à la version 5.002. Avec les version 5.004 et 5.005, l'option \s l'ignore.
2. PRCE ne tolère pas la répétition de quantificateurs dans les expressions. Perl le permet, mais cela ne signifie pas ce que vous pourriez penser. Par exemple, (?!a){3} ne s'interprète pas : les trois caractères suivants ne sont pas des "a". En fait, cela s'interprète comme : le caractère suivant n'est pas "a" trois fois.
3. Les occurrences de sous-masques qui interviennent dans des assertions négatives sont comptées, mais elles ne sont pas enregistrées dans le vecteur d'occurrences. Perl modifie ses variables numériques pour toutes les occurrences de sous-masque, avant que l'assertion ne vérifie le masque entier, et uniquement si les sous-masques ne trouvent qu'une seule occurrence.
4. Bien que les caractères nul soient tolérés dans la chaîne de recherche, ils ne sont pas acceptés dans le masque, car le masque est utilisé comme une chaîne C standard, terminée par le caractère nul. Il faut donc utiliser la séquence d'échappement "\0" dans le masque pour rechercher les caractères nul.
5. Les séquence d'échappement suivantes ne sont pas supportées par le Perl: \l, \u, \L, \U, \E, \Q. En fait, elles sont implémentées par la gestion intrinsèque de chaînes du Perl, et ne font pas partie de ses caractères spéciaux.
6. L'assertion \G du Perl n'est pas supportée car elle n'est pas pertinente pour faire des recherches avec des masques uniques.

7. De manière assez évidente, PCRE n’accepte pas la construction (?{code}).
8. Au moment de l’écriture de PCRE, Perl 5.005_02 avait quelques comportements étranges avec la capture des chaînes lorsqu’une partie du masque est redoublée. Par exemple, "aba" avec le masque /^(a(b)?)+$/ va affecter à $2 la valeur "b", mais la même manipulation avec "aabbaa" et /^(aa(bb)?)+$/ laissera $2 vide. Cependant, si le masque est remplacé par /^(aa(b(b))?)+$/ alors $2 (et d’ailleurs $3) seront correctement affectés. Avec le Perl 5.004, $2 sera correctement affecté dans les deux cas, et c’est aussi vrai avec PCRE. Si Perl évolue vers un autre comportement cohérent, PCRE s’adaptera probablement.
9. Une autre différence encore non résolue est le fait qu’en Perl 5.005_02 le masque /^(a)?(?(1)a|b)+$/ accepte la chaîne "a", tandis que PCRE ne l’accepte pas. Cependant, que ce soit avec Perl ou PCRE /^(a)?a/ et "a" laisseront $1 vide.
10. PCRE propose quelques extensions aux expressions régulières du Perl.
    a. (a) Bien que les assertions avec retour (lookbehind) soit obligée d’apparier une chaîne de longueur fixe, toutes les assertions avec retour peuvent avoir une longueur différente. Perl 5.005 leur impose d’avoir toutes la même longueur.
    b. (b) Si PCRE_DOLLAR_ENDONLY est activé, et que PCRE_MULTILINE n’est pas activé, le méta caractère $ ne s’applique qu’à la fin physique de la chaîne, et non pas avant les caractères de nouvelle ligne.
    c. (c) Si PCRE_EXTRA est activé, un antislash suivi d’une lettre sans signification spéciale est considérée comme une erreur.
    d. (d) SI PCRE_UNGREEDY est activé, la "gourmandise" des quantificateurs de répétition est inversées, ce qui est rend non gourmand par défaut, mais s’ils sont suivis de ?, il seront gourmands.

Introduction

La syntaxe et la sémantique des expressions régulière supportées par PCRE sont décrites ci-dessous. Les expressions régulières sont aussi décrites dans la documentation Perl, et dans un grand nombre d’autres livres, avec de nombreux exemples. Jeffrey Friedl’s "Mastering Regular Expressions", édité chez O’Reilly (ISBN 1-56592-257-3), les décrits en profondeur. Cette description est organisée comme une documentation de référence.

Une expression régulière est un masque, appliqué à une chaîne sujet, de gauche à droite. La plupart des caractères se représentent eux-mêmes. Un exemple trivial : un masque qui serait "`Le rapide renard gris`", pourra correspondre à une partie de la chaîne sujet qui sera identique au masque, comme par exemple "`Le rapide renard gris court dans la forêt`",

Méta-caractères

La puissance des expressions régulières provient de leur capacité à autoriser des alternatives et des quantificateurs de répétition dans le masque. Ils sont encodés dans le masque par des méta-caractères, qui ne représentent pas ce qu’ils sont, mais sont interprétés d’une certaine manière.

Il y a deux sortes de méta-caractères : ceux qui sont reconnus n’importe où dans un masque, hormis entre crochets, et ceux qui sont reconnus entre crochets.

A l’extérieur des crochets, les méta caractères sont :

/ antislash

: Caractère d’échappement, avec de multiples usages

^ Accent circonflexe

: Le début de la chaîne sujet (ou de ligne, en mode multiligne)

$ Dollar

: La fin de la chaîne sujet (ou de ligne, en mode multiligne)

. Point

: Remplace n’importe quel caractère, hormis le caractère de nouvelle ligne (par défaut) ;

[Crochet ouvrant

Caractère de début de définition de classe

] Crochet fermant

Caractère de fin de définition de classe

| Barre verticale

Caractère de début d'alternative

(Parenthèse ouvrante

Caractère de début de sous-masque

) Parenthèse fermante

Caractère de fin de sous-masque

? Point d'interrogation

Etend le sens de (; quantificateur de 0 ou 1; quantificateur de minimisation

* Etoile

Quantificateur de 0 ou plus

+ Plus

Quantificateur de 1 ou plus

{ Accolade ouvrante

Caractère de début de quantificateur minimum/maximum

} Accolade fermante

Caractère de fin de quantificateur minimum/maximum

La partie du masque qui est entourée de crochet et appelé une classe de caractères. Dans les classes de caractères, les seuls méta caractères autorisés sont :

\ Antislash

Caractère d'échappement, avec de multiples usages

^ Accent circonflexe

Négation de la classe, mais uniquement si placé tout au début de la classe

- Moins

Indique un intervalle de caractères

] Crochet fermant

Termine la classe de caractères

La section suivante décrit l'utilisation de chaque méta-caractères.

Antislash

Le caractère antislash a de nombreuses utilisations.

En premier lieu, s'il est suivi d'un caractère non alpha-numérique, il ne prendra pas la signification spéciale qui y est rattachée. Cette utilisation de l'antislash comme caractère d'échappement s'applique à l'intérieur et à l'extérieur des classes de caractères. Par exemple, pour recherche le caractère étoile "*", il faut écrire dans le masque : "*". Cela s'applique dans tous les cas, que le caractère qui suive soit un méta-caractère ou non. C'est un moyen sÛr pour s'assurer qu'un caractère sera recherché pour sa valeur littérale, plutôt que pour sa valeur spéciale. En particulier, pour rechercher les antislash, il faut écrire : "\\".

Si un masque est utilisé avec l'option PCRE_EXTENDED, les espaces blancs du masque, mais qui ne sont pas dans une classe de caractères, et les caractères entre dièses "#", ainsi que les nouvelles lignes sont ignorées. L'antislash peut être utilisé pour échapper et ainsi rechercher un espace ou un dièse.

La deuxième utilité de l'antislash est de pouvoir coder des caractères invisibles dans les masques. Il n'y a pas de restriction sur la place de ces caractères invisibles, hormis pour le caractère nul qui doit terminer le masque.

Lors de la préparation du masque, il est souvent plus pratique d'utiliser les séquences d'échappement suivantes, plutôt que le caractère binaire qu'elle représente :

\a

alarme, c'est-à-dire le caractère BEL (hex 07)

\cx

"control-x", avec x qui peut être n'importe quel caractère.

\e

escape (hex 1B)

\f

formfeed (hex 0C)

\n

nouvelle ligne (hex 0A)

\r

retour chariot (hex 0D)

\t

tabulation (hex 09)

\xhh

caractère en hexadécimal, de code hh

\ddd

caractère en octal, de code ddd, ou référence arrière

Dans la séquence "`\cx`" si "`x`" est en minuscule, il est converti en majuscule. Puis, le bit 6 (hex 40) est inversé. Ainsi "`\cz`" devient `1A`, mais "`\c{`" devient hex 3B, tandis que "`\c;`" devient hex 7B.

Après "`\x`", deux caractères hexadécimaeux sont lus (les lettres peuvent être en majuscule ou minuscule).

Après "`\0`", deux caractères octal sont lus. Dans chacun des cas, le méta-caractère tente de lire autant de caractère que possible. Ainsi la séquence "`\0\x\07`", sera comprise comme deux caractères nuls, suivi d'un caractère alarme (BEL). Assurez-vous que vous fournissez suffisamment de chiffres après le méta-caractère.

La gestion de la séquence "`\y`", avec y <> 0 est plutôt compliquée. En dehors des caractères de classes, PCRE va lire y et tous les caractères qui suivent comme des chiffres décimaux. Si y est plus petit que 10, ou bien s'il y a déjà eu au moins autant de parenthèses ouvrantes auparavant, la séquence est prise pour une référence arrière. Le détail sera vu ultérieurement, après la section sur les sous-masques.

A l'intérieur d'un caractère de classe, ou si y est plus grand que 10, et qu'il n'y a pas eu assez de parenthèses ouvrantes auparavant, PCRE lis jusqu'à 3 chiffres octals à la suite de l'antislash, et génère un octet unique, à partir des 8 bits de poids faible de la séquence. Tous les chiffres qui suivent ne sont pas interprétés, et se representent eux-mêmes. Par exemple:

\040

une autre manière d'écrire un espace

\40

identique, dans la mesure où il n'y a pas 40 parenthèses ouvrantes auparavant

\7

est toujours une référence arrière

\11

peut être une référence de retour, ou une tabulation

\011

toujours une tabulation

\0113

est une tabulation suivi du caractère "3"

\113

est le caractère 113 (étant donné qu'il ne peut y avoir plus de 99 références arrières)

\377

est un octet dont tous les bits sont à 1

\0113

peut être soit une référence arrière, soit le caractère `NULL`, suivi des caractères "8" et "1"

Les valeurs octales supérieures ou égales à 100 ne doivent pas être introduites par un 0, car seuls les trois premiers octets seront lus.

Toutes les séquences qui définissent une valeur d'un seul octet peuvent être utilisé dans les classes de caractères, et à l'extérieur. De plus, dans une classe de caractères, la séquence "`\b`" est interprétée comme un caractère effacer (backspace, hex 08). A l'extérieur d'une classe de caractères, il peut avoir d'autres significations (voir ci-dessous).

On peut encore se servir de l'antislash pour préciser des types génériques de valeurs :

\d

tout caractère décimal

\D

tout caractère qui n'est pas un caractère décimal

\S

tout caractère blanc

\s

tout caractère qui n'est pas un caractère blanc

\W

tout caractère de "mot"

\w

tout caractère qui n'est pas un caractère de "mot"

Chaque paire précédente définit une partition de la table des caractères : les deux ensembles sont disjoints. Un caractère satisfera soit un méta-caractère, soit l'autre.

Un caractère de "mot" sera une lettre, un chiffre ou le caractère souligné, c'est-à-dire un caractère qui pourra être une partie d'un mot Perl. La définition des lettres et chiffres est définie par les tables de caractères de PCRE, et peut varier suivant la table locale de caractère (voir "Tables de caractères locales ", ci-dessus. Par exemple, dans la configuration français ("fr"), certains caractères ont des codes supérieurs à 128, pour les caractères accentués, et ils seront compris par le méta caractère `\w`.

Ces séquences de caractères peuvent apparaître à l'intérieur ou à l'extérieur des classes de caractères. Elles remplacent à chaque fois un caractère du type correspondant. Si cette séquence est placée en fin de masque, et qu'il n'y a plus de caractère à comparer dans la chaîne sujet, la recherche échoue.

La quatrième utilisation de l'antislash intervient lors d'assertions simples. Une assertion impose une condition à un certain point, sans remplacer de caractère. L'utilisation de sous-masques pour réaliser des assertions plus complexes est décrites plus-bas. Les assertions avec antislash sont les suivantes :

\b

limite de mot

\B

pas limite de mot

\A

début de la chaîne sujet (indépendant du mode multi-lignes)

\Z

fin de la chaîne sujet ou nouvelle ligne à la fin de la chaîne sujet (indépendant du mode multi-lignes)

\z

fin de la chaîne sujet (indépendant du mode multi-lignes)

Ces assertions ne peuvent pas apparaître dans une classe de caractères (mais "\b" a une autre signification à l'intérieur d'une classe de caractères).

Une limite de mot est un emplacement dans la chaîne sujet ou un caractère et son suivant ne sont pas en même temps des caractères de mot, ou le contraire (on peut le voir comme `\w\W` ou `\W\w`), ou encore le premier ou le dernier caractère est un caractère mot.

Les assertions `\A`, `\Z`, et `\z` diffèrent des méta caractères `^` et `$` dans la mesure où ils ne sont pas dépendants des options, notamment PCRE_NOTBOL ou PCRE_NOTEOL. La différence entre `\Z` et `\z` tient au fait que `\Z` recherche les positions avant les nouvelles lignes et à la fin de la chaîne sujet, tandis que `\z` ne recherche que la fin de la chaîne.

Accent circonflexe et Dollar

En dehors d'une classe de caractères, avec les options par défaut, `^` est une assertion qui n'est vraie que si elle est placée tout au début de la chaîne. A l'intérieur d'une classe de caractères, `^` a un tout autre sens (voir ci-dessous).

`^` n'a pas besoin d'être le premier caractère du masque, si plusieurs alternatives sont proposées, mais il doit être placé en premier dans chaque alternative. Si toutes les alternatives commencent par `^`, alors le masque est dit ancré (il y a une autre construction qui porte cette appellation).

`$` est une assertion qui n'est vraie que si elle est placée tout en fin de chaîne ou juste avant un caractère de nouvelle ligne qui serait le dernier caractère de la chaîne. A l'intérieur d'une classe de caractères, `$` a un tout autre sens (voir ci-dessous).

`$` n'a pas besoin d'être le dernier caractère du masque, si plusieurs alternatives sont proposées, mais il doit être placé en dernier dans chaque alternative. Si toutes les alternatives finissent par `$`, alors le masque est dit ancré (il y a une autre construction qui porte cette appellation). `$` n'a pas de valeur particulière dans une classe de caractères.

La signification de `$` peut changer, de manière à l'amener à ce qu'il ne puisse se trouver qu'en toute fin de la chaîne sujet. Cela se fait en ajoutant l'option PCRE_DOLLAR_ENDONLY au moment de la compilation, ou de l'exécution. Cette option est inopérante sur `\Z`.

La signification de `^` peut changer, de manière à l'amener à ce qu'il puisse se trouver immédiatement avant et immédiatement après un caractère de nouvelle ligne "`\n`". Cela se fait en ajoutant l'option PCRE_MULTILINE au moment de la compilation ou de l'exécution. Par exemple, le masque `/^abc$/` accepte la chaîne "`def\nabc`" uniquement en mode multi-lignes. Par conséquent, toutes les parties du masques qui commencent par "`^`" ne sont pas ancrées, en mode multi-lignes. L'option PCRE_DOLLAR_ENDONLY est ignorée si l'option PCRE_MULTILINE est choisie.

Notez que les méta caractères `\A`, `\Z`, et `\z` peuvent servir à répérer le début et la fin du sujet, et toutes les parties du masque qui commenceront par `\A` seront toujours ancrées, avec l'option PCRE_MULTILINE ou non.

Point

En dehors d'une classe de caractères, un point remplace n'importe quel caractère, même invisible et à l'exception du caractère de nouvelle ligne. Avec l'option PCRE_DOTALL le point remplace n'importe quel caractère, même le caractère de nouvelle ligne. La gestion des points et complètement indépendante de `^` et `$`. Le seul point commun est que les deux ont un comportement particulier vis à vis des caractère de nouvelle ligne.

Le point n'a pas de comportement particulier dans une classe de caractères.

Crochets

Un crochet ouvrant [introduit une classe de caractères, et le crochet fermant]la conclut. Le crochet fermant n'a pas de signification en lui-même. Si le crochet fermant est nécessaire à l'intérieur d'une classe de caractères, il faut qu'il soit le premier caractère (après un ^ éventuel) ou échappé avec un antislash.

Une classe de caractères remplace un seul caractère dans la chaîne sujet, à moins que le premier caractère de la classe soit un accent circonflexe ^, qui représente une négation : le caractère ne doit pas se trouver dans la classe. Si ^ est nécessaire dans la classe, il suffit qu'il ne soit pas le premier caractère, ou bien qu'il soit échappé avec un antislash.

Par exemple, le caractère [aeiou] remplace n'importe quelle voyelle minuscule, tandis que [^aeiou] remplace n'importe quelle caractère qui n'est pas une voyelle minuscule. ^ est une notation pratique pour spécifier des caractères qui sont dans une classe, en ne citant que ceux qui n'y sont pas. Le comportement est inchangé.

Avec l'option d'insensibilité à la casse, toutes les lettres d'une classe de caractères représentent en même temps la majuscule et la minuscule. Par exemple, [aeiou] représentera "A" ou "a", et [^aeiou] n'acceptera pas ni "A", tandis que sans l'option, elle l'accepterait.

Le caractère de nouvelle ligne n'est pas traité de manière spéciale dans les classes de caractères, quelque soit l'option PCRE_DOTALL ou PCRE_MULTILINE. Une classe telle que [^a] acceptera toujours une nouvelle ligne.

Le signe moins (-) est utilisé pour spécifier un intervalle de caractères, dans une classe. Par exemple, [d-m] remplace toutes les lettres entre d et m inclus. Si le caractère moins est requis dans une classe, il faut l'échapper avec un antislash, ou le faire apparaître à une position ou il ne pourra pas être interprété comme une indication d'intervalle, c'est-à-dire au début ou à la fin de la classe.

Il n'est pas possible d'avoir le caractère crochet fermant "]" comme fin d'intervalle. Un masque tel que [W-]46] est compris comme la classe de caractères contenant deux caractères ("W" et "-") suivi de la chaîne littérale "46]", ce qui fait qu'il va accepter "W46]" ou "-46]". Cependant, si "]" est échappé avec un antislash, le masque [W-\]46] est interprété comme une classe d'un seul caractère, contenant un intervalle de caractères.

La valeur octale ou hexadécimale de "]" peut aussi être utilisée pour déterminer les limites de l'intervalle. Les intervalles travaillent sur des séquences ASCII. Ils peuvent aussi être précisées avec des valeurs numériques, par exemple "[\000-\037]". Si cet intervalle inclus des lettres utilisées avec une option d'insensibilité de casse, les majuscules ou minuscules correspondantes seront aussi incluses. Par exemple, "[C-c]" est équivalent é "[][\^_‘wxyzabc]", avec l'option d'insensibilité de casse. Si la table locale de caractères est "fr", "[\xc8-\xcb]" correspond aux caractères accentués.

Les types de caractères \d, \D, \S, \s, \w, \W peuvent aussi intervenir dans les classes de caractères. Par exemple, "[][\^_‘wxyzabc][\dABCDEF]" acceptera n'importe quel caractère hexadécimal. Un accent circonflexe peut aussi être utilisé pour spécifier adroitement des ensembles de caractères plus restrictifs : par exemple [^\W_] accepte toutes les lettres et les chiffres, mais pas les soulignés. Tous les caractères non alpha- numériques autres que \, -, ^ (placés en début de chaîne) et] n'ont pas de signification particulière, mais ils ne perdront rien à être échappés.

Barre verticale

La barre verticale | sert à séparer des alternatives. Par exemple, dans le masque "/dupont|martin/" recherche soit "dupont", soit "martin". Le nombre d'alternatives n'est pas limité, et il est même possible d'utiliser la chaîne vide. Lors de la recherche, toutes les alternatives sont essayées, de gauche à droite, et la première qui est acceptée est utilisée.

Si les alternatives sont dans un sous-masque, elle ne réussiront que si le masque principal réussi aussi.

Options internes

Les options PCRE_CASELESS, PCRE_MULTILINE, PCRE_DOTALL et PCRE_EXTENDED peuvent être changée à l'intérieur du masque lui-même, avec des séquences mises entre "(?" et ")". Les options sont :

i

PCRE_CASELESS

m

PCRE_MULTILINE

s

PCRE_DOTALL

x

PCRE_EXTENDED

Par exemple, `(?im)` rend le masque insensible à la casse, et multi-lignes. Il est possible d'annuler ces options en les faisant précéder par un signe `-` : par exemple `(?im-sx)`, ajoutera les options PCRE_CASELESS et PCRE_MULTILINE mais annulera les options PCRE_DOTALL et PCRE_EXTENDED. Si une option apparaît avant et après le signe moins, l'option sera annulée.

Le domaine d'application de ces options dépend de la position de la séquence d'option. Pour toutes les séquences d'options qui sont hors des sous-masques (définis plus loin), l'effet est le même que si l'option avait été fixée dès le début de la recherche. Les exemples suivants se comportent tous de la même façons : `(?i)abc`, `a(?i)bc`, `ab(?i)c`, `abc(?i)`, et sont parfaitement équivalents au masque `abc` avec l'option PCRE_CASELESS. En d'autres termes, activer des séquences d'options dans le corps principal du masque revient à appliquer l'option à tout le masque, sauf ordre contraire dans les sous-masques. S'il y a plusieurs séquences d'options qui portent sur la même option, la dernière s'appliquera.

Si une option intervient dans un sous-masque, le comportement est différent. C'est un changement de comportement apparu en Perl 5.005. Une option à l'intérieur d'un sous-masque n'affecte que cette partie du masque, ce qui fait que `(a(?i)b)c` acceptera `abc` et `aBc` mais aucune autre chaîne (en supposant que PCRE_CASELESS n'est pas utilisé). Cela signifie que les options permettent d'avoir différente configuration de recherche pour différentes parties du masque.

Une séquence d'options dans une alternative affecte toute l'alternative. Par exemple : `(a(?i)b|c)` accepte "`ab`", "`aB`", "`c`", et "`C`", même si, comme dans le cas de "`C`", la première alternative qui porte l'option n'est pas prise en compte. Sinon, cela risque d'introduire des comportements très étranges : les options spécifiques à PCRE telles que PCRE_UNGREEDY et PCRE_EXTRA peuvent être modifiées de la même manière, en utilisant respectivement les caractères U et X. L'option `(?X)` est particulière, car elle doit toujours intervenir avant toutes les autres options, même au niveau du masque entier. Il vaut mieux l'activer au début du masque.

Sous-masques

Les sous-masques sont délimités par des parenthèses, et peuvent être imbriquées. Ajouter des sous-masques a deux utilités :

1. Délimiter des alternatives. Par exemple, le masque `char(don|mant|)` acceptera les mots "`char`", "`charmant`", ou "`charmant`". Sans les parenthèses, il n'accepterait que "`chardon`", "`mant`" ou la chaîne vide "".

2. Le sous-masque est considéré comme capturant : lorsqu'une chaîne sujet est acceptée par le masque complet, les sous-masques sont transmis à l'appelant grâce à un vecteur de sous-masques. Les parenthèses ouvrantes sont comptées de gauche à droite, (commençant à 1). Par exemple, soit la chaîne sujet "`le roi soleil`" qui est utilisée avec le masque suivant : `Le ((roi|prince) (soleil|charmant))` les sous-masques capturé sont "`roi soleil`", "`roi`", et "`soleil`", numérotés respectivement 1, 2, et 3.

L'ubiquité des parenthèses n'est pas toujours simple d'emploi. Il y a des moments où regrouper des sous-masques est nécessaire, sans pour autant capturer la valeur trouvée. Si une parenthèse ouvrante est suivie de "`?:`", le sous-masque ne capture pas la chaîne assortie, et ne sera pas compté lors de la numérotation des captures. Par exemple, avec la chaîne "`le prince charmant`", utilisé avec le masque `Le (( ?roi|prince) (soleil|charmant))` les chaînes capturées seront "`prince charmant`" et "`charmant`", numérotés respectivement 1 et 2.

Le nombre maximal de chaîne capturées est de 99, et le nombre total de sous-masque (capturant ou non) ne doit pas dépasser 200.

`(?i:samedi|dimanche)` et `(?:(?i) samedi | dimanche)` : De plus, comme les séquences d'options sont valables sur toute une alternative, les masques ci-dessus accepteront aussi bien "DIMANCHE" que "Dimanche".

Répétitions

Les répétitions sont spécifiées avec des quantificateurs, qui peuvent être placés à la suite des caractères suivants :

a

Un caractère unique, même s'il s'agit d'un méta caractère

[abc]

Une classe de caractères

\2

Une référence de retour (Voir section suivante)

(a|b|c)

Un sous-masque avec parenthèses (à moins que ce ne soit une assertion, voir plus loin)

Les quantificateurs généraux précisent un nombre minimum et maximum de répétitions possibles, donnés par deux nombres entre accolades, et séparés par une virgule. Ces nombres doivent être plus petits que 65536, et le premier nombre doit être égal ou inférieur au second. Par exemple `z{2,4}` accepte "`zz`", "`zzz`", ou "`zzzz`". L'accolade fermante n'a pas de signification par elle-même.

Si le second nombre est omis, mais que la virgule est là, cela signifie qu'il n'y a pas de limite supérieure. Si le second nombre et la virgule sont omis, le quantificateur correspond au nombre exact de répétition attendues. Par exemple : accepte n'importe quelle succession d'au moins 3 voyelles minuscules, tandis que `\d{d}` n'accepte que 8 chiffres exactement.

Une accolade ouvrante qui apparaît à une position où un quantificateur n'est pas accepté, ou si la syntaxe des quantificateurs n'est pas respectée, sera considérée littérale. Par exemple, "`{,6}`" n'est pas un quantificateur, mais une chaîne de 4 caractères.

Le quantificateur {0} est autorisé, mais l'expression est alors ignorée.

*

équivalent à {0,}

+

équivalent à {1,}

?

équivalent à {0,1}

Il est possible de constituer des boucles infinies en créant un sous-masque sans caractères, mais pourvu d'un quantificateur sans limite supérieure. Par exemple "`(a?)*`.

Les versions plus anciennes de Perl et PCRE généraient alors une erreur au moment de la compilation. Cependant, étant donné qu'il existe des situations où ces constructions peuvent être utiles, ces masques sont désormais autorisés. Cependant, si la répétion du sous-masque ne trouve aucun caractère, la boucle est interrompue.

Par défaut, les quantificateurs sont dits "gourmands", c'est à dire, qu'ils cherchent d'abord à trouve le nombre maximal de répétitions qui autorise le succès de la recherche. L'exemple classique posé par cette gourmandise est la recherche de commentaires d'un programme en C. Les commentaires apparaissent entre les séquences `/*....*/` et à l'intérieur de ces délimiteurs, les `*` et `/` sont autorisés. Appliquer le masque `/\*.*\*/` à la chaîne `/* first commet */ not comment /* second comment */` ne peut réussir, car le masque travaille sur toute la chaîne, à cause de la gourmandise du caractère `.*`.

Cependant, un quantificateur suivi d'un point d'interrogation cesse d'être gourmand, et au contraire, ne recherche que le nombre minimum de répétition. Dans ces conditions, le masque `/\*.*?\*/` trouvera bien les commentaires du code C. La signification des autres quantificateurs n'est pas changée.

Attention à ne pas confondre l'utilisation du point d'interrogation ici avec son utilisation comme quantificateur lui-même. A cause cette ambiguité, il peut apparaître des situations où il faut le doubler : `\d??\d`. Ce masque va tenter de lire un seul chiffre, mais le cas échéant, il acceptera 2 chiffres pour permettre à la recherche d'aboutir. Si l'option PCRE_UNGREEDY est activée, (une option qui n'est pas disponible avec Perl) alors les quantificateurs sont non gourmand par défaut, mais peuvent être rendu gourmand au cas par cas, en ajoutant un point d'interrogation après. En d'autres termes, cette option inverse le comportement par défaut.

Lorsqu'un sous-masque est quantifié avec un nombre minimum de répétitions, qui soit plus grand que 1, ou avec un maximum de répétitions, le masque compilé aura besoin de plus de place de stockage, proportionnellement au minimum et au maximum.

Si un masque commence par `.*` ou `.{0,}` et que l'option PCRE_DOTALL (équivalent en Perl à /s) est activée, c'est-à-dire en autorisant le remplacement des nouvelles lignes par un méta-caractère, alors le masque est implicitement ancré, car tout ce qui suit va être mangé par la première séquence, et se comportera comme si le masque se terminait par le méta caractère `\A`. Dans le cas où on sait d'avance qu'il n'y aura pas de caractère de nouvelle ligne, activer l'option PCRE_DOTALL et commencer le masque par `.*` permet d'optmiser le masque.

Alternativement, on peut utiliser ^ pour ancrer explicitement le masque. Lorsqu'un sous-masque capturant est répété, la valeur capturée est la dernière. Par exemple, après que "(inter[net]{3}\s*)+" ai été appliqué à "internet interne", la valeur de la chaîne capturée est "interne".

Cependant, s'il y a des sous-masques imbriqués, la valeur capturée correspondante peut l'avoir été lors des précédentes itérations. Par exemple : /(a|(b))+/ accepte "aba" et la deuxième valeur capturée est "b".

Références arrières

En dehors des classes de caractères, un antislash suivi d'un nombre plus grand que 0 (et possiblement plusieurs chiffres) est une référence arrière (c'est à dire vers la gauche) dans le masque, en supposant qu'il y ait suffisamment de sous-masques capturants précédants.

Cependant, si le nombre décimal suivant l'antislash est plus petit que 10, il sera toujours considéré comme une référence arrière, et cela génèrera une erreur si le nombre de capture n'est pas suffisant. En d'autres termes, il faut qu'il existe suffisamment de parenthèses ouvrantes à gauche de la référence, surtout si la référence est inférieure à 10.

Reportez-vous à la section "antislash" pour avoir de plus amples détails à propos du nombre de chiffres qui suivent l'antislash.

La référence arrière remplace ce qui a été capturé par un sous-masque dans le masque courant, plutôt que remplace le sous-masque lui-même. Ainsi (calme|rapide) et \1ment trouvera "calme et calmement" et "rapide et rapidement", mais pas "calme et rapidement". Si la recherche tient compte de la casse, alors la casse de la chaîne capturée sera importante. Par exemple, ((?i)rah)\s+\1 trouve "rah rah" et "RAH RAH", mais pas "RAH rah", même si le sous-masque capturant initial ne tenait pas compte de la casse.

Il peut y avoir plusieurs références arrières dans le même sous-masque. Si un sous-masque n'a pas été utilisé dans une recherche, alors les références arrières échoueront. Par exemple "(a|(bc))\2" ne réussira jamais si la chaîne sujet commence par "a" plutôt que par "bc".

Etant donné qu'il peyt y avoir jusqu'à 99 références arrières, tous les chiffres après l'antislash sont considérés comment faisant potentiellement partie de la référence arrière. Si le masque recherche un chiffre après la référence, alors il faut impérativement utiliser des délimiteurs pour terminer la référence arrière.

> Si l'option PCRE_EXTENDED est activée, on peut utiliser un espace. Sinon, un commentaire vide fait l'affaire. Une référence arrière qui intervient à l'intérieur de parenthèses auquel elle fait référence échouera dès que le sous-masque sera utilisé. Par exemple, (a\1) échouera toujours. Cependant, ces références peuvent être utiles dans les sous-masques répétitifs. Par exemple, le masque "(a|b\1)+" pourra convenir pour "a", "aba", "ababaa", etc....

A chaque itération du sous-masque, la référence arrière utilise le résultat du dernier sous-masque. Pour que cela fonctionne, il faut que la première itération n'ai pas besoin d'utiliser la référence arrière. Cela arrive avec les alternatives, comme dans l'exemple ci-dessus, ou avec un quantificateur de minimum 0.

Assertions

Une assertion est un test sur les caractères suivants ou précédent celui qui est en cours d'étude. Ce test ne consomme par de caractère (ie, on ne déplace pas le pointeur de caractères). Les assertions simples sont codées avec \b, \B, \A, \Z, \z, ^ et $, et sont décrites précédemment.

Il existe cependant un type d'assertions plus complexes, codées sous la forme de sous-masques. Il en existe deux types : celles qui travaillent au-delà de la position courante (\w+(?=;)), et celles qui travaillent en deça ((?!)\w+).

Une assertion se comporte comme un sous-masque, hormis le fait qu'elle ne déplace pas le pointeur de position. Les assertions avant commencent par (?= pour les assertions positives, et par (?!, pour les assertions négatives. Par exemple : \w+(?=;) s'assure qu'un mot est suivi d'un point-virgule, mais n'inclus pas le point virgule dans la capture. D'autre part, (?!foo)bar en est proche, mais ne trouve pas une occurrence de "bar" qui soit précédée par quelque chose d'autre que "foofoo"; il trouve toutes les occurrences de "bar", quelque soit ce qui le précéde, car l'assertion (?!foo) est toujours vraie quand les trois caractères suivants sont "bar". Une assertion arrière est ici nécessaire.

Les assertions arrières commencent par (?<= pour les assertions positives, et (?<! pour les assertions négatives. Par exemple : (?<!foo)bar trouve les occurrences de "bar" qui ne sont pas précédées par "foo".

Le contenu d'une référence arrière est limité de telle façon que les chaînes qu'il utilise soient toujours de la même taille. Cependant, lorsqu'il y a plusieurs alternatives, elles n'ont pas besoin d'être de la même taille. Par exemple, (?<=bullock|donkey) est autorisé, tandis que (?<!dogs?|cats?) provoque une erreur de compilation. Les alternatives qui ont des longueurs différentes ne sont autorisées qu'au niveau supérieur des assertions arrières. C'est une

amélioration du fonctionnement de Perl 5.005, qui impose aux alternatives d'avoir toutes la même taille. Une assertion telle que `(?<=ab(c|de))` n'est pas autorisée, car l'assertion de bas niveau (la deuxième, ici) a deux alternatives de longueurs différentes. Pour la rendre acceptable, il faut écrire `(?<=abc|abde)`

L'implémentation des assertions arrières déplace temporairement le pointeur de position vers l'arrière, et cherche à vérifier l'assertion. Si le nombre de caractères est différent, la position ne sera pas correcte, et l'assertion échouera. La combinaison d'assertions arrières avec des sous-masques peut être particulièrement pratique à fin des chaînes. Un exemple est donné à la fin de cette section.

Plusieurs assertions peuvent intervenir successivement. Par exemple, le masque `(?<=\d{3})(?<!999)foo` recherche les chaînes "`foo`" précédées par trois chiffres qui ne sont pas "999". Notez que chaque assertion est appliquées indépendemment, au même point de la chaîne à traiter. Tout d'abord, il est vérifié que les trois premiers caractères ont tous des chiffres, puis on s'assure que ces trois caractères ne sont pas "`999`". Le masque précédant n'accepte pas "`foo`" précédé de 6 caractères, les trois premiers étant des chiffres et les trois suivants étant différents de "`999`". Par exemple, ce masque n'acceptera pas la chaîne "`123abcfoo`". Pour ce faire, il faut utiliser le masque suivant : `(?<=\d{3}...)(?<!999)foo`. Dans ce masque, la première assertion vérifie les six premiers caractères, s'assure que les trois premiers sont des entiers, et la deuxième assertion s'assure que les trois derniers caractères ne sont pas "`999`".

De plus, les assertions peuvent être imbriquées : `(?<=(?<!foo)bar)baz` recherche les occurrences de "`baz`" qui sont précédées par "`bar`", qui, à son tour, n'est pas précédé par "`foo`". Au contraire, `(?<=\d{3}(?!999)...)foo` est un autre masque, qui recherche les caractères "`foo`", précédés par trois chiffres, suivis trois autres caractères qui ne forment pas "`999`". Les assertions ne sont pas capturantes, et ne peuvent pas être répétées. Si une assertion contient des sous-masques capturants en son sein, ils seront compris dans le nombre de sous-masques capturants du masque entier. La capture est réalisée pour les assertions positives, mais cela n'a pas de sens pour les assertions négatives.

200 assertions au maximum sont autorisées.

Sous-masques uniques

Avec les quantificateurs de répétitions, l'échec d'une recherche conduit normalement à une autre recherche, avec un nombre différent de répétitions, pour voir si le masque ne s'applique pas dans d'autres conditions. Parfois, il est pratique d'éviter ce comportement, soit pour changer la nature de la recherche, soit pour la faire abandonner plus tôt, si on pense qu'il n'est pas besoin d'aller plus loin.

Considérons par exemple, le masque `\d+foo` appliqué à la ligne `123456bar`. Après avoir tenté d'utiliser les 6 chiffres suivi de "`foo`" qui font échouer, l'action habituelle sera de réessayer avec 5 chiffres, puis avec 4, et ainsi de suite jusqu'à l'échec final.

Un sous-masque évalué une seule fois permettrait d'indiquer que lorsqu'une partie du masque est trouvée, elle n'a pas besoin d'être réévaluée à chaque tentative. Ceci conduirait à ce que la recherche échoue immédiatement après le premier test. Ces assertions ont leur propre notation, commençant avec `(?>` comme ceci : `(?>\d+)bar`.

Ce type de parenthèses verrouille le sous-masque qu'il contient un fois qu'il a été trouvé, et empêche un échec ultérieur d'y repasser, mais autorise à revenir plus loin en arrière. Une autre description est que les sous-masques de ce type recherche les chaînes de caractères, et les ancre le sous-masque à l'intérieur de la chaîne.

Les sous-masques uniques ne sont pas capturants. Des cas simples comme ceux présentés ci-dessus peuvent être pris comme des situations maximisantes, qui réservent le maximum de caractères. En effet, alors que `\d+` et `\d+?` ajustent le nombre de chiffres trouvés de manière à laisser la possibilité au masque de réussir, `(?>\d+)` ne peut retenir que la séquence entière de chiffres. Cette construction peut contenir un nombre arbitraire de sous-masques complexes, et ils peuvent être imbriqués.

Les sous-masques uniques ne peuvent être utilisés qu'avec les assertions arrières, pour effectuer une recherche efficace en fin de chaîne. Considérons un masque simple tel que "`abcd$`" appliqué à une très longue chaîne qui ne lui correspond pas. A cause du système de recherche de gauche à droite, PCRE va commencer par rechercher un "`a`" dans la chaîne sujet, puis vérifier si ce qui suit convient au reste du masque. Si le masque est spécifié sous la forme `^.*abcd$` alors, la séquence `.*` remplace en premier lieu la chaîne entière, et échoue, repart en arrière, et remplace tous les caractères sauf le dernier, échoue, retourne en arrière, prend un caractère de moins, etc... et ainsi de suite. Encore une fois, la recherche du "`a`" passe en revue toute la chaîne de gauche à droite, ce qui n'est pas très efficace. Par contre, si le masque était écrit `^(?>.*)(?<=abcd)` alors il n'y aurait pas de retour en arrière, pour satisfaire la séquence `.*`, car elle ne peut que remplacer toute la chaîne. L'assertion arrière consécutive va alors faire un test sur les 4 derniers caractères. Si elle échoue, la recherche est immédiatement interrompue.

Pour les chaînes très longues, cette approche fait la différence en terme de performances et de temps de recherche. Lorsqu'un masque contient une répétition illimitée dans un sous-masque, qui contient lui-même un nombre illimité de

répétiteur, l'utilisation des sous-masques à utilisation unique sont la seule façon d'éviter l'échec de la recherche à après un temps de calcul trop long.

Le masque `(\D+|<\d+>)*[!?]` recherche un nombre illimité de sous-chaînes, qui contiennent soit des non-chiffres, soit des chiffres inclus dans <>, suivi soit par `!` ou par `?`. Lorsqu'il trouve une solution, ce masque va très vite. Mais, lorsqu'il est appliqué à une chaîne telle que : `aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa`, il lui faut beaucoup de temps pour annoncer un échec. Cela est dÛ au fait que la chaine peut être divisée en deux sous-chaînes d'un grand nombre de façons, et qu'elles ont toutes été essayées. (Cet exemple utilisait `[!?]` plutôt qu'un caractère simple, car PCRE et PHP utilise une optimisation qui leur permettent de détecter rapidement l'échec lorsqu'un caractère unique est trouvé. Il se souvient du dernier caractère qui est attendu, et s'aperçoit rapidement qu'il n'y a pas ce caractère).

Si le masque utilisé est `((?>\D+)|&;lt;\d+>)*[!?]` les séquences de chiffres ne peuvent pas être trouvées, et l'échec intervient rapidement.

Les sous-masques conditionnels

Il est possible de lier un sous-masque à une condition, ou de choisir entre deux sous-masques alternatifs, en fonction du résultat d'une assertion, ou suivant les résultats de recherche précédents.

Les deux formes possibles de sous-masques conditionnels sont `(?(condition)masque positif)` et `(?(condition) masque positif | masque négatif)`.

Si les conditions sont satisfaites, le masque positif est utilisé, sinon, le masque négatif est utilisé, si présent. S'il y a plus de deux alternatives, une erreur est générée à la compilation.

Il y a deux types de conditions : si le texte entre les parenthèses est une séquence de chiffres, alors la condition est satisfaite si le sous-masque correspondant à ce numéro a réussi. Considérons le masque suivant, qui contient des espaces non significatifs pour le rendre plus compréhensible (on supposera l'option PCRE_EXTENDED activée) et qui est divisé en trois parties pour simplifier les explications : `( \( )? [^()]+ (?(1) \) )`.

La première partie recherche une parenthèse ouvrante optionnelle, et si elle existe, elle est capturée. La deuxième partie recherche un séquence de caractères qui ne contiennent pas de parenthèses. La troisième partie est conditionnée à la première, et s'assure que s'il y avait une parenthèse ouvrante, il en existe une fermante. Si une parenthèse ouvrante a été trouvée, elle a été capturée, et donc la première capture existe, et la condition est exécutée. Sinon, elle est ignorée.

Ce masque recherche donc une séquence de lettres, éventuellement placées entre parenthèse. Si la condition n'est pas une séquence de chiffres, il faut que ce soit une assertion. Ce peut être une assertion positive ou négative, arrière ou avant. Considérons le masque suivant (même conditions que le précédent) et avec deux alternatives en seconde ligne : `(?(?=[^a-z]*[a-z])\d{2}[a-z]{3}-\d{2} | \d{2}-\d{2}-\d{2} )`. La condition est une assertion avant positive, qui recherche une séquence optionnelle de caractères non-lettre. En d'autres termes, elle teste la presence d'au moins une lettre dans la chaîne sujet. Si une lettre est trouvée, la recherche se poursuit avec la première alternative, et sinon, avec la seconde. Ce masque recherche des chaînes de la forme `dd-aaa-dd` ou `dd-dd-dd`, avec "`aaa`" qui sont des lettres, et `dd` qui sont des chiffres.

Commentaires

La séquence `(?#` marque le début d'un commentaire, qui se termine à la prochaine parenthèse fermante. Les parenthèses imbriquées ne sont pas autorisées. Les caractères entre ces délimiteurs ne jouent alors aucun rôle dans le masque.

Si l'option PCRE_EXTENDED est activée, les caractères dièses # non échappés en dehors d'une classe de caractères introduisent un commentaire qui continuera jusqu'à la prochaine ligne dans le masque.

Masques récursifs

Considérons le cas où il faut recherche dans une chaîne, avec un niveau d'imbrications infini de parenthèses. Sans l'aide de la récursivité, le mieux que nous puissions obtenir est de créer un masque avec un niveau fixé de profondeur d'imbrication. Il n'est pas possible de traiter des masques à niveau d'imbrications variable. PCRE fournit un nouvel outil expérimental qui permet d'utiliser la récursivité dans les masques (entre autre). L'option `(?R)` est fournie pour servir la cause de la récursivité. Le masque suivant résoud le problème des parenthèses (l'option PCRE_EXTENDED est utilisée pour ignorer les espaces) : `\( ( (?>[^()]+) | (?R) )* \)`

Tout d'abord, le masque recherche une parenthèse ouvrante. Puis, il recherche n'importe quel nombre de sous-chaînes qui sont soit des séquences de caractères non-parenthèses, ou bien une recherche récursive avec le même masque (i.e. une chaîne correctement incluse entre parenthèses). Finalement, il recherche une parenthèse fermante.

Cet exemple particulier contient un nombre illimité de répétitions imbriquées, ce qui fait que l'utilisation de sous-chaînes à utilisation unique pour rechercher les séquence de caractères non-parenthèses est important, lorsqu'il s'applique à une chaîne qui n'est pas valide. Par exemple, si on l'applique à "`(aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa()`" la réponse arrive rapidement. Sinon, si les sous-chaînes à utilisation unique ne sont pas utilisées, la recherche peut prendre un très long temps, car il existe de très nombreuses combinaisons de `+` et `*` à tester avant de conclure à l'échec.

Les valeurs utilisées pour capturer les sous-masques sont celles utilisées par les niveaux les plus hauts de récursivités, auquel la valeur est fixée. Si le masque précédent est utilisé avec `(ab(cd)ef)` la valeur de la parenthèse capturante est "`ef`", qui est la dernière valeur lue au niveau supérieur. Si de nouvelles parenthèses sont ajoutées, par exemple : `\( ( ( (?>[^()]+) | (?R) )* ) \)` alors la chaîne capturée est "`ab(cd)ef`", c'est-à-dire le contenu de la parenthèses capturant de plus haut niveau. S'il y a plus de 15 parenthèses capturantes dans une chaîne, PCRE doit utiliser plus de mémoire pour stocker ces données. S'il ne peut obtenir cette mémoire supplémentaire, il ne fait que sauver les 15 premières, car il n'y a pas moyen de générer une erreur de mémoire lors d'une récursion.

Performances

Certaines séquences de recherches sont plus efficaces que d'autres. Ainsi, il est plus efficace d'utiliser une classe de caractères telle que `[aeiou]` plutôt qu'une alternative `(a|e|i|o|u)`.

En général, le masque le plus simple, qui permette la recherche désirée est le plus efficace. Le livre de Jeffrey Friedl's contient de nombreuses études à propos de l'optimisation des expressions régulières.

Lorsqu'un masque commence par.* et que l'option PCRE_DOTALL est activée, le masque est implicitement ancré par PCRE, étant donné qu'il ne peut que rechercher au début de la chaîne. Cependant, si option PCRE_DOTALL n'est pas activée, PCRE ne peut faire aucune optimisation car le méta-caractères point "`.`" ne remplace pas une nouvelle ligne, et si la chaîne sujet contient des nouvelles lignes, le masque peut trouver une solution qui serait située juste après une de ces nouvelles lignes, et non pas seulement au début de la chaîne sujet. Par exemple, le masque, `(.*)second` acceptera la chaîne "`premier \net second`" (avec "`\n`" qui remplace la nouvelle ligne), et la première chaîne capturée sera "`et`".

Afin d'effectuer la recherche, PCRE va essayer d'appliquer le masque à partir de chaque début de ligne. Si vous utilisez un tel masque avec des chaînes qui ne contiennent pas de caractères de nouvelles lignes, les meilleures performances seront atteintes avec l'option PCRE_DOTALL, ou en ancrant le masque avec `^.*`. Cela évite à PCRE de scanner toute la chaîne pour rechercher un caractère de nouvelle ligne et recommencer la recherche.

Attention aux masques qui contiennent des quantificateurs infinis imbriqués. Ils peuvent demander un temps de calcul très long, lorsqu'appliqués à une chaîne qui ne correspond pas à ce masque. Par exemple, `(a+)*` peut accepter "`aaaa`" de 33 manières différentes, et ce nombre croit rapidement avec la taille de la chaîne (le quantificateur `*` peut prendre les valeurs de 0, 1, 2, 3, ou 4, et pour chaque cas non nul, le quantificateur `+` peut prendre différentes valeurs).

Lorsque le reste de la chaîne est tel que l'on s'achemine vers un échec, PCRE doit en principe vérifier toutes les possibilités, et cela prend un temps extrêmement long. Un optmiseur repère les cas les plus simples, tel que `(a+)*b` où un caractère simple suit les quantificateurs. Avant de partir dans les procédures standard de recherche, PCRE s'assure qu'il y a au moins un "`b`" dans la chaîne, et si ce n'est pas le cas, l'échec est annoncé immédiatement. Sinon, il n'y a pas d'optimisation dans la recherche. Vous pouvez voir la différence de comportement avec le masque suivant : `(a+)*\d`. Le premier retourne un échec quasi-immédiatement, s'il est appliqué à une ligne de "`a`", alors que le second masque prend un temps significatif pour une chaîne de plus de 20 caractères.

LXXIV. Expressions régulières

Les expressions régulières sont utilisées pour effectuer des manipulations complexes de chaînes de caractères. Les fonctions sont :

- **ereg()**
- **ereg_replace()**
- **eregi()**
- **eregi_replace()**
- **split()**
- **spliti()**

Ces fonctions requièrent toutes une expression régulière comme premier argument. PHP utilise les expressions régulières avancées de POSIX (POSIX 1003.2). Pour avoir tous les détails sur ces expressions, reportez vous aux pages de manuel inclues dans le répertoire de la distribution PHP.

Exemple 1. Expressions régulières

```
<?php
ereg("abc",$string);
/* Retourne TRUE si "abc"
   est trouvé quelque part dans la chaîne $string. */
ereg("^abc",$string);
/* Retourne TRUE si  "abc"
   est trouvé au début de la chaîne $string. */
ereg("abc$",$string);
/* Retourne TRUE si  "abc"
   est trouvé à la fin de la chaîne  $string. */
eregi("(ozilla.[23]|MSIE.3)",$HTTP_USER_AGENT);
/* Retourne TRUE si  le client
   est Netscape 2, 3 ou MSIE 3. */
ereg("([[:alnum:]]+) ([[:alnum:]]+) ([[:alnum:]]+)",
     $string,$regs);
/* Introduit trois mots séparés par des espaces
   dans les chaînes $regs[1], $regs[2] et $regs[3]. */
$string = ereg_replace("^","<BR>",$string);
/* Insère une balise <BR> au début de la chaîne $string. */
$string = ereg_replace("$","<BR>",$string);
/* Insère une balise <BR> à la fin de la chaîne $string. */
$string = ereg_replace("\n","",$string);
/* Supprime toutes les nouvelles lignes de $string. */
?>
```

ereg (PHP 3, PHP 4 >= 4.0b1)

Expression régulière standard.

```
int ereg (string pattern, string string [, array regs])
```

Recherche dans la chaîne *string* les séquences de caractères qui correspondent au masque *pattern*.

Si au moins une séquence est trouvée (éventuellement dans les parenthèses capturantes de *pattern*), et que la fonction est appelée avec un troisième argument *regs*, les résultats seront enregistrés dans *regs*. $regs[1] contiendra la première parenthèse capturante (celle qui commence le plus tôt), $regs[2] contiendra la deuxième parenthèse capturante (celle qui commence après la première), et ainsi de suite. $regs[0] contient une copie de la chaîne.

Si **ereg()** trouve ses solutions pour les parenthèses capturantes, *$regs* contiendra exactement 10 éléments, même si il y avait plus ou moins de 10 parenthèses capturantes qui étaient valides. Cela n'a aucun effet sur les capacités de la fonction **ereg()** à trouver d'autres sous chaînes. Si aucune valeur n'est trouvée, $regs ne sera pas modifié par **ereg()**.

La recherche est sensible à la casse.

ereg() retourne `TRUE` si une occurrence a été trouvée dans la chaîne et `FALSE` dans le cas contraire, ou si une erreur est survenue.

L'exemple suivant prend une date au format ISO (YYYY-MM-DD) et l'affiche sous la forme DD.MM.YYYY :

Exemple 1. Exemple ereg()

```
<?php
if ( ereg( "([0-9]{4})-([0-9]{1,2})-([0-9]{1,2})", $date, $regs ) ) {
    echo "$regs[3].$regs[2].$regs[1]";
} else {
    echo "Format de date invalide : $date";
}
?>
```

Voir aussi **eregi()**, **ereg_replace()** et **eregi_replace()**.

ereg_replace (PHP 3, PHP 4 >= 4.0b1)

Remplacement par expression régulière.

```
string ereg_replace (string pattern, string replacement, string string)
```

ereg_replace() effectue une recherche par expression régulière dans la chaîne *string* en recherchant les occurrences de *pattern*, puis les remplace par la chaîne *replacement*.

La chaîne modifiée est retournée. (Ce qui signifie que la chaîne originale sera retournée si aucune occurrence n'est trouvée).

Si *pattern* contient des parenthèses capturantes, *replacement* pourra contenir des séquences de la forme `\\`*digit*, qui seront remplacées par le texte capturé par la n-ième parenthèse capturante. `\\0` correspond à la chaîne originale complète. De 0 à 9 parenthèses capturantes peuvent être utilisées. Les parenthèses peuvent être imbriquées, et leur numéro d'ordre est défini par leur parenthèse ouvrante.

Si aucune occurrence n'est trouvée, la chaîne *string* sera retournée intacte.

Par exemple, le code suivant affiche "Ceci etait un test" trois fois :

Exemple 1. Exemple avec ereg_replace()

```
<?php
$string = "Ceci est un test";
echo ereg_replace( " est", " etait", $string );
echo ereg_replace( "( )est ", "\\1etait", $string );
echo ereg_replace( "(( )est)", "\\2etait", $string );
?>
```

Notez bien que si vous utilisez une valeur de type entier dans le paramètre de remplacement *`replacement`*, vous risquez de ne pas obtenir le résultat escompté. Tout cela parce que **ereg_replace()** va interpréter le nombre comme la valeur ordinale d'un caractère, et l'utiliser. Par exemple :

Exemple 2. Exemple avec ereg_replace()

```
<?php
/* Cet exemple ne fonctionne pas comme voulu. */
$num = 4;
$string = "Cette chaîne a quatre mots.";
$string = ereg_replace('quatre', $num, $string);
echo $string;   /* Affichage : 'Cette chaîne a  mots.' */
/* Ceci est bon. */
$num = '4';
$string = "Cette chaîne a quatre mots.";
$string = ereg_replace('quatre', $num, $string);
echo $string;   /* Affichage : 'Cette chaîne a 4 mots.' */
?>
```

Voir aussi **ereg()**, **eregi()** et **eregi_replace()**.

eregi (PHP 3, PHP 4 >= 4.0b1)

Recherche par expression régulière insensible à la casse.

```
int eregi (string pattern, string string [, array regs])
```

eregi() est identique à **ereg()**, hormis le fait qu'elle ignore la casse des caractère lors de la recherche sur les caractère alphabétiques.

Voir aussi **ereg()**, **ereg_replace()** et **eregi_replace()**.

eregi_replace (PHP 3, PHP 4 >= 4.0b1)

Remplacement par expression régulière insensible à la casse.

```
string eregi_replace (string pattern, string replacement, string string)
```

eregi_replace() est identique à **ereg_replace()**, hormis le fait qu'elle ne tient pas compte de la casse des caractères alphabétiques.

Voir aussi **ereg()**, **eregi()** et **ereg_replace()**.

split (PHP 3, PHP 4 >= 4.0b1)

Scinde une chaîne en un tableau, grâce à une expression régulière.

```
array split (string pattern, string string [, int limit])
```

split() retourne une tableau de chaînes : chacune d'entre elle est une sous-chaîne de *`string`* délimitée par les occurrences trouvées de l'expression régulière *`pattern`*. Si une erreur survient, retourne `FALSE`.

Pour lire les 5 premiers champs d'une ligne du fichier `/etc/passwd`:

Exemple 1. Exemple avec split()

```
<?php
$passwd_list = split( ":", $passwd_line, 5 );
?>
```

Pour analyser une date qui est délimitée par des /, des points ou des tirets :

Exemple 2. Exemple avec split()

```
<?php
$date = "04/30/1973";
// Les délimiteurs peuvent être des /, des points ou des tirets
list( $month, $day, $year ) = split( '[/.-]', $date );
echo "Mois: $month; Jour: $day; Annee: $year<br>\n";
?>
```

Notez que *`pattern`*est insensible à la casse

Notez bien que si vous n'avez pas besoin de la puissance des expressions régulières, il est plus rapide d'utiliser **explode()**, qui n'utilise pas le moteur d'expressions régulières.

Notez aussi que *`pattern`* est une expression régulière. Si vous voulez utiliser n'importe quel caractère spécial des expressions régulières, vous devez les échapper. Si vous pensez que **split()** (ou toute autre expression régulière) se comporte bizarrement, lisez d'abord le fichier `regex.7`, inclus dans le dossier `regex/` de la distribution PHP . Il est au format manpage, et vous pourrez le lire avec une commande telle que **man /usr/local/src/regex/regex.7**.

Voir aussi : **explode()** et **implode()**.

spliti (PHP 4 >= 4.0.1)

Scinde une chaîne en un tableau, grâce à une expression régulière.

```
array spliti (string pattern, string string [, int limit])
```

spliti() est identique à **split()**, hormis le fait qu'elle ignore la casse.

Voir aussi **split()**, **explode()** et **implode()**.

sql_regcase (PHP 3, PHP 4 >= 4.0b1)

Prépare une expression régulière pour effectuer une recherche insensible à la casse.

```
string sql_regcase (string string)
```

sql_regcase() retourne une expression régulière valide qui acceptera la chaîne *`string`*, et toutes les variantes majuscule/minuscule possibles de cette chaîne. Cette expression sera construite à partir de la chaîne *`string`* en remplacant tous les caractères par des expressions entre crochets (des classes de caractères), contenant la lettre majuscule et minuscule. Si le caractère n'est pas une lettre, les crochets contiendront deux fois le caractère original.

Exemple 1. Exemple avec sql_regcase()

```
<?php
echo sql_regcase( "Foo bar" );
?>
```

affichera `[Ff][Oo][Oo] [Bb][Aa][Rr]`.

Cette expression sert à effectuer des recherches insensibles à la casse avec d'autres logiciels, qui n'acceptent les recherches insensibles à la casse.

LXXV. Satellite CORBA client extension

Avertissement

Ce module est *EXPERIMENTAL*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

Le module Satellite sert à accéder aux objets CORBA. Vous devez ajouter l'option idl_directory= entry dans `php.ini`: c'est le chemin jusqu'aux fichiers IDL.

OrbitObject (unknown)

Accède aux objets CORBA

```
new orbitobject (string ior)
```

orbitobject() crée un objet qui accèdera aux objets CORBA. Le paramètre *ior* doit être une chaîne contenant l'IOR (Interoperable Object Reference (référence interopérable d'objet)) qui identifie l'objet distant.

Exemple 1. Fichier IDL : MonInterface

```
interface MonInterface {
    void SetInfo (string info);
    string GetInfo();
    attribute int value;
}
```

Exemple 2. Code PHP pour accéder à MonInterface

```
<?php
$obj = new OrbitObject ($ior);
$obj->SetInfo ("Un Super Objet");
echo $obj->GetInfo();
$obj->value = 42;
echo $obj->value;
?>
```

OrbitEnum (unknown)

Utilise les énumérations CORBA

```
new orbitenum (string id)
```

orbitenum() représente l'énumeration identifiée par *id*. *id* peut être soit le nom d'une énumération (e.g "MonEnumeration"), ou bien l'identifiant du repository complet (e.g. "IDL:MyEnum:1.0").

Exemple 1. Fichier d'exemple IDL : MonEnumeration

```
enum MonEnumeration {
    a,b,c,d,e
};
```

Exemple 2. Code PHP pour accéder à MonEnumeration

```
<?php
$enum = new OrbitEnum ("MonEnumeration");
echo $enum->a; /* écrit 0 */
echo $enum->c; /* écrit 2 */
echo $enum->e; /* écrit 4 */
?>
```

OrbitStruct (unknown)

Utilise une structure CORBA

```
new orbitstruct (string id)
```

orbitstruct() représente une structure identifiée par le paramètre *id*. *id* peut être soit le nom d'une structure (e.g "MaStructure"), ou bien l'identifiant du repository complet (e.g. "IDL:MaStructure:1.0").

Exemple 1. Fichier d'exemple IDL : MaStructure

```
struct MaStructure {
    short shortvalue;
    string stringvalue;
};
interface UneInterface {
  void SetValues (MaStructure values);
  MaStructure GetValues();
}
```

Exemple 2. Code PHP pour accéder à MaStructure

```
<?php
$obj = new OrbitObject ($ior);
$initial_values = new OrbitStruct ("IDL:MaStructure:1.0");
$initial_values->shortvalue = 42;
$initial_values->stringvalue = "HGTTG";
$obj->SetValues ($initial_values);
$values = $obj->GetValues();
echo $values->shortvalue;
echo $values->stringvalue;
?>
```

satellite_caught_exception (PHP 4 >= 4.0.3)

Indique si une exception a été émise

```
bool satellite_caught_exception  ()
```

satellite_caught_exception() retourne TRUE si une exception a été émise lors du dernier appel à une fonction Orbit.

Exemple 1. Fichier IDL exemple : PlusDeFromage

```
<?php
/* ++?????++ Erreur PlusDeFromage. Recommence tout au début. */
exception PlusDeFromageErreur {
    int parameter;
}
interface UneAutreInterface {
    void AskWhy() raises (PlusDeFromage);
```

```
}
?>
```

Exemple 2. Code PHP pour gérer les exceptions CORBA

```
<?php
$obj = new OrbitObject ($ior);
$obj->AskWhy();
if (satellite_caught_exception()) {
    if ("IDL:PlusDeFromage:1.0" == satellite_exception_id()) {
        $exception = satellite_exception_value();
        echo $exception->parameter;
    }
}
?>
```

satellite_exception_id (PHP 4 >= 4.0.3)

Lit l’identifiant de repository de la dernière exception

```
string satellite_exception_id ()
```

satellite_exception_id() retourne une chaîne d’identification d’un repository. (E.g. "IDL:MyException:1.0".) Pour un exemple, voyez **satellite_caught_exception()**.

satellite_exception_value (PHP 4 >= 4.0.3)

Lit la structure de la dernière exception

```
OrbitStruct satellite_exception_value  ()
```

satellite_exception_value() retourne une structure d’exception. Pour un exemple, voyez **satellite_caught_exception()**.

LXXVI. Sémaphores et gestion de la mémoire partagée

Ce module fourni un système de sémaphore. Ce système utilise les sémaphores System V. Les sémaphores peuvent être utilisés pour fournir un accès exclusif à certaines ressources de la machine, ou pour limiter le nombre de processus qui utilisent en même temps une ressource.

Ce module fournit aussi un système de mémoire partagée, qui utilise la mémoire partagée System V. Cette mémoire partagée permet d'accéder à des variables globales. Les différents démons httpd et mêmes d'autres programmes (tels que Perl, C, ...) permettent un tel échange de données global. N'oubliez pas que la mémoire partagée n'est pas protégées contre l'accès simultané. Il vous faudra utiliser les sémaphores pour assurer la synchronisation.

Tableau 1. Limites de la mémoire partagée sous Unix OS

SHMMAX	Taille maximale de mémoire partagée, par défaut, 131072 octets.
SHMMIN	Taille minimale de mémoire partagée, par défaut, 1 octet.
SHMMNI	Nombre maximal de segment de mémoire partagé, par défaut 100.
SHMSEG	Taille maximale de mémoire partagée par processus, par défaut 6.

sem_get (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Retourne un identifiant de sémaphore.

```
resource sem_get (int key [, int max_acquire [, int perm]])
```

sem_get() retourne un identifiant positif de sémaphore en cas de succès, et FALSE en cas d'erreur.

sem_get() retourne un identifiant qui pourra être utilisé pour accéder à un sémaphore System V. Le sémaphore est créé, si nécessaire, en utilisant les bits de permission (par défaut, 0666). Le nombre de processus qui peuvent réserver simultanément le sémaphore est précisé dans max_acquire (par défaut à 1). Actuellement, cette valeur n'est affectée que si le processus est le seul processus actuellement attaché au sémaphore.

Un deuxième appel à **sem_get()** avec la même clé retournera un identifiant différent, mais les deux identifiants permettront d'accéder au même sémaphore.

Voir aussi **sem_acquire()** et **sem_release()**.

Note : sem_get() n'est pas disponibles sous Windows.

sem_acquire (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Réserve un sémaphore.

```
int sem_acquire (resource sem_identifier)
```

sem_acquire() retourne TRUE en cas de succès, et FALSE sinon.

sem_acquire() se bloque (si nécessaire) jusqu'à ce que le sémaphore puisse être réservé. Un processus qui tente de réserver un sémaphore qu'il a déjà reservé restera en attente indéfinie, si cette acquisition excède le nombre max_acquire de réservation simultanée.

A la fin d'un script, tous les sémaphores réservés mais non explicitement libérés seront libérés automatiquement, et une alerte sera générée.

Voir aussi **sem_get()** et **sem_release()**.

sem_release (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Libère un sémaphore.

```
int sem_release (int sem_identifier)
```

Retourne TRUE en cas de succès, FALSE en cas d'erreur.

sem_release() libère le sémaphore s'il a été réservé par le processus courant. Sinon, génère une erreur.

Après libération du sémaphore, **sem_acquire()** peut être appelé pour le réserver à nouveau.

Voir aussi : **sem_get()** et **sem_acquire()**.

Note : Cette fonction n'est pas disponibles sous Windows.

shm_attach (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Crée ou ouvre un segment de mémoire partagée.

```
resource shm_attach (int key [, int memsize [, int perm]])
```

shm_attach() retourne un identifiant qui permettra d'accéder au System V de mémoire partagée. Au premier appel, la mémoire sera créée, avec la taille mem_size (par défaut: sysvshm.init_mem dans `php3.ini`, sinon 10000 octets) et avec les permissions perm(par défaut : 666).

Aux appels suivants avec la même clé *`key`*, **shm_attach()** retournera un nouvel identifiant, mais cet identifiant accédera toujours à la même portion de mémoire partagée. Dans ce cas, *`memsize`* et *`perm`*seront ignorés.

Note : Cette fonction n'est pas disponibles sous Windows.

shm_detach (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Libère un segment de mémoire partagée.

```
resource shm_detach (int shm_identifier)
```

shm_detach() libère le segment de mémoire partagée identifié par *`shm_identifier`* et créé par **sem_get()**. N'oubliez pas que cette mémoire partagée existe toujours sous Unix, et que les données sont toujours accessibles.

shm_remove (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Supprime un segment de mémoire partagée sous Unix.

```
int shm_remove (resource shm_identifier)
```

shm_remove() supprime un segment de mémoire partagée sous Unix. Toutes les données seront supprimées.

Note : shm_remove() n'est pas disponibles sous Windows.

shm_put_var (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Insère ou modifie une variable de la mémoire partagée.

```
int shm_put_var (resource shm_identifier, int variable_key, mixed variable)
```

shm_put_var() insère ou modifie la variable *`variable`* avec la clé *`variable_key`*. Tous les types de variables (double, int, string, array, objects...) sont supportés.

Note : shm_put_var() n'est pas disponibles sous Windows.

shm_get_var (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Lit une variable dans la mémoire partagée.

```
mixed shm_get_var (resource shm_identifier, int variable_key)
```

shm_get_var() retourne la variable repérée par *variable_key*. La variable est toujours présente en mémoire partagée.

Note : shm_get_var() n'est pas disponibles sous Windows.

shm_remove_var (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Efface une variable de la mémoire partagée.

```
int shm_remove_var (int shm_identifier, int variable_key)
```

shm_remove_var() efface la variable *variable_key* de la mémoire partagée et libère la mémoire.

Note : shm_remove_var() n'est pas disponibles sous Windows.

LXXVII. SESAM

SESAM/SQL-Server est une base de données mainframe, développée par Fujitsu Siemens Computers, Allemagne. Elle fonctionne sur les serveur mainframe, sous BS2000/OSD.

Sur de nombreuses installation BS2000 en production, SESAM/SQL-Server a prouvé ...

- La facilité de connectivité Java, Web et client/serveur
- La disponibilité de plus de 99.99%,
- La capacité de gérer des dizaines et mêmes des centaines de milliers d'utilisateurs.

Désormais, il existe une interface PHP pour SESAM, qui donne l'accès à cette base aux scripts PHP.

Notes de configuration : Il n'y a pas de support pour l'interface SESAM si PHP est un CGI : elle ne fonctionne que comme module Apache. En module Apache, l'interface SESAM peut être configurée avec les directives Apache.

Tableau 1. Directives de configuration SESAM

Directive	Signification
`php3_sesam_oml`	Nom de la librairie BS2000 PLAM contenant le module du pilote SESAM. Ceci est obligatoire pour utiliser les fonctions SESAM. Exemple: `php3_sesam_oml $.SYSLNK.SESAM-SQL.030`

`php3_sesam_configfile`	Nom du fichier de configuration de l'application SESAM. Ceci est obligatoire pour utiliser les fonctions SESAM. Exemple:
	`php3_sesam_configfile $SESAM.SESAM.CONF.AW`
	Ce fichier contient généralement une configuration comme celle ci (voir dans le manuel de référence SESAM):
	`CNF=B`
	`NAM=K`
	`NOTYPE`

`php3_sesam_messagecatalog`	Nom du fichier de messages SESAM. Dans la plus part des cas, cette directive n'est pas nécessaire. Uniquement si le fichier de messages SESAM n'es pas installé dans la table de messages BS2000. Il peut alors être choisi avec cette directive. Exemple: `php3_sesam_messagecatalog $.SYSMES.SESAM-SQL.030`

En plus de la configuration de l'interface PHP/SESAM, vous devez aussi configurer le serveur SESAM-Database sur votre mainframe, comme d'habitude. Cela signifie notamment qu'il faut :

- démarrer le gestionnaire de base SESAM (DBH)
- connecter les bases avec le gestionnaire de bases SESAM

Pour connecter un script PHP au serveur de bases SESAM, les paramètres `CNF` et `NAM` de la configuration SESAM selectionnée doivent correspondre à l'id du gestionnaire de base démarré.

Dans le cas des bases de données distribuées, vous devez démarrer un agent SESAM/SQL-DCN, avec la table de distribution incluant le nom de l'hôte et de la base de données.

La communication entre PHP (fonctionnant sur le sous-système POSIX) et le gestionnaire de base (fonctionnant hors

du sous-système POSIX) est réalisée par un pilote spécial appelé SQLSCI et le module de connexion SESAM, qui utilise la mémoire partagée. A cause de la mémoire partagée, et parce que PHP est une partie statique du serveur web, les accès à la base de données sont extrêmement rapide, car il ne requièrent pas de connexion distante via ODBC, JDBC ou UTM.

Seul un chargeur de stub (stub loader, SESMOD) est compilé dans PHP. Les modules de connexion SESAM proviennent de la librairie OML PLAM. Dans la configuration, vous devez indiquer à PHP le nom de la librairie PALM, et le fichier de lien à utiliser pour la configuration de SESAM (En SESAM V3.0, SQLSCI est disponible dans la librairie d'outils SESAM (SESAM Tool Library), qui fait partie de la distribution standard).

Les commandes SQL imposent que les guillemets simples soient doublés pour être interprété litéralement (contrairement à d'autres bases de données qui utilisent un guillemet simple, précédé d'un antislash), il est recommandé d'activer les directives PHP `php3_magic_quotes_gpc` et `php3_magic_quotes_sybase`.

Exécutions : A cause des limitations du modèle de processus BS2000, le pilote peut être chargé uniquement après que le serveur Apache ait généré le processus fils. Cela ralentit légèrement le traitement de la première requête, mais toutes les requêtes suivantes seront effecutée à pleine vitesse.

Lorsque vous définissez explicitement le catalogue de messages SESAM, ce catalogue ser chargé à chaque fois que le pilote est chargé (i.e., au moment de la requête initiale). Le système d'exploitation BS2000 affiche un message après avoir correctement chargé le catalogue de messages, qui sera envoyé au fichier d'erreurs Apache. BS2000 ne permet pas la suppression de ce message, qui va remplir progressivement ce fichier.

Assurez vous que la librairie SESAM OML PLAM et le fichier de configuration SESAM sont accessibles par l'utilisateur qui fait tourner le serveur web. Sinon, le serveur ne sera pas capable de charger le pilote, ou d'appeler les fonctions SESAM. L'accès à la base doit être donné à cet utilisateur. Sinon, les connexions SESAM échoueront.

Types de curseurs : Les curseurs de résultat sont alloués pour les requêtes SQL de selection, peuvent être soit "séquentiels", soit "à défilement" ("scrollable"). Les curseurs à défilement sont beaucoup plus gourmands en mémoire, et le mode par défaut est séquentiel.

Lorsque vous utilisez les curseurs à défilement, le curseur peut être positionné librement dans le résultat. Pour chaque requête à défilement, il existe des valeurs globales de types de défilement (initialisée à :`SESAM_SEEK_NEXT`) et la position peut être fixée une seule fois par **sesam_seek_row()** ou bien à chaque appel, avec la fonction **sesam_fetch_row()**. Lorsque vous lisez une ligne avec un curseur à défilement, le traitement suivant est effectué à partir des valeurs globales de type de défilement et de position :

Tableau 2. Scrolled Cursor Post-Processing

Type de défilement	Action
`SESAM_SEEK_NEXT`	aucun
`SESAM_SEEK_PRIOR`	aucun
`SESAM_SEEK_FIRST`	le type de défilement devient `SESAM_SEEK_NEXT`
`SESAM_SEEK_LAST`	le type de défilement devient `SESAM_SEEK_PRIOR`
`SESAM_SEEK_ABSOLUTE`	incrémente automatiquement la valeur interne de position
`SESAM_SEEK_RELATIVE`	aucune. conserve les valeurs globales par défaut de position, ce qui permet, par exemple de lire toutes les 10 lignes, en arrière.

Porting note : En PHP, il est naturel de commencer les index à zéro (plutôt que 1), et quelques adaptations ont été faite pour l'interface SESAM : à chaque fois qu'un tableau indexé commence à l'index 1 en SESAM natif, l'interface PHP utilisera l'index 0 comme point de départ. Par exemple, lorsque vous lisez des données avec **sesam_fetch_row()**, la première colonne sera à l'index 0, et les suivantes suivront juqu'au nombre de colonne (exclus) du résultat ($array["count"]). Lors du portage d'applications depuis d'autres langage évolués vers le PHP, soyez attentifs à ce changement. A chaque fois que c'est nécessaire, la description d'une fonction PHP SESAM indique que l'index du tableau commence à 0.

Sécurité : Lorsque vous autorisez l'accès à une base de données SESAM, le serveur web doit avoir le minimum de privilèges possible. Pour la plus part des bases de données, seul le droit de lecture doit être fourni. Suivant votre

utilisation, ajoutez d'autres droits d'accès au fur et à mesure de vos besoins. Ne donnez jamais le contrôle total de vos bases à un utilisateur du web! Limitez l'accès aux scripts PHP qui doivent administrer la base en utilisant un mot de passe et/ou une sécurisation SSL.

Migration d'une autre base SQL : Deux langage SQL ne sont jamais 100% compatibles. Lorsque vous portez une application SQL depuis une autre interface vers SESAM, certaines adaptation doivent être faîtes. Les différences suivantes sont les plus courantes :

- Types de données spécifiques

  Certains types de données spécifiques à une base doivent être remplacés par les types de données standard SQL. (i.e., `TEXT` doit être remplacé par `VARCHAR(taille max)`).

- Mots reservés comme identifiants SQL.

  En SESAM (comme dans le standard SQL), les mots reservés utilisés comme identifiants doivent être entourés de guillemets doubles (ou renommés).

- Taille d'affichage des données.

  Les types de données SESAM ont une taille de stockage, mais par de taille d'affichage. A la place de `int(4)` (c'est à dire : les entiers jusqu'à '9999'), SESAM requiert simplement `int`, pour une taille implicite de 31 bits. De même, les seuls types de date disponible dans SESAM sont : `DATE`, `TIME(3)` et `TIMESTAMP(3)`.

- Les types de données `unsigned` (non signé), `zerofill` (complété avec des zéros), ou `auto_increment`

  `Unsigned` et `zerofill` ne sont pas supportés. `Auto_increment` est automatique (utilisez `"INSERT ... VALUES(*, ...)"` au lieu de `"... VALUES(0,...)"` pour profiter des auto-increment implicites de SESAM.

- **int ... DEFAULT '0000'**

  Les variables numériques ne doivent pas être initialisées avec des constantes de type chaîne de caractères. Utilisez **DEFAULT 0** à la place. Pour initialiser une date, la chaîne doit être préfixée avec le type de date adapté, tel que :
  ```
  CREATE TABLE exmpl (xtime timestamp(3) DEFAULT TIMESTAMP '1970-01-01 00:00:00.000' NOT NULL);
  ```

- **$count = xxxx_num_rows();**

  Certaines bases de données essaient d'estimer le nombre de lignes d'un résultat, même grossièrement approximativement. SESAM ne connait pas le nombre de lignes avant de les avoir lues lui-même. Si vous avez vraiment besoin de les compter, utilisez la commande **SELECT COUNT(...) WHERE ...**, qui vous dira combien de lignes sont disponibles. Une deuxième requête devrait vous retourner tous ces résultats.

- **DROP TABLE lenom;**

  Avec SESAM, dans la commande **DROP TABLE**, le nom de la table doit être suivi du mot clé `RESTRICT` ou `CASCADE`. Avec `RESTRICT`, une erreur est retounrée si il y a des objets dépendant (par exemple, des vues), tandis qu'avec `CASCADE`, les objets dépendants seront supprimés en même temps que la table.

Notes sur l'utilisation de types SQL divers : SESAM ne supporte pas le type BLOB. Une future version de SESAM devra le faire.

L'interface PHP effectue automatiquement les conversions suivantes lors de la lecture de lignes de résultats SQL :

Tableau 3. Conversion de type SQL vers PHP

Type SQL	Type PHP
SMALLINT, INTEGER	"integer" (entier)

Type SQL	Type PHP
NUMERIC, DECIMAL, FLOAT, REAL, DOUBLE	"double" (nombre à virgule flottante)
DATE, TIME, TIMESTAMP	"string"(chaîne de caractères)
VARCHAR, CHARACTER	"string"(chaîne de caractères)

Lorsque vous lisez une ligne entière, le résultat est retourné sous la forme d'un tableau. Les champs vides ne sont pas remplis, et vous aurez à vérifier vous même l'existence des champs ((utilisez **isset()** ou **empty()** pour tester les champs vides). Cela donne plus de contrôle à l'utilisateur sur l'apparence des champs que si les champs vides étaient représenté par des chaînes vides).

Support des "champs multiples" de SESAM : La fonctionnalité spéciale des "champs multiples" de SESAM permet à une colonne de contenir un tableau de champs. Un tel "champs multiple" peut être créé comme ceci :

Exemple 1. Création d'une colonne de champs multiples

```
CREATE TABLE multi_field_test
(
     pkey  CHAR(20) PRIMARY KEY,
     multi(3)   CHAR(12)
)
```

et peut être remplie avec :

Exemple 2. Affectation d'une colonne de type "champs multiple"

```
INSERT INTO multi_field_test ( pkey, multi(2..3) )
  VALUES ( 'Second', <'first_val','second_val>)
```

Notez que (comme c'est le cas ci-dessus), les sous-champs vides initiaux sont ignorés, et que le tableau est alors compacté, ce qui fait que l'exemple ci-dessus conduit à un tableau multi(1..2) au lieu de multi(2..3).

Lors de la lecture d'une ligne, les "champs multiples" sont mis en colonne. Dans l'exemple ci-dessu, "pkey" prend l'index 0, et les trois colonnes "multi(1..3)" sont accessibles depuis les index 1 à 3.

Pour de plus amples détails sur SESAM, reportez vous à la documentation SESAM/SQL-Server (http://its.siemens.de/lobs/its/techinf/oltp/sesam/manuals/index_en.htm) en anglais ou SESAM/SQL-Server (http://its.siemens.de/lobs/its/techinf/oltp/sesam/manuals/index_gr.htm) en allemand, disponibles toutes deux en ligne, ou en manuels.

sesam_connect (PHP 3 CVS only)

Ouvre une connexion SESAM

```
boolean sesam_connect (string catalog, string schema, string user)
```

sesam_connect() retourne TRUE si une connexion à la base SESAM a été faite, ou FALSE en cas d'erreur.

sesam_connect() établit une connexion au serveur SESAM. La connexion est toujours "persistante", en ce sens que le pilote sera chargé par la première requête avec la librairie SESAM OML PLAM. Les appels suivants réutiliseront le pilote chargé, son catalogue *catalog*, son schéma *schema* et son utilisateur *user*.

Lors de la création d'une base de données, le nom *catalog* est spécifié dans les directives de configuration SESAM avec la commande **//ADD-SQL-DATABASE-CATALOG-LIST ENTRY-1 = *CATALOG(CATALOG-NAME = catalogname,...)**

schema référence le schéma de base voulu (voir dans le manuel SESAM).

user spéficie l'un des utilisateurs qui est autorisé à accéder à la comibinaison *catalog* et/ou *schema*. Notez que *user* est complétement indépendant de l'utilisateur système et des protections HTTP par mot de passe. Il n'apparait que dans la configuration SESAM.

Voir aussi **sesam_disconnect()**.

Exemple 1. Connexion à une base SESAM

```
<?php
if (! sesam_connect ("moncatalogue", "monschema", "toto")
     die("Impossible de se connecter à SESAM");
?>
```

sesam_disconnect (PHP 3 CVS only)

Déconnexion d'une base SESAM

```
boolean sesam_disconnect (void)
```

sesam_disconnect() retourne toujours TRUE.

sesam_disconnect() ferme le lien logique à la base de données SESAM (sans réellement déconnecter et démonter le pilote).

Notez que ceci n'est généralement pas nécessaire, car la connexion ouverte est automatiquement fermée à la fin du script. Les données qui ne seront pas validées seront alors annulées, grâce à un **sesam_rollback()** implicite.

sesam_disconnect() ne ferme pas les connexions persistantes : elle invalide simplement les catalogues *catalog*, schéma *schema* et utilisateur *user* courants, de manière à ce que les prochains appels à des fonctions SESAM échouent.

Voir aussi : **sesam_connect()**.

Exemple 1. Déconnexion d'une base SESAM

```
<?php
if (sesam_connect ("moncatalogue", "monschema", "toto")) {
... quelques requêtes et d'autres trucs ...
sesam_disconnect();
}
?>
```

sesam_settransaction (PHP 3 CVS only)

Modifie les paramètres de transaction SESAM

```
boolean sesam_settransaction (int isolation_level, int read_only)
```

sesam_settransaction() retourne TRUE si les valeurs sont valides et que a modification a été réussie. FALSE sinon.

sesam_settransaction() remplace les valeurs par défaut du niveau d'isolation ("isolation level") et de lecture seule ("read-only") fixée par le fichier de configuration SESAM), afin d'optimiser les requêtes ultérieures et garantir la cohérence de la base. Ces valeurs ne sont utilisées que pour la prochaine transaction.

sesam_settransaction() ne peut être appelée qu'avant le début de la transaction. Elle est inefficace si la transaction a déjà commencé.

Pour simplifier l'utilisation de cette fonction dans les scripts PHP, les constantes suivantes ont été définies en PHP (reportez vous au manuel SESAM pour avoir des détails sur leur signification) :

Tableau 1. Valeurs valides pour le paramètre *Isolation_Level*

Valeur	Constante	Signification
1	SESAM_TXISOL_READ_UNCOMMITTED	Lecture sans validation
2	SESAM_TXISOL_READ_COMMITTED	Lecutre avec validation
3	SESAM_TXISOL_REPEATABLE_READ	Lecture récurente
4	SESAM_TXISOL_SERIALIZABLE	Sérialisable

Tableau 2. Valeurs valides pour le paramètre *Read_Only*

Valeur	Constante	Signification
0	SESAM_TXREAD_READWRITE	Lecture/écriture
1	SESAM_TXREAD_READONLY	Lecture seule

Les valeurs modifiées par **sesam_settransaction()** remplaceront les valeurs par défaut spécifiée dans le fichier de configuration SESAM.

Exemple 1. Modifier les paramètres de configuration SESAM

```
<?php
sesam_settransaction(SESAM_TXISOL_REPEATABLE_READ,
                     SESAM_TXREAD_READONLY);
?>
```

sesam_commit (PHP 3 CVS only)

Valide la transaction SESAM en cours

```
boolean sesam_commit (void)
```

sesam_commit() retourne TRUE en cas de succès et FALSE sinon.

sesam_commit() valide toutes les modifications de tables en attente sur la base.

Notez qu’il n’y a pas de mode "auto-commit", comme dans d’autres bases de données, car cela peut conduire à une perte accidentelle de données. Les données non valides à la fin d’un script (ou au moment de l’appel de **sesam_disconnect()**) seront annulées par un appel implicite à **sesam_rollback()**.

Voir aussi : **sesam_rollback()**.

Exemple 1. Valider une transaction SESAM

```
<?php
if (sesam_connect ("moncatalogue", "monschema", "toto")) {
  if (!sesam_execimm("INSERT INTO mytable VALUES (*, ’Small Test’, <0, 8, 15>)"))
     die("insertion manquée");
  if (!sesam_commit())
     die("insertion réussie");
}
?>
```

sesam_rollback (PHP 3 CVS only)

Annule une transaction SESAM

```
boolean sesam_rollback (void)
```

sesam_rollback() retourne TRUE en cas de succès et FALSE en cas d’erreur.

sesam_rollback() annule toutes les modifications en cours sur la base. Les curseurs de résultat et les descripteurs de résultats seront affectés.

A la fin de chaque script, et dans chaque appel à **sesam_disconnect()**, un appel implicite à **sesam_rollback()** est fait, annulant toutes les transactions non validées dans la base.

Voir aussi : **sesam_commit()**.

Exemple 1. Annulation d’une transaction SESAM

```
<?php
if (sesam_connect ("moncatalogue", "monschema", "toto")) {
  if (sesam_execimm("INSERT INTO matable VALUES (*, ’Petit Test’, <0, 8, 15>)")
   && sesam_execimm("INSERT INTO autretable VALUES (*, ’Autre Test’, 1)"))
     sesam_commit();
  else
     sesam_rollback();
}
?>
```

sesam_execimm (PHP 3 CVS only)

Exécute immédiatement une requête SQL

```
string sesam_execimm (string query)
```

sesam_execimm() retourne un identifiant de résultat SESAM en cas de succès, et FALSE sinon.

sesam_execimm() exécute immédiatement la requête *query* (i.e., une requête de type UPDATE, INSERT ou DELETE qui ne retourne aucun résultat, et n’a aucune variables d’entrées ou de sorties). Les requêtes de types "SELECT" ne

peuvent pas être utilisées avec la fonctoin **sesam_execimm()**. **sesam_execimm()** mofidie la valeur *affected_rows*, pour lecture ultérieure avec **sesam_affected_rows()**.

Notez que **sesam_query()** peut gérer les requêtes immédiates et les requêtes de selection. Utilisez **sesam_execimm()** uniquement si vous connaissez le type de requête auparavant. Une tentative de requête de selection avec **sesam_execimm()** retournera `$err["sqlstate"] == "42SBW"`.

L'identifiant de résultat retourné ne peut pas être utilisé pour lire quoi que ce soit, mais il peut être passé à **sesam_affected_rows()**; il n'est retourné que pour symétrie avec la fonction **sesam_query()**.

```
<?php
$stmt = "INSERT INTO matable VALUES('un', 'deux')";
$result = sesam_execimm ($stmt);
$err = sesam_diagnostic();
print("sqlstate = ".$err["sqlstate"]."\n".
      "Nombre de lignes affectées = ".$err["rowcount"]." == ".
      sesam_affected_rows($result)."\n");
?>
```

Voir aussi : **sesam_query()** et **sesam_affected_rows()**.

sesam_query (PHP 3 CVS only)

Exécute une requête SESAM

```
string sesam_query (string query [, boolean scrollable])
```

sesam_query() retourne un identifiant de résultat SESAM en cas de succès, ou FALSE en cas d'erreur.

L'identifiant de résultat est utilisé par d'autres fonctions sesam pour lire les valeurs.

sesam_query() envoie une requête à la base active. Elle peut exécuter aussi bien une requête immédiate (DELETE, UPDATE ou INSERT), ou une requête de selection. Si une requête immédiate est exécutée, aucun curseur n'est alloué, et il ne sera pas possible d'utiliser les fonctions **sesam_fetch_row()** ou **sesam_fetch_result()**. Pour les requêtes de selection, un descripteur de résultat et un curseur (scrollable ou séquentiel, suivant le paramètre optionnel *scrollable* passé) sear alloué. Si *scrollable* est omis, le curseur sera séquentiel.

Lorsque vous utilisez les curseurs à défilement, le curseur peut être positionné librement dans le résultat. Pour chaque requête à défilement, il existe des valeurs globales de types de défilement (initialisée à :SESAM_SEEK_NEXT) et la position peut être fixée une seule fois par **sesam_seek_row()** ou bien à chaque appel, avec la fonction **sesam_fetch_row()**.

Pour les requêtes immédiates, le nombre de lignes affectées est sauvé, et est accessible par la fonction **sesam_affected_rows()**.

Voir aussi : **sesam_fetch_row()** et **sesam_fetch_result()**.

Exemple 1. Liste toutes les lignes de table "phone" sous forme de table HTML

```
<?php
if (!sesam_connect("phonedb", "demo", "toto"))
  die("cannot connect");
$result = sesam_query("select * from phone");
if (!$result) {
  $err = sesam_diagnostic();
  die($err["errmsg"]);
}
echo "<TABLE BORDER>\n";
// Ajoute l'entête de titre comme nom de colonne
if ($cols = sesam_field_array($result)) {
  echo " <TR><TH COLSPAN=".$cols["count"].>Result:</TH></TR>\n";
  echo " <TR>\n";
  for ($col = 0; $col < $cols["count"]; ++$col) {
    $colattr = $cols[$col];
```

```
    /* étend les entêtes de la table au dessus des champs multiples */
    if ($colattr["count"] > 1) {
      echo "  <TH COLSPAN=".$colattr["count"].">".$colattr["name"].
           "(1..".$colattr["count"].")</TH>\n";
      $col += $colattr["count"] - 1;
    }
    else
      echo "  <TH>" . $colattr["name"] . "</TH>\n";
  }
  echo " </TR>\n";
}
do {
  // Lit les résultats par bloc de 100
  $ok = sesam_fetch_result($result,100);
  for ($row=0; $row < $ok["rows"]; ++$row) {
    echo " <TR>\n";
    for ($col = 0; $col < $ok["cols"]; ++$col) {
      if (isset($ok[$col][$row]))
        echo "  <TD>" . $ok[$col][$row] . "</TD>\n";
      else
        echo "  <TD>-empty-</TD>\n";
    }
    echo " </TR>\n";
  }
} while ($ok["truncated"]); // tant qu'il a y encore des données
echo "</TABLE>\n";
// libère les ressources
sesam_free_result($result);
?>
```

sesam_num_fields (PHP 3 CVS only)

Retourne le nombre de colonne dans un résultat

```
int sesam_num_fields (string result_id)
```

Après avoir appelé **sesam_query()** avec une requête de selection, **sesam_num_fields()** indique le nombre de colonnes du résultat identifié par *result_id*. Retourne FALSE en cas d'erreur.

Pour les requêtes immédiates, la valeur zéro est retournée. Les champs multiples SESAM compte autant que leur taille respective, c'est à dire qu'un champs multiple de trois colonnes compte comme trois colonne.

Voir aussi : **sesam_query()** et **sesam_field_array()** pour savoir distinguer les champs multiples des colonnes standard.

sesam_field_name (PHP 3 CVS only)

Retourne le nom d'une colonne

```
int sesam_field_name (string result_id, int index)
```

sesam_field_name() retourne le nom du champs *index* dans le résultat identifié par *result_id*, ou FALSE en cas d'erreur.

Pour les requêtes immédiates, ou les colonnes dynamiques, une chaîne vide est retournée.

> **Note :** Les colonnes sont indexées à partir de 0, et non pas 1.

Voir aussi : **sesam_field_array()**. Cette fonction fournit une interface simple aux noms et types de colonnes, et permet la detection des champs multiples.

sesam_diagnostic (PHP 3 CVS only)

Retourne l'état de la dernière requête SESAM

```
array sesam_diagnostic (void)
```

sesam_diagnostic() retourne un tableau associatif avec l'état et les codes de la dernière requête SQL. Les éléments du tableau sont :

Tableau 1. Informations retournées par sesam_diagnostic()

Elément	Contenu
$array["sqlstate"]	code d'erreur à 5 chiffres (voir le manuel SESAM pour obtenir une description des valeurs possibles de SQLSTATE)
$array["rowcount"]	nombre de lignes affectées dans la dernière requête immédiate (update/insert/delete) : uniquement après une requête immédiate.
$array["errmsg"]	message d'erreur lisible : uniquement après une erreur
$array["errcol"]	numéro de colonne de la dernière erreur (indexée à partir de 0, -1 si indéfinies. uniquement après une erreur).
$array["errlin"]	numéro de ligne de la dernière erreur (indexée à partir de 0, -1 si indéfinies. uniquement après une erreur).

Dans l'exemple suivant, une erreur de syntaxe (E SEW42AE ILLEGAL CHARACTER) est affichée avec la requête SQL, et en désignant la position de l'erreur :

Exemple 1. Afficher une erreur SESAM

```
<?php
// Fonction qui affiche un message d'erreur formaté
// en affichant la position de l'erreur dans le message d'erreur
function PrintReturncode($exec_str)
{
   $err = Sesam_Diagnostic();
   $colspan=4; // 4 colonnes pour : sqlstate, errlin, errcol, rowcount
   if ($err["errlin"] == -1)
     --$colspan;
   if ($err["errcol"] == -1)
     --$colspan;
   if ($err["rowcount"] == 0)
     --$colspan;
   echo "<TABLE BORDER>\n";
   echo "<TR><TH COLSPAN=".$colspan."><FONT COLOR=red>ERROR:</FONT> ".
   htmlspecialchars($err["errmsg"])."</TH></TR>\n";
   if ($err["errcol"] >= 0) {
     echo "<TR><TD COLSPAN=".$colspan."><PRE>\n";
     $errstmt = $exec_str."\n";
     for ($lin=0; $errstmt != ""; ++$lin) {
       if ($lin != $err["errlin"]) { // $lin is less or greater than errlin
         if (! ($i = strchr($errstmt, "\n")))
           $i = "";
         $line = substr($errstmt, 0, strlen($errstmt)-strlen($i)+1);
         $errstmt = substr($i, 1);
         if ($line != "\n")
```

```
            print htmlspecialchars($line);
          }
          else {
            if (! ($i = strchr($errstmt, "\n")))
              $i = "";
            $line = substr($errstmt, 0, strlen($errstmt)-strlen($i)+1);
            $errstmt = substr($i, 1);
            for ($col=0; $col < $err["errcol"]; ++$col)
              echo (substr($line, $col, 1) == "\t") ? "\t" : ".";
            echo "<FONT COLOR=RED><BLINK>\\</BLINK></FONT>\n";
            print "<FONT COLOR=\"#880000\">".htmlspecialchars($line)."</FONT>";
            for ($col=0; $col < $err["errcol"]; ++$col)
              echo (substr($line, $col, 1) == "\t") ? "\t" : ".";
            echo "<FONT COLOR=RED><BLINK>/</BLINK></FONT>\n";
          }
        }
        echo "</PRE></TD></TR>\n";
    }
    echo "<TR>\n";
    echo " <TD>sqlstate=" . $err["sqlstate"] . "</TD>\n";
    if ($err["errlin"] != -1)
      echo " <TD>errlin=" . $err["errlin"] . "</TD>\n";
    if ($err["errcol"] != -1)
      echo " <TD>errcol=" . $err["errcol"] . "</TD>\n";
    if ($err["rowcount"] != 0)
      echo " <TD>rowcount=" . $err["rowcount"] . "</TD>\n";
    echo "</TR>\n";
    echo "</TABLE>\n";
}
if (!sesam_connect("moncatalogue", "phoneno", "toto"))
  die("cannot connect");
$stmt = "SELECT * FROM phone\n".
        "  WHERE@ LASTNAME='KRAEMER'\n".
        "  ORDER BY FIRSTNAME";
if (! ($result = sesam_query($stmt)))
  PrintReturncode($stmt);
?>
```

Voir aussi : **sesam_errormsg()** pour un accès simplifié aux messages d'erreur.

sesam_fetch_result (PHP 3 CVS only)

Retourne tout ou partie d'un résultat SESAM

```
mixed sesam_fetch_result (string result_id [, int max_rows])
```

sesam_fetch_result() retourne un tableau avec les lignes du résultat identifié par *result_id*, éventuellement limité à un maximum de *max_rows* Noteaz que les lignes et les colonnes sont indexées à partir de 0.

Tableau 1. Résultat de sesam_fetch_result()

Elément du tableau	Contents
int $arr["count"]	Nombre de colonnes dans le résultat (ou zéro si c'était une requête immédiate).
int $arr["rows"]	Nombre de ligne dans le résultat (entre zéro et *max_rows*)

Elément du tableau	Contents
boolean $arr["truncated"]	TRUE si le nombre de ligne était d'au moins *max_rows*, FALSE sinon. Notez que même si cette valeur est à TRUE, le prochain appel à **sesam_fetch_result()** peut retourner aucune ligne parce qu'il n'y a plus d'entrées.
mixed $arr[col][row]	les valeurs du résultat à la ligne row et colonne col. Le résultat est un tableau multidimensionnel. row va de 0 à $arr["rows"]-1, et col de 0 à $arr["count"]-1). Les champs peuvent être vides : vous devez vérifier leur existence avec la fonction **isset()**. Le type retourné dépend du type SQL déclaré pour cette colonne (voir Introduction SESAM pour connaître les conversions utilisées). Les champs multiples SESAM sont traités comme des séquences de colonnes.

Notez que la quantité de mémoire utilisée par des requêtes peut se révéler gigantesque. Utilisez alors *max_rows* pour limiter le nombre maximum de lignes retournées, à moins que vous ne soyez absolument sÛr que votre résultat ne consommera pas toute la mémoire disponible.

Voir aussi : **sesam_fetch_row()**, et **sesam_field_array()** pour vérifier les champs multiples. Voyez **sesam_query()** pour une exemple complet avec **sesam_fetch_result()**.

sesam_affected_rows (PHP 3 CVS only)

Lit le nombre de lignes affectées par une requête immédiate

```
int sesam_affected_rows (string result_id)
```

result_id est un identifiant valide de résultat, retourné par **sesam_query()**.

sesam_affected_rows() retourne le nombre de lignes affectées par la requête associée à *result_id*.

sesam_affected_rows() ne retourne de valeur cohérente que lorsqu'utilisée avec une requête immédiate (INSERT/UPDATE/DELETE), car SESAM ne fournit aucune information de nombre de lignes affectées pour les requêtes de selection.

Voir aussi : **sesam_query()** et **sesam_execimm()**

```
<?php
$result = sesam_execimm ("DELETE FROM PHONE WHERE LASTNAME = '".strtoupper($name)."'");
if (! $result) {
    ... error ...
}
print sesam_affected_rows($result)." entries with last name ".$name." deleted.\n";
?>
```

sesam_errormsg (PHP 3 CVS only)

retourne le message d'erreur

```
string sesam_errormsg (void)
```

sesam_errormsg() retourne le message d'erreur SESAM associé à la dernière requête SQL.

```
<?php
if (!sesam_execimm($stmt))
  printf("%s<br>\n", sesam_errormsg());
?>
```

Voir aussi : **sesam_diagnostic()** pour la liste complète des états de requêtes SQL.

sesam_field_array (PHP 3 CVS only)

Retourne des informations sur une colonne

```
array sesam_field_array (string result_id)
```

result_id is a valid result id returned by **sesam_query()**.

sesam_field_array() retourne un tableau contenant les informations (nom de colonne, type, précision...) sur une colonne dans le résultat associé à *result_id*.

Tableau 1. Informations retournées par sesam_field_array()

Index	Contenu
int $arr["count"]	Nombre total de colonnes dans le résultat (ou zéro si la requête était immédiate). Les champs multiples de SESAM sont linéarisés, et traités comme autant de colonnes.
string $arr[col]["name"]	Le nom de la colonne `col`, avec `col` qui vaut entre 0 et `$arr["count"]-1`. La valeur retournée peut être une chaîne vide (pour les colonnes dynamiquement générées). Les champs multiples SESAM sont linéarisés, et traités comme autant de colonnes, avec le même nom.
string $arr[col]["count"]	L'attribut "count" décrit le facteur de répétition quand la colonne a été déclarée comme un champs multiple. Généralement, cet attribut est à 1. La première colonne d'un champs multiple contient le nombre de répétitions, tandis que les colonnes suivantes ont un facteur de répétition mis à 1. Ceci peut être utilisé pour detecter les champs multiples. Reportez vous à l'exemple de la fonction **sesam_query()** pour avoir un exemple d'utilisation.
string $arr[col]["type"]	Type de variable PHP pour les données de la colonne `col`, où `col` vaut de 0 à `$arr["count"]-1`. La valeur retournée peut être l'une de celles-ci : • "integer" • "double" • "string" , suivant le type de données SQL. Les champs multiples SESAM sont linéarisés et traités comme autant de colonnes ayant le même type PHP.

Index	Contenu
string $arr[col]["sqltype"]	Type dedonnées SQL de la colonne `col`, où `col` vaut de 0 à `$arr["count"]-1`. La valeur retournée peut être l'une de celle-ci : • "CHARACTER" • "VARCHAR" • "NUMERIC" • "DECIMAL" • "INTEGER" • "SMALLINT" • "FLOAT" • "REAL" • "DOUBLE" • "DATE" • "TIME" • "TIMESTAMP" , décrivant le type de données SQL. Les champs multiples SESAM sont linéarisés et traités comme autant de colonnes du même type.
string $arr[col]["length"]	La taille de l'attribut, au sens SQL, de la colonne `col`, où `col` vaut de 0 à `$arr["count"]-1`. La longueur est utilisée avec les champs "CHARACTER" et "VARCHAR", pour spécifier la taille maximale de la colonne. Les champs multiples SESAM sont linéarisés et traités comme autant de colonnes ayant la même taille SQL.
string $arr[col]["precision"]	La précision de la colonne `col`, au sens SQL, où `col` vaut de 0 à `$arr["count"]-1`. La précision est utilisée avec les champs numériques et de date. Les champs multiples SESAM sont linéarisés et traités comme autant de colonnes ayant la même précision SQL.
string $arr[col]["scale"]	L'échelle de la colonne `col`, au sens SQL, où `col` vaut de 0 à `$arr["count"]-1`. L'échelle est utilisée avec les champs numériques. Les champs multiples SESAM sont linéarisés et traités comme autant de colonnes ayant la même échelle SQL.

Voir aussi **sesam_query()**, pour un exemple d'utilisation de **sesam_field_array()**.

sesam_fetch_row (PHP 3 CVS only)

Lit une ligne dans un tableau

```
array sesam_fetch_row (string result_id [, int whence [, int offset]])
```

sesam_fetch_row() retourne un tableau qui correspond à la ligne lue dans le résultat *`result_id`*, ou FALSE s'il n'y a plus de ligne.

Le nombre de colonnes du résultat est retourné dans un élémen du tableau associatif retourné $array["count"]. Comme certaines lignes peuvent être vides, la fonction **count()** ne peut être utilisée avec le tableau ainsi retourné par **sesam_fetch_row()**.

result_id est un identifiant de résultat valide retourné par **sesam_query()** (avec une requête de selection seulement!).

whence est un parmètre optionnel lors d'une opération de lecture sur un curseur à défilement, qui peut prendre une des valeurs suivantes :

Tableau 1. Valeurs valides pour *whence*

Valeur	Constante	Signification
0	SESAM_SEEK_NEXT	Lecture séquentielle (après la lecture, la position est déplacé à SESAM_SEEK_NEXT)
1	SESAM_SEEK_PRIOR	Lecture séquentielle à rebours (après la lecture, la position est déplacé à SESAM_SEEK_PRIOR)
2	SESAM_SEEK_FIRST	Repositionnement au début (après la lecture, la position est déplacée à SESAM_SEEK_NEXT)
3	SESAM_SEEK_LAST	Repositionnement à la fin (après la lecture, la position est déplacée à SESAM_SEEK_PRIOR)
4	SESAM_SEEK_ABSOLUTE	Repositionnement absolu à *offset* (index commencant à 0. Après la lecture, la position est placé à SESAM_SEEK_ABSOLUTE, et le pointeur interne est auto-incrémenté).
5	SESAM_SEEK_RELATIVE	Repositionnement relatif à *offset*, où *offset* peut être positif ou négatif

Ce paramètre n'est valable que pour les curseurs à défilement.

Lors de l'utilisation de curseurs à défilement, le curseur peut être librement repositionné. Si le paramètre *whence* est omis, les valeur par défaut seront utilisées (initialisées à : SESAM_SEEK_NEXT, et modifiée par **sesam_seek_row()**). Si *whence* est fourni, sa valeur remplacera les valeurs par défaut.

offset est un parmètre optionnel qui n'est utilisé (et nécessaire) que si *whence* vaut soit SESAM_SEEK_RELATIVE ou SESAM_SEEK_ABSOLUTE. Ce paramètre n'est valable que pour les curseurs à défilement.

sesam_fetch_row() lit une ligne de données dans le résultat *result_id*. La ligne est retournée sous forme d'un tableau (indexé de 0 à $array["count"]-1). Les champs peuvent être vides : il faut vous assurer de leur existence en utilisant la fonction **isset()**. Le type de la valeur retournée dépend du type SQL déclaré dans la base (voir introduction SESAM pour connaître les conversion utilisées). Les champs multiples SESAM sont linéarisés, et traités comme autant de colonnes.

Les prochains appels à **sesam_fetch_row()** liront la prochaine ligne (ou la précédente, ou la n-ième, suivant le type de défilement) dans le résultat, ou FALSE, s'il n'y a plus de lignes.

Exemple 1. Exemple avec sesam_fetch_row()

```
<?php
$result = sesam_query ("SELECT * FROM phone\n".
                       "  WHERE LASTNAME='".strtoupper($name)."'\n".
                       "  ORDER BY FIRSTNAME", 1);
if (! $result) {
    ... erreur ...
}
// Affiche la table dans l'ordre inverse
print "<TABLE BORDER>\n";
$row = sesam_fetch_row ($result, SESAM_SEEK_LAST);
while (is_array($row)) {
    print " <TR>\n";
```

```
    for($col = 0; $col < $row["count"]; ++$col) {
        print "  <TD>".htmlspecialchars($row[$col])."</TD>\n";
    }
    print " </TR>\n";
    // utilise la valeur implicite de SESAM_SEEK_PRIOR
    $row = sesam_fetch_row ($result);
}
print "</TABLE>\n";
sesam_free_result ($result);
?>
```

Voir aussi : **sesam_fetch_array()** qui retourne un tableau associatif, et **sesam_fetch_result()** qui retourne plusieurs lignes en même temps.

sesam_fetch_array (PHP 3 CVS only)

Lit une ligne dans un tableau associatif

```
array sesam_fetch_array (string result_id [, int whence [, int offset]])
```

sesam_fetch_array() retourne un tableau qui correspond à la ligne lue dans le résultat *result_id*, ou FALSE si il n'y a pas d'autres lignes.

sesam_fetch_array() est une version alternative de **sesam_fetch_row()**. Au lieu de stocker les données dans un tableau à indice numérique, il enregistre les données dans un tableau associatif, en utilisant les noms des champs comme clés.

result_id est un identifiant de résultat valide retourné par **sesam_query()** (avec une requête de selection seulement!).

Pour connaître les valeurs valides des options *whence* et *offset*, reportez vous à **sesam_fetch_row()**.

sesam_fetch_array() lit une ligne de données dans le résultat *result_id*. La ligne est retournée sous forme d'un tableau associatif. Chaque colonne est enregistrée avec leur nom comme index. Les noms des colonnes sont convertis en minuscules.

Les colonnes sans noms (par exemple, les résultats d'opérations arithmétiques) et les champs vides ne sont pas stockés dans ce tableau. De plus, si deux colonnes ont le même noms, la dernière colonne écrasera la précédente. Dans cette situation, utilisez de préférence **sesam_fetch_row()** ou bien, faîtes un alias de la colonne.

```
SELECT TBL1.COL AS FOO, TBL2.COL AS BAR FROM TBL1, TBL2
```

Une gestion spéciale permet de lire les champs multiples, qui sinon, auraient toutes le même nom. Pour chaque colonne d'un champs multiple, le nom d'index est créé en ajoutant le numéro de sous-index à la suite du nom de la colonne. Ces sous indices sont numérotés à partir de 1.

```
CREATE TABLE ... ( ... MULTI(3) INT )
```

Les index associatifs utilisé pour les valeurs individuelles du champs multiple sont : "multi(1)", "multi(2)", et "multi(3)", respectivement.

Les prochains appels à **sesam_fetch_array()** liront la prochaine ligne (ou la précédente, ou la n-ième, suivant les attributs de défilement), jusqu'à ce qu'il n'y ait plus de lignes.

Exemple 1. Exemple avec sesam_fetch_array()

```
<?php
$result = sesam_query ("SELECT * FROM phone\n".
                       "  WHERE LASTNAME='".strtoupper($name)."'\n".
                       "  ORDER BY FIRSTNAME", 1);
if (! $result) {
    ... error ...
}
// Affiche la table
```

```
print "<TABLE BORDER>\n";
while (($row = sesam_fetch_array ($result)) && count($row) > 0) {
    print " <TR>\n";
    print "  <TD>".htmlspecialchars($row["firstname"])."</TD>\n";
    print "  <TD>".htmlspecialchars($row["lastname"])."</TD>\n";
    print "  <TD>".htmlspecialchars($row["phoneno"])."</TD>\n";
    print " </TR>\n";
}
print "</TABLE>\n";
sesam_free_result ($result);
?>
```

Voir aussi : **sesam_fetch_row()** qui retourne un tableau numérique.

sesam_seek_row (PHP 3 CVS only)

Déplace un curseur à défilement

```
boolean sesam_seek_row (string result_id, int whence [, int offset])
```

result_id est un indentifiant de résultat valide (requête de sélection, et curseur à défilement créé avec **sesam_query()**).

whence modifie la valeur globale par défaut pour le type de défilement, spécifie le type de défilement à utiliser lors des opérations de lectures ultérieurs. Les valeurs valides sont les suivantes :

Tableau 1. Valeurs valides pour *whence*

Valeur	Constante	Signification
0	SESAM_SEEK_NEXT	Lecture séquentielle (après la lecture, la position est déplacé à SESAM_SEEK_NEXT)
1	SESAM_SEEK_PRIOR	Lecture séquentielle à rebours (après la lecture, la position est déplacé à SESAM_SEEK_PRIOR)
2	SESAM_SEEK_FIRST	Repositionnement au début (après la lecture, la position est déplacée à SESAM_SEEK_NEXT)
3	SESAM_SEEK_LAST	Repositionnement à la fin (après la lecture, la position est déplacée à SESAM_SEEK_PRIOR)
4	SESAM_SEEK_ABSOLUTE	Repositionnement absolu à *offset* (index commencant à 0. Après la lecture, la position est placé à SESAM_SEEK_ABSOLUTE, et le pointeur interne est auto-incrémenté).
5	SESAM_SEEK_RELATIVE	Repositionnement relatif à *offset*, où *offset* peut être positif ou négatif

offset est optionnel. Il ne sert que lorsque *whence* vaut soit SESAM_SEEK_RELATIVE, soit SESAM_SEEK_ABSOLUTE.

sesam_free_result (PHP 3 CVS only)

Libère les ressources

```
int sesam_free_result (string result_id)
```

sesam_free_result() libère les ressources reservées par la requête *`result_id`*. retourne `FALSE` en cas d’erreur.

LXXVIII. Sessions

La gestion des sessions avec PHP est un moyen de sauver des informations entre deux accès. Cela permet notamment de construire des applications personnalisées, et d'accroître l'attrait de votre site.

Si vous connaissez déjà la gestion des sessions avec phplib, vous remarquerez que certains concepts sont similaires.

Chaque visiteur qui accède à votre site se voit assigner un numéro d'identifiant, appelé plus loin "identifiant de session". Celui-ci est enregistré soit dans un cookie, chez le client, soit dans l'URL.

Les sessions vous permettront d'enregistrer des variables pour les préserver et les réutiliser tout au long de la visites de votre site. Lorsqu'un visiteur accèdere à votre site, PHP vérifiera automatiquement (si session.auto_start est à 1) ou manuellement (explicitement avec **session_start()** ou implicitement avec **session_register()**) si une session a déjà été ouverte. Si une telle session existe déjà, l'environnement précédent sera recréé.

Toutes les variables à enregistrer seront enregistrées sur le disque à la fin de chaque requête. Les variables enregistrées mais non définies seront marquées comme telles. Lors des accès ultérieurs, elles ne seront définies que si l'utilisateur le fait.

Les options `track_vars` et `gpc_globals` modifient la façon dont les variables sont rechargées.

> **Note :** Depuis PHP 4.0.3, `track_vars` est toujours activée.

Si `track_vars` est activée, et `register_globals` désactivée, alors les variables de session seront accessibles uniquement dans le tableau associatif global $HTTP_STATE_VARS. Les variables de session lues seront disponibles dans $HTTP_STATE_VARS.

Exemple 1. Enregistrer une variable lorsque l'option `track_vars` est activée

```
<?php
  session_register("compte");
  $HTTP_SESSION_VARS["compte"]++;
?>
```

Si `register_globals` est activée, alors les variables de session seront placées dans les variables globales associées.

Exemple 2. Enregistrer une variable lorsque `register_globals` est activée

```
<?php
  session_register("compte");
  $compte++;
?>
```

Si les deux options `track_vars` et `register_globals` sont activées, alors les variables globales et $HTTP_STATE_VARS contiendront les valeurs de session.

Il y a deux modes de propagation de l'identifiant de session :

- Cookies
- Paramètre URL

Le module de session supporte les deux techniques. La méthode par cookies est optimale, mais étant donné son peu de fiabilité (les clients peuvent refuser ou effacer les cookies), on ne peut pas se contenter de cette technique. La deuxième méthode place l'identifiant de session directement dans l'URL.

PHP est capable de gérer ceci de manière transparente, lorsque vous le compilez avec l'option `--enable-trans-sid`. Dans ce cas, les URL relatives seront modifiées pour contenir l'identifiant de session automatiquement. Sinon, vous

pouvez toujours utiliser la constante `SID`, qui sera définie si le client n'envoie pas le cookie approprié. `SID` prend la forme de `session_name=session_id`, ou bien, c'est une chaîne vide.

> **Note :** La fonction qui gérera l'écriture des données ne sera appelée qu'une fois que le script aura envoyé toutes ses données. Ainsi, les affichages tentés par cette fonction ne pourront jamais être reçus par le navigateur. Si un tel affichage est nécessaire, il est conseillé d'écrire les debugs dans un fichier.

L'exemple suivant montre comment enregistrer une variable, et comment relier correctement des pages avec SID.

Exemple 3. Compter le nombre de hits d'un utilisateur.

```
<?php
  session_register("compteur");
  $compteur++;
?>
Salut visiteur, vous avez vu cette page <?php echo $compteur; ?> times.<P>
<php?
# le <?=SID> est nécessaire pour transmettre l'identifiant de session
# au cas où les utilisateurs auraient inactivé les cookies
?>

Pour continuer, <A HREF="nextpage.php?<?=SID"?>clique ici</?>
```

Le `<?=SID->` n'est pas nécessaire, si l'option `--enable-trans-sid` a été utilisée pour compiler PHP.

> **Note :** Les URL absolues sont considérées comme des sites externes, et PHP ne leur attribuera pas le SID, qui pourrait représenter un risque de trou de sécurité.

Pour enregistrer ces informations dans une base de données, il vous faut utiliser la fonction **session_set_save_handler()**. Il faudra alors implémenter la fonction suivante pour l'adapter à MySQL ou à toute autre base de données :

Le système de gestion des sessions dispose d'un grand nombre d'options, qui sont placées dans le fichier `php.ini`. En voici un survol rapide :

- `session.save_handler` définit les noms des fonctions qui seront utilisées pour enregistrer et retrouver les données associées à une session. Par défaut, les sessions sont enregistrées dans des `fichiers`.
- `session.save_path` définit l'argument qui est passé à la fonction de sauvegarde. Si vous utilisez la sauvegarde par fichier, cet argument est le chemin jusqu'au dossier où les fichiers sont créés. Par défaut, le dossier est `/tmp`.

> **Avertissement**
>
> Si le dossier que vous utilisez a les droits de lecture universels, comme `/tmp` (valeur par défaut), les autres utilisateurs du serveur peuvent aussi lire ces fichiers, et s'immiscer dans vos sessions.

- `session.name` spécifie le nom de la session, qui sera utilisé comme nom de cookie. Par défaut : `PHPSESSID`.
- `session.auto_start` indique qu'une session doit commencer automatiquement lors de la premier requête. Par défaut, la valeur est à `0` (inactivé).
- `session.lifetime` fixe la durée de vie, en secondes, du cookie envoyé au client. La valeur 0 signifie "jusqu'à ce que le client soit fermé". Par défaut à `0` (inactivé).

- `session.serialize_handler` définit le nom de la fonction qui sera utilisée pour enregistrer et relire les donnés. Actuellement, c'est un format interne de PHP (nom : `php`) et WDDX (nom : `wddx`). WDDX n'est utilisable que si PHP a été compilé avec le support WDDX. Par défaut, c'est le mode `php` qui est sélectionné.
- `session.gc_probability` précise la probabilité que la routine gc (garbage collection) soit lancée, en pourcentage. Par défaut, la valeur est à `1`.
- `session.gc_maxlifetime` fixe la durée, en secondes, au-delà de laquelle les données considérées comme inutiles seront supprimées.
- `session.referer_check` détermine si l'identifiant de session (session id) utilisé par des sites externes seront éliminés. Si les identifiants de session sont propagés avec la méthode des URL, des utilisateurs qui n'en connaîtraient pas l'utilité risquent de divulguer ces valeurs, et cela mènera à des problèmes de sécurité. Cette option y remédie. Par défaut : `0`.
- `session.entropy_file` est le chemin jusqu'à une source externe (fichier) d'entropie, qui sera utilisée lors de la création de l'identifiant de session. Par exemple, `/dev/random` ou `/dev/urandom` qui sont disponibles sur de nombreux systèmes UNIX.
- `session.entropy_length` précise le nombre d'octets qui seront lus dans le fichier ci-dessus. Par défaut, `0` (inactivé).
- `session.use_cookies` cookies indique si le module doit utiliser des cookies pour enregistrer l'identifiant de session chez le client. Par défaut, `1` (activé).
- `session.cookie_path` spécifie le chemin à utiliser avec session_cookie. Par défaut, `/`.
- `session.cookie_domain` spécifie le domaine à utiliser avec session_cookie. Par défaut, rien du tout.
- `session.cache_limiter` spécifie le contrôle du cache, à utiliser avec les pages de session (nocache/private/public). Par défaut, `nocache`.
- `session.cache_expire` spécifie la durée de vie des pages de session cachées, en minutes, mais sans que cela ait d'effets sur le limiteur "nocache". Par défaut, `180`.
- `session.use_trans_sid` indique si le support du SID est activé ou pas, lors de la compilation avec l'option `--enable-trans-sid`. Par défaut, elle vaut `1` (activée).
- `url_rewriter.tags` spefifies which html tags are rewritten to include session id if transient sid support is enabled. Defaults to `a=href,area=href,frame=src,input=src,form=fakeentry`

Note : La gestion des sessions a été ajoutée en PHP 4.0.

session_start (PHP 4 >= 4.0b1)

Initialise les données de session

```
boolean session_start (void)
```

session_start() crée une session (ou continue la session courante, en fonction de l'identifiant de session passé par une variable GET ou par un cookie)

session_start() retourne toujours TRUE.

> **Note : session_start()** a été ajoutée en PHP 4.0.

session_destroy (PHP 4 >= 4.0b1)

Détruit toutes les données enregistrées d'une session

```
boolean session_destroy (void)
```

session_destroy() détruit toutes les données associées à la session courante.

session_destroy() retourne TRUE en cas de succès, et FALSE sinon.

session_name (PHP 4 >= 4.0b1)

Affecte et/ou retourne le nom de la session courante

```
string session_name ([string name])
```

session_name()retourne le nom de la session courante. Si *name* est fourni, le nom de la session changera, et prendra la valeur fournie.

Le nom de session fait référence à l'identifiant de session dans les cookies. Il ne doit contenir que des caractères alpha-numériques; il doit être court et descriptif. (i.e. surtout pour les utilisateurs d'alertes de cookie). Le nom de session est remis à une valeur par défaut, enregistrée dans session.name au moment du démarrage. Ainsi, vous devez appeler **session_name()** à chaque requête (et avant **session_start()** ou **session_register()**).

Exemple 1. Exemple avec session_name()

```
<?php
# Change le nom de la session à WebsiteID
$previous_name = session_name ("WebsiteID");
echo "L'ancien nom de la session était $previous_name<P>";
?>
```

> **Note : session_name()** a été ajoutée en PHP 4.0.

session_module_name (PHP 4 >= 4.0b1)

Affecte et/ou retourne le module courant de session courante

```
string session_module_name ([string module])
```

session_module_name() affecte et/ou retourne le module courant de session courante. Si *module* est fourni, ce module sera utilisé à la place du courant.

Note : session_module_name() a été ajoutée en PHP 4.0.

session_save_path (PHP 4 >= 4.0b1)

Affecte et/ou retourne le chemin de sauvegarde de la session courante

```
string session_save_path ([string path])
```

session_save_path() retourne le chemin du dossier utilisé pour enregistrer les données de sessions. Si *path* est fourni, le chemin prendra alors la valeur fournie.

Note : Sur certains systèmes d'exploitation, il vous faudra peut être fournir un chemin vers un système de sauvegarde qui peut gérer efficacement de grandes quantités de petits fichiers : par exemple, sous Linux, reiserfs peut être plus efficace que ext2fs.

Note : session_save_path() a été ajoutée en PHP 4.0.

session_id (PHP 4 >= 4.0b1)

Affecte et/ou retourne l'identifiant de session courante

```
string session_id ([string id])
```

session_id() retourne l'identifiant de session courante. Si *id* est fourni, il remplacera l'identifiant courant de la session.

La constante SID peut aussi être utilisée pour retrouver le nom de la session courante et son identifiant, comme chaîne à ajouter dans les URL.

Note : session_id() a été ajoutée en PHP 4.0.

session_register (PHP 4 >= 4.0b1)

Enregistre une variable dans la session courante

```
boolean session_register (mixed name [, mixed ...])
```

session_register() enregistre une variable avec le nom *name* dans la session courante. **session_register()** accepte un nombre d'arguments variable. Ces arguments peuvent être soit des chaînes représentants le nom de la variable, soit un tableau, contenant des chaînes ou d'autres tableaux (cas d'un tableau récursif).

session_register() retourne TRUE lorsque les variables sont correctement enregistrées.

> **Note :** Actuellement, il n'est pas possible d'enregistrer des variables de ressources dans une session. Par exemple, vous ne pouvez pas créer une connexion à une base de données dans un script, la stocker dans une session, et retrouver cette connexion lors de la prochaine utilisation de la session. Les fonctions qui retournent des ressources PHP sont identifées par leur valeur de retour `resource`, dans leur fonction de définition. Vous pouvez aussi les retrouver dans les annexes.

> **Note : session_register()** a été ajoutée en PHP 4.0.

session_unregister (PHP 4 >= 4.0b1)

Supprime une variable dans la session courante

```
boolean session_unregister (string name)
```

session_unregister() supprime la variable nommée *name* dans la session courante.

session_unregister() retourne TRUE lorsque la variable a été correctement supprimée de la session.

> **Note : session_unregister()** a été ajoutée en PHP 4.0.

session_unset (PHP 4 >= 4.0b4)

Détruit toutes les variables de session

```
void session_unset (void)
```

session_unset() détruit toutes les variables de session couramment enregistrées.

session_is_registered (PHP 4 >= 4.0b1)

Indique si une variable a été enregistrée dans la session ou pas

```
boolean session_is_registered (string name)
```

session_is_registered() retourne TRUE s'il y a une variable du nom de *name* enregistrée dans la session courante.

Note : session_is_registered() a été ajoutée en PHP 4.0.

session_get_cookie_params (PHP 4 >= 4.0RC2)

Lit les paramètres du cookie de session

```
array session_get_cookie_params  (void)
```

session_get_cookie_params() retourne un tableau avec les paramètres du cookie de la session courante. Le tableau contient les éléments suivants :

- "lifetime" - La durée de vie du cookie.
- "path" - Le chemin de stockage du cookie.
- "domain" - Le domaine du cookie.

session_set_cookie_params (PHP 4 >= 4.0b4)

Modifie les paramètres du cookie de session

```
void session_set_cookie_params (int lifetime [, string path [, string domain]])
```

session_set_cookie_params() modifie les paramètres du cookie de session, tels qu'ils ont été définis dans le fichier `php.ini`. L'effet de cette fonction ne dure que le temps du script.

session_decode (PHP 4 >= 4.0b1)

Décode les données de session à partir d'une chaîne

```
boolean session_decode (string data)
```

session_decode() décode les données de session à partir de la chaîne *`data`*, et affecte les valeurs des variables de session.

Note : session_decode() a été ajoutée en PHP 4.0.

session_encode (PHP 4 >= 4.0b1)

Encode les données de session dans une chaîne

```
boolean session_encode (void)
```

session_encode() retourne les données de session dans une chaîne.

Note : session_encode() a été ajoutée en PHP 4.0.

session_set_save_handler (PHP 4 >= 4.0b4)

Définit les fonctions utilisateurs de stockage des sessions

```
void session_set_save_handler  (string open, string close, string read, string write,
string destroy, string gc)
```

session_set_save_handler() définit les fonctions utilisateurs de stockage et chargement des sessions. Cela est particulièrement pratique pour spécifier une autre méthode de stockage que celle fournie en standard avec PHP. Notamment, il est possible de stocker les sessions dans une base de données.

> **Note :** Vous devez donner à l'option de configuration *session.save_handler* la valeur de *user* dans votre fichier php.ini pour que **session_set_save_handler()** soit effective.

L'exemple suivant fournit un exemple de stockage de session dans un fichier, similaire aux fonctions standards de PHP. Cet exemple peut être facilement étendu pour utiliser un stockage en base de données, en utilisant votre base préférée.

Exemple 1. Exemple avec session_set_save_handler()

```
<?php
function open ($save_path, $session_name) {
  global $sess_save_path, $sess_session_name;
  $sess_save_path = $save_path;
  $sess_session_name = $session_name;
  return(TRUE);
}
function close() {
  return(TRUE);
}
function read ($id) {
  global $sess_save_path, $sess_session_name;
  $sess_file = "$sess_save_path/sess_$id";
  if ($fp = @fopen($sess_file, "r")) {
    $sess_data = fread($fp, filesize($sess_file));
    return($sess_data);
  } else {
    return("");
  }
}
function write ($id, $sess_data) {
  global $sess_save_path, $sess_session_name;
  $sess_file = "$sess_save_path/sess_$id";
  if ($fp = @fopen($sess_file, "w")) {
    return(fwrite($fp, $sess_data));
  } else {
    return(FALSE);
  }
}
function destroy ($id) {
  global $sess_save_path, $sess_session_name;
  $sess_file = "$sess_save_path/sess_$id";
  return(@unlink($sess_file));
}
/*************************************************
 * ATTENTION - Vous devez implémenter une routine  *
 * d'entretien des sessions ici.                   *
 *************************************************/
```

```
function gc ($maxlifetime) {
  return TRUE;
}
session_set_save_handler ("open", "close", "read", "write", "destroy", "gc");
session_start();
// utilisez vos sessions normalement
?>
```

session_cache_limiter (PHP 4 >= 4.0.3)

Lit et/ou modifie le limiteur de cache

```
string session_cache_limiter ([string cache_limiter])
```

session_cache_limiter() retourne le nom du limiteur de cache courant. Si *cache_limiter* est spécifié, le nom du limiteur de cache est remplacé par cette nouvelle valeur.

Le limiteur de cache contrôle l'envoi des en-têtes HTTP envoyés au client. Ces en-têtes déterminent les règles de mise en cache des pages. En utilisant la valeur de `nocache`, par exemple, vous désactiverez la mise en cache côté client. La valeur de `public`, cependant, le permettra. `private` aussi, tout en étant légèrement plus restrictive que `public`.

Le limiteur de cache est remis à sa valeur par défaut, stockée dans `session.cache_limiter`, initialisée au lancement. Vous devrez donc appeler **session_cache_limiter()** pour chaque requête (et avant l'appel à **session_start()**).

Exemple 1. Exemples avec session_cache_limiter()

```
<?php
# Met le limiteur de cache à 'private'
session_cache_limiter('private');
$cache_limiter = session_cache_limiter();
echo "Le limiteur de cache vaut actuellement $cache_limiter<P>";
?>
```

Note : session_cache_limiter() a été ajoutée dans PHP 4.0.3.

LXXIX. Mémoire partagée

Shmop est un ensemble de fonctions simples pour gérer la mémoire partagée avec PHP (lecture, écriture, création et suppressions de segments de mémoire paragée UNIX). Ces fonctions ne fonctionnent pas sous Windows, car ce système d'exploitation ne supporte pas la mémoire partagée. Pour utiliser les fonctions shmop, compilez PHP avec l'option --enable-shmop parameter.

> **Note :** Toutes les fonctions décrites ci-dessous commencent par `shm_` poru les versions jusqu'à PHP 4.0.3, mais en PHP 4.0.4 et plus récent, elles sont préfixées par `shmop_`.

Exemple 1. Introduction à la mémoire partagée

```
<?php
// Crée 100 octets de mémoire partagée avec
// un identifiant système "0xff3"
$shm_id = shmop_open(0xff3, "c", 0644, 100);
if(!$shm_id) {
echo "Impossible de créer la mémoire partagée\n";
}
// Lire la taille de la mémoire partagée
$shm_size = shmop_size($shm_id);
echo "Un bloc de SHM de taille ".$shm_size. " a été créé.\n";
// Ecriture d'une chaîne de test dans ce segment
$shm_bytes_written = shmop_write($shm_id, "mon bloc de mémoire partagée", 0);
if($shm_bytes_written != strlen("mon bloc de mémoire partagée")) {
echo "Impossible d'écrire toutes les données en mémoire\n";
}
// Lecture du segment
$my_string = shmop_read($shm_id, 0, $shm_size);
if(!$my_string) {
echo "Impossible de lire toutes les données en mémoire\n";
}
echo "Les données mis en mémoire partagées sont : ".$my_string."\n";
//Maintenant, effacons le bloc, et fermons le segment de mémoire
if(!shmop_delete($shm_id)) {
echo "Impossible d'effacer le segment de mémoire";
}
shmop_close($shm_id);
?>
```

shmop_open (PHP 4 >= 4.0.4)

Crée ou ouvre un bloc de mémoire partagée

```
resource shmop_open (int key, string flags, int mode, int size)
```

shmop_open() peut créer ou ouvrir un bloc de mémoire partagée.

shmop_open() prend 4 paramètres: la clé, qui sera l'identifiant système pour le bloc. Ce paramètre peut être passé comme un décimal ou un héxadécimal. Le deuxième paramètre est un groupe d'options :

- "a" pour accès (utilise IPC_EXCL) utilisez cette option pour ouvrir un bloc déjà existant.
- "c" pour création (utilise IPC_CREATE) utilisez cette option pour créer un nouveau bloc.

Le troisième paramètre est le mode, c'est à dire les permissions que vous donnez à ce bloc. Ce sont les mêmes que pour les fichiers. Ces permissions doivent être passées sous forme d'octal (i.e. 0644). Le dernier paramètre est la taille du bloc de mémoire, en ocets.

> **Note :** Les troisième et quatrième paramètres doivent être passés à 0 si vous voulez ouvrir un bloc de mémoire partagée déjà existant. En cas de succès **shmop_open()** retourne un identifiant que vous pouvez utiliser pour accéder à la mémoire que vous venez de créer.

Exemple 1. Créer un nouveau bloc

```
<?php
$shm_id = shmop_open(0x0fff, "c", 0644, 100);
?>
```

Cet exemple ouvre un nouveau bloc de mémoire partagée, dont l'identifiant est 0x0fff.

shmop_read (PHP 4 >= 4.0.4)

Lit un bloc

```
string shmop_read (resource shmid, int start, int count)
```

shmop_read() lit une chaîne dans une bloc de mémoire partagée.

shmop_read() prend 3 paramètres: *`shmid`*, qui est un identifiant de mémoire partagée, créé par **shmop_open()**, *`start`* qui est la position à partir de laquelle on commence à lire dans la mémoire et *`count`*, le nombre d'octets à lire.

Exemple 1. Lire un bloc de mémoire partagée

```
<?php
$shm_data = shmop_read($shm_id, 0, 50);
?>
```

Cet exemple lit 50 octets dans le bloc de mémoire partagée `$shm_data`.

shmop_write (PHP 4 >= 4.0.4)

Ecrire dans un bloc de mémoire partagée

```
int shmop_write (resource shmid, string data, int offset)
```

shmop_write() écrit une chaîne dans un bloc de mémoire partagée.

shmop_write() prend 3 paramètres: *`shmid`*, qui est un identifiant de mémoire partagée, créé par **shmop_open()**, *`data`* qui est la chaîne à écrire dans la mémoire et *`offset`*, la position à partir de laquelle il faut commencer à écrire.

Exemple 1. Ecrire un bloc de mémoire partagée

```
<?php
$shm_bytes_written = shmop_write($shm_id, $my_string, 0);
?>
```

Cet exemple écrit les données de la chaîne `$my_string` dans un bloc de mémoire partagée. `$shm_bytes_written` représentera le nombre d'octets écrits.

shmop_size (PHP 4 >= 4.0.4)

Lire la taille du bloc de mémoire partagée

```
int shmop_size (resource shmid)
```

shmop_size() sert à connaître la taille, en octets, d'un bloc de mémoire partagée.

shmop_size() prend comme argument *`shmid`*, un identifiant de bloc de mémoire partagée créé par **shmop_open()**, et retourne un entier, qui représente le taille de ce bloc.

Exemple 1. Lire la taille d'un bloc de mémoire partagée

```
<?php
$shm_size = shmop_size($shm_id);
?>
```

Cet exemple lit la taille du bloc identifié par `$shm_id`, et le place dans `$shm_size`.

shmop_delete (PHP 4 >= 4.0.4)

Détruit un bloc de mémoire partagée

```
int shmop_delete (resource shmid)
```

shmop_delete() sert à détruire un bloc de mémoire partagée.

shmop_delete() prend un identifiant de mémoire partagée *`shmid`*, créé par **shmop_open()**. En cas de succès, la fonction retourne 1, et sinon, 0.

Exemple 1. Effacement d’un bloc de mémoire partagée

```
<?php
shmop_delete($shm_id);
?>
```

Ce exemple efface le bloc de mémoire partagée identifié par `$shm_id`.

shmop_close (PHP 4 >= 4.0.4)

Ferme un bloc de mémoire partagée

```
int shmop_close (resource shmid)
```

shmop_close() sert à fermer un bloc de mémoire partagée.

shmop_close() prend un identifiant de mémoire partagée, *`shmid`*, créé par **shmop_open()**.

Exemple 1. Fermeture d’un bloc de mémoire partagée

```
<?php
shmop_close($shm_id);
?>
```

Cet exemple ferme le bloc de mémoire partagée identifié par `$shm_id`.

LXXX. Shockwave Flash

PHP a la capacité de créer des animations Shockwave Flash grâce au module de Paul Haeberli : libswf module. Vous pouvez télécharger libswf à http://reality.sgi.com/grafica/flash/. Une fois que vous avez libswf, tout ce qui reste à faire est de configurer PHP avec `--with-swf[=DIR]` oú DIR est le dossier qui accueille les dossiers de include et lib. Le dossier include doit contenir le fichier swf.h file et le dossier lib doit contenir le fichier libswf.a. Si vous décompressez la distribution de libswf, les deux fichiers seront dans le même dossier. Par conséquent, vous devrez les mettre dans le dossier ad hoc manuellement.

Une fois que vous avez réussi à installer PHP avec Shockwave Flash, vous pouvez créer des animations Flash avec PHP. Vous serez surpris du résultat. Essayez donc ceci :

Exemple 1. Exemple SWF

```
<?php
swf_openfile ("test.swf", 256, 256, 30, 1, 1, 1);
swf_ortho2 (-100, 100, -100, 100);
swf_defineline (1, -70, 0, 70, 0, .2);
swf_definerect (4, 60, -10, 70, 0, 0);
swf_definerect (5, -60, 0, -70, 10, 0);
swf_addcolor (0, 0, 0, 0);
swf_definefont (10, "Mod");
swf_fontsize (5);
swf_fontslant (10);
swf_definetext (11, "This be Flash wit PHP!", 1);
swf_pushmatrix ();
swf_translate (-50, 80, 0);
swf_placeobject (11, 60);
swf_popmatrix ();
for ($i = 0; $i < 30; $i++) {
    $p = $i/(30-1);
    swf_pushmatrix ();
    swf_scale (1-($p*.9), 1, 1);
    swf_rotate (60*$p,  'z');
    swf_translate (20+20*$p, $p/1.5, 0);
    swf_rotate (270*$p,  'z');
    swf_addcolor ($p, 0, $p/1.2, -$p);
    swf_placeobject (1, 50);
    swf_placeobject (4, 50);
    swf_placeobject (5, 50);
    swf_popmatrix ();
    swf_showframe ();
}
for ($i = 0; $i < 30; $i++) {
    swf_removeobject (50);
    if (($i%4) == 0) {
        swf_showframe ();
    }
}
swf_startdoaction ();
swf_actionstop ();
swf_enddoaction ();
swf_closefile ();
?>
```

Cela va produire une animation, proche de celle ci (http://www.designmultimedia.com/swfphp/test.swf) (mais traduite en anglais).

> **Note :** Le support de Flash a été ajouté dans PHP 4.0RC2.
>
> La librairie libswf n'est pas disponible pour Windows : son développement a été stoppé, et les sources ne sont plus disponibles pour permettre le portage vers d'autres systèmes.

swf_openfile (PHP 4 >= 4.0RC2)

Ouvre un nouveau fichier Shockwave Flash

```
void swf_openfile  (string filename, float width, float height, float framerate, float r,
float g, float b)
```

swf_openfile() crée un nouveau fichier *filename* de largeur *width*, et de hauteur *height*, à la vitesse de *framerate*, de couleur de fond RGB (*r*, *g*, *b*).

swf_openfile() doit être la première fonction à appeler, sous peine d'erreur mémoire (segmentation fault). Si vous voulez envoyer votre production au client HTML, utilisez le nom de fichier "php://stdout" (le support de ceci est prévue pour la version 4.0.1 et ultérieur).

swf_closefile (PHP 4 >= 4.0RC2)

Ferme le fichier courant Shockwave Flash.

```
void swf_closefile  (int [return_file])
```

swf_closefile() ferme le fichier courant, qui a été ouvert avec **swf_openfile()**. Si le paramètre *return_file* a été fourni, il contiendra le fichier SWF fermé.

Exemple 1. Création d'un fichier Flash simple, basé sur une entrée de l'utilisateur, et sauvegarde dans une base.

```
<?php
// La variable $text est fournie par l'utilisateur
// Variables globales pour l'accès à la base de données
// utilisée dans la fonction wf_savedata())
$DBHOST = "localhost";
$DBUSER = "sterling";
$DBPASS = "secret";
swf_openfile ("php://stdout", 256, 256, 30, 1, 1, 1);
    swf_definefont (10, "Ligon-Bold");
        swf_fontsize (12);
        swf_fontslant (10);
    swf_definetext (11, $text, 1);
    swf_pushmatrix ();
        swf_translate (-50, 80, 0);
        swf_placeobject (11, 60);
    swf_popmatrix ();
    swf_showframe ();
    swf_startdoaction ();
        swf_actionstop ();
    swf_enddoaction ();
$data = swf_closefile (1);
$data ?
  swf_savedata ($data) :
  die ("Error could not save SWF file");
// void swf_savedata (string data)
// Sauve le fichier généré dans la base de données
// pour accès ultérieur
function swf_savedata ($data)
{
    global $DBHOST,
           $DBUSER,
           $DBPASS;
    $dbh = @mysql_connect ($DBHOST, $DBUSER, $DBPASS);
    if (!$dbh) {
        die (sprintf ("Error [%d]: %s",
                      mysql_errno (), mysql_error ()));
```

```
    }
    $stmt = "INSERT INTO swf_files (file) VALUES ('$data')";
    $sth = @mysql_query ($stmt, $dbh);
    if (!$sth) {
        die (sprintf ("Error [%d]: %s",
                      mysql_errno (), mysql_error ()));
    }
    @mysql_free_result ($sth);
    @mysql_close ($dbh);
}
?>
```

swf_labelframe (PHP 4 >= 4.0RC2)

Nomme le frame courant.

```
void swf_labelframe  (string name)
```

swf_labelframe() donne le nom *name* au frame courant.

swf_showframe (PHP 4 >= 4.0RC2)

Affiche le frame courant.

```
void swf_showframe  (void)
```

swf_showframe() affiche le frame courant.

swf_setframe (PHP 4 >= 4.0RC2)

Fixe le frame courant.

```
void swf_setframe  (int framenumber)
```

swf_setframe() selectionne le frame *framenumber* comme frame actif.

swf_getframe (PHP 4 >= 4.0RC2)

Retourne le numéro de frame courant.

```
int swf_getframe  ()
```

swf_getframe() retourne le numéro de frame courant.

swf_mulcolor (PHP 4 >= 4.0RC2)

Fixe la couleur globale de multiplication (? : the global multiply color).

```
void swf_mulcolor  (float r, float g, float b, float a)
```

swf_mulcolor() fixe la valeur globale de multiplication (the global multiply color...) à la couleur *rgba*. Cette couleur est utilisée (implicitement) par **swf_placeobject()**, **swf_modifyobject()** et **swf_addbuttonrecord()**. La couleur d'un objet sera multipliée par *rgba* lorsque l'objet est placé sur la scène.

Note : Les valeurs de *rgba* peuvent être positives ou négatives.

swf_addcolor (PHP 4 >= 4.0RC2)

Fixe la couleur globale d'addition (? : the global add color).

```
void swf_addcolor  (float r, float g, float b, float a)
```

swf_mulcolor() fixe la valeur globale de multiplication (the global multiply color...) à la couleur *rgba*. Cette couleur est utilisée (implicitement) par **swf_placeobject()**, **swf_modifyobject()** et **swf_addbuttonrecord()**. La couleur d'un objet sera ajouté à *rgba* lorsque l'objet est placé sur la scène.

Note : Les valeurs de *rgba* peuvent être positives ou négatives.

swf_placeobject (PHP 4 >= 4.0RC2)

Place un objet sur la scène.

```
void swf_placeobject  (int objid, int depth)
```

swf_placeobject() place l'objet *objid* dans le frame courant, à la profondeur *depth*. *objid* et *depth* doivent être compris entre 1 et 65535.

swf_placeobject() utilise la couleur courante de multiplication (specifiée par **swf_mulcolor()**) et la couleur courante d'addition (specifiée par **swf_addcolor()**) pour colorer l'objet, et utilise la matrice courante pour positionner l'objet.

Note : Le support des couleurs RGBA est complet.

swf_modifyobject (PHP 4 >= 4.0RC2)

Modifie un objet.

```
void swf_modifyobject  (int depth, int how)
```

swf_modifyobject() modifie la position et/ou la couleur de l'objet situé à la profondeur de *depth*. L'argument *how* détermine ce qui doit être modifié. *how* peut prendre les valeurs de MOD_MATRIX, MOD_COLOR ou la combinaison des deux.

MOD_COLOR utilise la couleur courante de multiplication (specifiée par **swf_mulcolor()**) et la couleur courante d'addition (specifiée par **swf_addcolor()**) pour colorer l'objet, et MOD_MATRIX utilise la matrice courante pour positionner l'objet.

swf_removeobject (PHP 4 >= 4.0RC2)

Enlève un objet.

```
void swf_removeobject  (int depth)
```

swf_removeobject() enlève l'objet situé à la profondeur *depth* de la scène.

swf_nextid (PHP 4 >= 4.0RC2)

Retourne le prochain identifiant d'objet libre.

```
int swf_nextid  ()
```

swf_nextid() retourne le prochain identifiant d'objet libre.

swf_startdoaction (PHP 4 >= 4.0RC2)

Commence la déscription d'une liste d'action pour la frame courante.

```
void swf_startdoaction  ()
```

swf_startdoaction() commence la déscription d'une liste d'actions pour la frame courante. Cette fonction doit être appelée avant que les actions ne soient définies pour le cadre courant.

swf_actiongotoframe (PHP 4 >= 4.0RC2)

Joue un frame puis stoppe.

```
void swf_actiongotoframe  (int framenumber)
```

swf_actiongotoframe() se déplace jusqu'au frame *framenumber*, le joue, puis s'arrête.

swf_actiongeturl (PHP 4 >= 4.0RC2)

Retourne l'URL d'une animation Shockwave Flash.

```
void swf_actiongeturl  (string url, string target)
```

swf_actiongeturl() lit l'URL *url*, avec la destination *target*.

swf_actionnextframe (PHP 4 >= 4.0RC2)

Avance d'un frame.

```
void swf_actionnextframe ()
```

swf_actionnextframe() avance d'un frame le frame courant.

swf_actionprevframe (PHP 4 >= 4.0RC2)

Recule d'un frame.

```
void swf_actionprevframe ()
```

swf_actionnextframe() recule d'un frame le frame courant.

swf_actionplay (PHP 4 >= 4.0RC2)

Joue l'animation flash à partir du frame courant.

```
void swf_actionplay ()
```

swf_actionplay() joue l'animation Flash à partir du frame courant.

swf_actionstop (PHP 4 >= 4.0RC2)

Arrête l'animation flash.

```
void swf_actionstop ()
```

swf_actionstop() arrête l'animation Flash au frame courant.

swf_actiontogglequality (PHP 4 >= 4.0RC2)

Choisi le niveau de qualité haut ou bas.

```
void swf_actiontogglequality ()
```

swf_actiontogglequality() modifie le niveau de qualité haut ou bas.

swf_actionwaitforframe (PHP 4 >= 4.0RC2)

Ignore les actions si le frame n'est pas chargé.

```
void swf_actionwaitforframe (int framenumber, int skipcount)
```

swf_actionwaitforframe() vérifie que le frame *framenumber* a bien été chargé. Si ce n'est pas le cas, elle ignore les actions *skipcount*. Cela est très utile pour les séquences du type "Chargement...".

swf_actionsettarget (PHP 4 >= 4.0RC2)

Fixe le contexte des actions.

```
void swf_actionsettarget  (string target)
```

swf_actionsettarget() fixe le contexte des actions. Vous pouvez utiliser cette fonction pour contrôler d'autres animations Flash qui seraient en fonctionnement.

swf_actiongotolabel (PHP 4 >= 4.0RC2)

Affiche le frame nommé.

```
void swf_actiongotolabel  (string label)
```

swf_actiongotolabel() affiche le frame de nom *label*, puis stoppe.

swf_enddoaction (PHP 4 >= 4.0RC2)

Termine l'action courante.

```
void swf_enddoaction  ()
```

swf_startdoaction() termine l'action courante, démarrée par **swf_startdoaction()**.

swf_defineline (PHP 4 >= 4.0RC2)

Définit une ligne.

```
void swf_defineline  (int objid, float x1, float y1, float x2, float y2, float width)
```

swf_defineline() définit une ligne commencant aux coordonnées (*x1*, *y1*), et finissant au point de coordonnées (*x2*, *y2*). Elle aura la largeur de *width*.

swf_definerect (PHP 4 >= 4.0RC2)

Définit un rectangle.

```
void swf_definerect  (int objid, float x1, float y1, float x2, float y2, float width)
```

swf_definerect() définit un rectangle, de coin supérieur gauche aux coordoonées (*x1*,*y1*), et de coin inférieur droit aux coordonnées (*x2*, *y2*). L'épaisseur des bords est données par le paramètre *width*. *width*, 0.0 le rectangle sera rempli.

swf_definepoly (PHP 4 >= 4.0.0)

Définit un polygone.

```
void swf_definepoly  (int objid, array coords, int npoints, float width)
```

swf_definepoly() définit un polygone, dont les coordonnées des sommets sont placés dans le tableau *coords*). *npoints* est le nombre de points contenu dans le tableau *coords*. *width* est la largeur des bords du polygone. Si *width* vaut 0.0, le polygone sera rempli.

swf_startshape (PHP 4 >= 4.0RC2)

Commence une forme complexe.

```
void swf_startshape  (int objid)
```

swf_startshape() commence une forme complexe, qui sera reperé par l'identifiant d'objet *objid*.

swf_shapelinesolid (PHP 4 >= 4.0RC2)

Fixe le style courant de ligne.

```
void swf_shapelinesolid  (float r, float g, float b, float a, float width)
```

swf_shapelinesolid() permet de choisir le style de ligne, à savoir la couleur et la largeur. Si *width* vaut 0.0, les lignes ne seront pas dessinées.

swf_shapefilloff (PHP 4 >= 4.0RC2)

Inactive le remplissage.

```
void swf_shapefilloff  ()
```

swf_shapefilloff() inactive le remplissage pour la forme courante.

swf_shapefillsolid (PHP 4 >= 4.0RC2)

Fixe la couleur pour le style courant de remplissage.

```
void swf_shapefillsolid  (float r, float g, float b, float a)
```

swf_shapefillsolid() fixe la couleur pour le style courant de remplissage à *rgba*.

swf_shapefillbitmapclip (PHP 4 >= 4.0RC2)

Choisi le mode de remplissage par texture.

```
void swf_shapefillbitmapclip  (int bitmapid)
```

Choisi le mode de remplissage par texture : les espaces vides seront remplis avec la bitmap *bitmapid*.

swf_shapefillbitmaptile (PHP 4 >= 4.0RC2)

Choisi le mode de remplissage par texture repétée.

```
void swf_shapefillbitmaptile  (int bitmapid)
```

Choisi le mode de remplissage par texture : les espaces vides seront remplis avec la bitmap *bitmapid*, repétée autant de fois qu'il le faut (mode carrelage).

swf_shapemoveto (PHP 4 >= 4.0RC2)

Change la position courante.

```
void swf_shapemoveto  (float x, float y)
```

swf_shapemoveto() fixe la position courante au point de coordonnées (*x*, *y*).

swf_shapelineto (PHP 4 >= 4.0RC2)

Dessine une ligne.

```
void swf_shapelineto  (float x, float y)
```

swf_shapelineto() dessine une ligne entre la position courante et le point de coordonnées (*x*, *y*). La position courante devient alors (*x*, *y*).

swf_shapecurveto (PHP 4 >= 4.0RC2)

Dessine une courbe de Bézier quadratique entre deux points.

```
void swf_shapecurveto  (float x1, float y1, float x2, float y2)
```

swf_shapecurveto() dessine la courbe de Bézier quadratique entre les points de coordonnées (*x1* , *y1*) et (*x2*, *y2*). La position courante devient alors (*x2*, *y2*).

swf_shapecurveto3 (PHP 4 >= 4.0RC2)

Dessine une courbe Bézier cubique.

```
void swf_shapecurveto3  (float x1, float y1, float x2, float y2, float x3, float y3)
```

Dessine une courbe de Bézier cubique, en utilisant les points de coordoonnées (*x1*, *y1*) et (*x2*,*y2*) comme points de contrôle et le point de coordonnées (*x3*, *y3*) comme point final. La position finale devient alors la position courante.

swf_shapearc (PHP 4 >= 4.0RC2)

Dessine une arc de cercle.

```
void swf_shapearc  (float x, float y, float r, float ang1, float ang2)
```

swf_shapearc() dessine un arc de cercle, depuis l'angle *ang1* jusqu'à l'angle *ang2*. Le centre du cercle est aux coordonnées (*x*, *y*), et de rayon *r*.

swf_endshape (PHP 4 >= 4.0RC2)

Complète la définition de la forme courante.

```
void swf_endshape  ()
```

swf_endshape() complète la définition de la forme courante.

swf_definefont (PHP 4 >= 4.0RC2)

Définit une police.

```
void swf_definefont  (int fontid, string fontname)
```

swf_definefont() définit la police *fontname* et lui affecte l'identifiant *fontid*. Cette police devient alors la police courante.

swf_setfont (PHP 4 >= 4.0RC2)

Change la police courante.

```
void swf_setfont  (int fontid)
```

swf_setfont() remplace la police courante par la police repérée par l'identifiant *fontid*.

swf_fontsize (PHP 4 >= 4.0RC2)

Change la taille de la police.

```
void swf_fontsize  (float size)
```

swf_fontsize() remplace la taille de la police par la taille *size*.

swf_fontslant (PHP 4 >= 4.0RC2)

Change l'inclinaison de la police courante.

```
void swf_fontslant  (float slant)
```

swf_fontslant() fixe l'inclinaison de la police courante à *slant*. Les valeurs positives values créeront une inclinaison vers la droite, et les valeurs négatives, vers la gauche.

swf_fonttracking (PHP 4 >= 4.0RC2)

Change l'espacement des caractères.

```
void swf_fonttracking  (float tracking)
```

swf_fonttracking() change l'espacement, et lui affecte la valeur de *tracking*. Cette fonction sert à accroître l'espace entre les lettres et le texte. Les valeurs positives accroissent cet espace, et les valeurs négatives le réduisent.

swf_getfontinfo (PHP 4 >= 4.0RC2)

Retourne la hauteur du A majuscule, et du x minuscule.

```
array swf_getfontinfo  ()
```

swf_getfontinfo() retourne la hauteur du A majuscule, et du x minuscule, dans un tableau associatif :

- Aheight - La hauteur du A majuscule, en pixels.
- xheight - La hauteur du x minuscule, en pixels.

swf_definetext (PHP 4 >= 4.0RC2)

Définit une chaîne de texte.

```
void swf_definetext  (int objid, string str, int docenter)
```

swf_definetext() définit la chaîne de texte *str*, en utilisant la police courante. *docenter* indique si la chaîne doit être centrée (valeur de 1), ou pas.

swf_textwidth (PHP 4 >= 4.0RC2)

Retourne la longueur d'une chaîne.

```
float swf_textwidth  (string str)
```

swf_textwidth() retourne la longueur de la chaîne *str*, en pixels, en utilisant la police courante.

swf_definebitmap (PHP 4 >= 4.0RC2)

Définit une image bitmap.

```
void swf_definebitmap  (int objid, string image_name)
```

swf_definebitmap() définit une bitmap à partir d'une image au format GIF, JPEG, RGB ou FI. L'image sera convertie en Flash JPEG ou Flash color map.

swf_getbitmapinfo (PHP 4 >= 4.0RC2)

Lit les informations sur une image.

```
array swf_getbitmapinfo  (int bitmapid)
```

swf_getbitmapinfo() retourne un tableau d'informations sur l'image bitmap repérée par *bitmapid*. Le tableau a les éléments suivants :

- "size" - La taille en octets de l'image.
- "width" - La largeur en pixels de l'image.
- "height" - La hauteur en pixels de l'image.

swf_startsymbol (PHP 4 >= 4.0RC2)

Définit un symbole.

```
void swf_startsymbol  (int objid)
```

swf_startsymbol() définit un identifiant d'objet comme symbole. Les symboles sont des petites animations Flash qui peuvent être jouées simultanément. *objid* est l'identifiant d'objet que vous voulez définir comme symbole.

swf_endsymbol (PHP 4 >= 4.0RC2)

Termine la définition de symbole.

```
void swf_endsymbol  ()
```

swf_endsymbol() termine la définition de symble, qui a été commencée avec **swf_startsymbol()**.

swf_startbutton (PHP 4 >= 4.0RC2)

Commence la définition d'un bouton.

```
void swf_startbutton  (int objid, int type)
```

swf_startbutton() commence la définition d'un bouton. *type* peut prendre les valeurs de TYPE_MENUBUTTON ou TYPE_PUSHBUTTON. La constante TYPE_MENUBUTTON permet au focus de traverser lorsque la souris est cliquée, alors que TYPE_PUSHBUTTON ne le permet pas.

swf_addbuttonrecord (PHP 4 >= 4.0RC2)

Contrôle la situation, l'aparance et la zone active du bouton courant.

```
void swf_addbuttonrecord  (int states, int shapeid, int depth)
```

swf_addbuttonrecord() permet de modifier les caractéristiques d'un bouton. *states*, définit les états du bouton autorisés : ce peut être : BSHitTest, BSDown, BSOver ou BSUp. *shapeid* est l'apparance du bouton, c'est à dire l'objet qui représente le bouton. *depth* est la profondeur de placement du bouton, dans le frame courant.

Exemple 1. Exemple avec swf_addbuttonrecord()

```
swf_startButton ($objid, TYPE_MENUBUTTON);
    swf_addButtonRecord (BSDown|BSOver, $buttonImageId, 340);
    swf_onCondition (MenuEnter);
        swf_actionGetUrl ("http://www.designmultimedia.com", "_level1");
    swf_onCondition (MenuExit);
        swf_actionGetUrl ("", "_level1");
swf_endButton ();
```

swf_oncondition (PHP 4 >= 4.0RC2)

Décrit une transition utilisée pour déclencher une liste d'actions.

```
void swf_oncondition  (int transition)
```

swf_oncondition() décrit une transition qui va déclencher une liste d'actions. Il y a plusieurs types de transition possibles, les suivantes sont destinées aux boutons de type TYPE_MENUBUTTON:

- IdletoOverUp
- OverUptoIdle
- OverUptoOverDown
- OverDowntoOverUp
- IdletoOverDown
- OutDowntoIdle
- MenuEnter (IdletoOverUp|IdletoOverDown)
- MenuExit (OverUptoIdle|OverDowntoIdle)

Pour les types TYPE_PUSHBUTTON voici les options :

- IdletoOverUp

- OverUptoIdle
- OverUptoOverDown
- OverDowntoOverUp
- OverDowntoOutDown
- OutDowntoOverDown
- OutDowntoIdle
- ButtonEnter (IdletoOverUp|OutDowntoOverDown)
- ButtonExit (OverUptoIdle|OverDowntoOutDown)

swf_endbutton (PHP 4 >= 4.0RC2)

Termine la définition du bouton courant.

```
void swf_endbutton  ()
```

swf_endbutton() termine la définition du bouton courant.

swf_viewport (PHP 4 >= 4.0RC2)

Sélectionne une nouvelle zone pour un dessin ultérieur.

```
void swf_viewport  (double xmin, double xmax, double ymin, double ymax)
```

swf_viewport() sélectionne une nouvelle zone pour y dessiner ultérieurement. La zone est définie de *xmin* à *xmax* et de *ymin* à *ymax*. Si cette fonction n'est pas appelée, les valeurs par défaut sont celles de l'écran courant.

swf_ortho (PHP 4 >= 4.0.1)

Définit une projection orthogonale entre les coordonnées utilisateur et le port courant.

```
void swf_ortho  (double xmin, double xmax, double ymin, double ymax, double zmin, double
zmax)
```

swf_ortho() définit une projection orthogonale entre les coordonnées utilisateur et le port courant.

swf_ortho2 (PHP 4 >= 4.0RC2)

Définit une projection orthogonale à 2 dimensions entre les coordonnées utilisateur et le port courant.

```
void swf_ortho2  (double xmin, double xmax, double ymin, double ymax)
```

swf_ortho2() définit une projection orthogonale à 2 dimensions entre les coordonnées utilisateur et le port courant. C'est la projection par défaut des animations Flash. Si vous souhaitez une perspective, utilisez plutôt **swf_perspective()**.

swf_perspective (PHP 4 >= 4.0RC2)

Définit une projection orthogonale à 3 dimensions entre les coordonnées utilisateur et le port courant

```
void swf_perspective  (double fovy, double aspect, double near, double far)
```

swf_perspective() définit une projection orthogonale à 3 dimensions entre les coordonnées utilisateur et le port courant. Le paramètre *fovy* est l'angle de vue de la direction y. Le paramètre *aspect* doit être choisi pour correspondre au ratio de la vue utilisée. *near* est le plan adjacent proche *far* est le plan adjacent distant.

> **Note :** Diverses distortions peuvent apparaître lors de ce genre de projection, car Flash ne dispose que d'une matrice à 2 dimensions. Certaines distortions font vraiment tâche d'encre.

swf_polarview (PHP 4 >= 4.0RC2)

Défini le point de vue de l'utilisateur en coordonnées polaire.

```
void swf_polarview  (double dist, double azimuth, double incidence, double twist)
```

swf_polarview() définit la position de l'utilisateur en coordonnées polaires. *dist* est la distance entre le point de vue et l'origine. *azimuth* définit l'angle azimutal dans le plan x,y mesuré en distance depuis l'axe y. *incidence* définit l'angle d'incidence dans le plan y,z, mesuré en distance depuis l'axe z. Finalement, *twist* est l'angle de rotation du point de vue sur la ligne de vue, en utilisant la règle de la main droite.

swf_lookat (PHP 4 >= 4.0RC2)

Définit une transformation de vue.

```
void swf_lookat  (double view_x, double view_y, double view_z, double reference_x, double
reference_y, double reference_z, double twist)
```

swf_lookat() définit une transformation de vue, en donnant la position de la vue, de coordonnées (*view_x*, *view_y* et *view_z*) et les coordonnées du point de référence dans la scène, de coordonnées (*reference_x*, *reference_y*, *reference_z*). Le paramètre *twist* contrôle la rotation le long de l'axe des z de l'utilisateur.

swf_pushmatrix (PHP 4 >= 4.0RC2)

Empile la matrice de transformation courante dans la pile.

```
void swf_pushmatrix  ()
```

swf_pushmatrix() empile la matrice de transformation courante dans la pile.

swf_popmatrix (PHP 4 >= 4.0RC2)

Dépile la matrice de transformation.

```
void swf_popmatrix  ()
```

swf_popmatrix() dépile la matrice de transformation.

swf_scale (PHP 4 >= 4.0RC2)

Homothétie.

```
void swf_scale  (double x, double y, double z)
```

swf_scale() fait une mise à l'échelle de *x* pour les coordonnées x, de *y* pour les coordonnées y et *z* pour les coordonnées z.

swf_translate (PHP 4 >= 4.0RC2)

Translate la transformation courante.

```
void swf_translate  (double x, double y, double z)
```

swf_translate() déplace la transformation courante de *x*, *y* et *z*, dans les directions x, y et z.

swf_rotate (PHP 4 >= 4.0RC2)

Rotation de la transformation courante.

```
void swf_rotate  (double angle, string axis)
```

swf_rotate() fait subir la roation d'angle *angle*, autour de l'axe *axis*. Les valeurs possibles pour *axis* sont : 'x' (axe x), 'y' (axe y) ou 'z' (axe z).

swf_posround (PHP 4 >= 4.0RC2)

Active l'approximation des translation d'objets.

```
void swf_posround  (int round)
```

swf_posround() active ou désactive l'approximation lors des translations, lorsque des objets sont placés ou déplacés. Il y a des situations oú le texte devient plus lisible lorsque l'approximation a été activée. *round* active l'approximation (1) ou la désactive (0).

LXXXI. SNMP

Afin de pouvoir utiliser les fonctions SNMP sous Unix, vous aurez besoin d’installer le package UCD SNMP (http://ucd-snmp.ucdavis.edu/). Sous Windows ces fonctions ne sont disponibles que sous NT, et pas sous Win95/98.

Important : Afin d’utiliser le package UCD SNMP, vous devez mettre la variable NO_ZEROLENGTH_COMMUNITY à 1 avant de compiler. Après avoir configuré UCD SNMP, éditez le fichier config.h et recherchez la valeur NO_ZEROLENGTH_COMMUNITY. Décommentez la ligne avec le #define. Cela doit ressembler à ceci :

```
#define NO_ZEROLENGTH_COMMUNITY 1
```

Si vous avez des erreurs "segmentation faults", lors de l’utilisation des commandes SNMP, c’est que vous n’avez pas suivi les recommendations précédentes. Si vous ne voulez pas recompiler UCD SNMP, vous pouvez aussi recompiler PHP avec l’option --enable-ucd-snmp-hack qui évitera cette erreur.

snmpget (PHP 3, PHP 4 >= 4.0b1)

Reçoit un objet SNMP.

```
string snmpget (string hostname, string community, string object_id [, int timeout [, int
retries]])
```

snmpget() retourne un objet SNMP en cas de succès, et FALSE en cas d'erreur.

snmpget() sert à lire une valeur d'un objet SNMP représenté par *object_id*. L'agent SNMP est défini par *hostname* et la communauté de lecture est spécifiée par le paramètre *community*.

$syscontact = snmpget("127.0.0.1", "public", "system.SysContact.0")

snmpset (PHP 3>= 3.0.12, PHP 4)

Envoie un objet SNMP.

```
bool snmpset (string hostname, string community, string object_id, string type, mixed
value [, int timeout [, int retries]])
```

snmpset() modifie la valeur de l'objet SNMP spécifié, en retournant TRUE en cas de succès et FALSE en cas d'erreur.

snmpget() sert à affecter une valeur donnée à un objet SNMP, référencé par *object_id*. L'agent SNMP est défini par *hostname* et la communauté de lecture est spécifiée par le paramètre *community*.

snmpwalk (PHP 3, PHP 4 >= 4.0b1)

Reçoit tous les objets SNMP d'un agent.

```
array snmpwalk (string hostname, string community, string object_id [, int timeout [, int
retries]])
```

snmpwalk() retourne un tableau d'objets SNMP, en commencant à partir de *object_id* comme racine, ou FALSE en cas d'erreur.

snmpwalk() sert à lire toutes les valeurs d'un agent SNMP, défini par *hostname*. *community* définit la communauté de lecture de l'agent. Un objet (*object_id* = NULL) sert de racine à l'arbre d'objet SNMP et tous les objets sous cette racine sont retournés dans un tableau. Si *object_id* est spécifié, tous les objets SNMP sous cet objet sont retournés.

```
<?php
$a = snmpwalk("127.0.0.1", "public", "");
?>
```

La fonction ci-dessus va retourner tous les objets SNMP d'un agent SNMP qui fonctionnerait sur l'hôte local (localhost). Il suffit alors de faire une boucle pour travailler avec chacun des objets.

```
<?php
for ($i=0; $i<count($a); $i++) {
    echo $a[$i];
}
?>
```

snmpwalkoid (PHP 3>= 3.0.8, PHP 4)

Demande d'informations d'arbre sur une entité du réseau.

```
array snmpwalkoid (string hostname, string community, string object_id [, int timeout [,
int retries]])
```

snmpwalkoid() retourne un tableau associatif, avec les identifiants d'objet et les objets associés, pour tous les objets situés sous la racine *object_id*, ou FALSE en cas d'erreur.

snmpwalkoid() sert à lire tous les identifiants d'objet, et leur valeurs respectives, depuis un serveur SNMP. *community* indique la communauté de lecture pour cet agent. Un *object_id* NULL signifie qu'il faut utiliser la racine de l'arbre SNMP et tous les objets sous cet arbre seront retournés. Si *object_id* est spécifié, tous les objets SNMP situés sous cet objet seront retournés.

La fonction ci-dessous va lire tous les objets de l'agent SNMP qui fonctionne sur l'hôte local. Il est alors possible de les passer en revue avec une boucle : l'existence de **snmpwalkoid()** et **snmpwalk()** est une question d'évolution. Ces deux fonctions sont fournies pour des raisons de compatbilité ascendante.

```
<?php
$a = snmpwalkoid("127.0.0.1", "public", "");
?>
```

La fonction ci-dessous va lire tous les objets de l'agent SNMP qui fonctionne sur l'hôte local. Il est alors possible de les passer en revue avec une boucle :

```
for (reset($a); $i = key($a); next($a)) {
    echo "$i: $a[$i]<br>\n";
}
```

snmp_get_quick_print (PHP 3>= 3.0.8, PHP 4)

Lit la valeur courante de l'option quick_print de la librairie UCD.

```
boolean snmp_get_quick_print (void )
```

snmp_get_quick_print() retourne la valeur courante, stockée dans la librairie UCD, de l'option quick_print. Par défaut, quick_print est inactivée.

```
<?php
$quickprint = snmp_get_quick_print();
?>
```

L'exemple ci-dessus devrait retourner FALSE, si quick_print est inactivée et, TRUE si quick_print est activée.

snmp_get_quick_print() est seulement disponible avec la librairie UCD SNMP. Cette fonction n'est pas disponible avec la librairie Windows SNMP.

Voir : **snmp_set_quick_print()** pour une description complète de l'affichage de quick_print.

snmp_set_quick_print (PHP 3>= 3.0.8, PHP 4)

Ecrit la valeur courante de l'option quick_print de la librairie UCD.

```
void snmp_set_quick_print (boolean quick_print)
```

snmp_set_quick_print() fixe la valeur de l'option quick_print de la librairie UCD SNMP. Lorsqu'elle a la valeur de (1), la librairie SNMP retournera des valeurs 'rapides'. Cela signifie que seule, la valeur sera retournée. Lorsqu'elle a la valeur de (0), la librairie va afficher d'autres informations (telles que l'adresse IP (IpAddress) ou OID). De plus, si quick_print n'est pas activée, la librairie affichera aussi des valeurs hexadécimales supplémentaires pour toutes les chaînes de trois caractères, ou moins.

Modifier quick_print est plus fréquent lorsqu'on utilise les valeurs retournées que lorsqu'on les affiche.

```
snmp_set_quick_print(0);
$a = snmpget("127.0.0.1", "public", ".1.3.6.1.2.1.2.2.1.9.1");
echo "$a<BR>\n";
snmp_set_quick_print(1);
$a = snmpget("127.0.0.1", "public", ".1.3.6.1.2.1.2.2.1.9.1");
echo "$a<BR>\n";
```

La première valeur affichée sera : 'Timeticks: (0) 0:00:00.00', tandis qu'avec quick_print activée, seul '0:00:00.00' sera affiché.

Par défaut, UCD SNMP retourne des valeurs détaillées, et quick_print sert à ne retourner que la valeur.

Actuellement, les chaînes sont toujours retournées avec des guillemets supplémentaires. Ceci sera corrigé ultérieurement.

snmp_set_quick_print() ne fonctionne qu'avec la librairie UCD SNMP. **snmp_set_quick_print()** n'est pas disponible avec la librairie Windows SNMP.

LXXXII. Sockets

> **Avertissement**
>
> Ce module est *EXPERIMENTAL*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

L'extension socket implémente une interface bas niveau avec les fonctions de communication par socket. Cela permet de mettre en place un serveur aussi bien qu'un client.

Les fonctions socket décrites ici sont rassemblées dans une extension PHP. Pour être activées, il faut utiliser l'option de compilation `--enable-sockets` au script **configure**.

Pour une interface client plus générique, reportez vous à **fsockopen()** et **pfsockopen()**.

Lorsque vous utiliserez les fonctions de sockets qui sont décrites ici, gardez bien à l'esprit que même si elles ont souvent des noms identiques aux fonctions C, elles ont souvent des prototypes différents. Lisez attentivement la documentation pour éviter les confusions.

Cela dit, ceux qui n'ont pas l'habitude de la programmation avec les sockets pourront trouver beaucoup de documentation pertinente dans les pages de manuel Unix, et de nombreux tutorial de programmation C sur le web, dont la plus part peuvent être repris après de légère modifications, en PHP.

Exemple 1. Exemple de programmation Socket : serveur TCP/IP

Cet exemple est un serveur perroquete : tout ce que vous lui envoyez vous est retourné. Changez les variables `address` et `port` pour les adapter à votre configuration, et lancez le script. Vous pouvez vous connecter au serveur avec une commande telle que **telnet 192.168.1.53 10000** (avec l'adresse et le port qui sont ceux de votre configuration). Pour vous déconnecter, tapez 'quit'.

```
<?php
error_reporting(E_ALL);
/* On autorise le script à attendre les connexions indéfiniment. */
set_time_limit(0);
/* Modifiez ces valeurs pour qu'elles soient celles de votre configuration */
$address = '192.168.1.53';
$port = 10000;
if (($sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
    echo "socket() a échoué : raison : " . strerror($sock) . "\n";
}
if (($ret = bind($sock, $address, $port)) < 0) {
    echo "bind() a échoué : raison: " . strerror($ret) . "\n";
}
if (($ret = listen($sock, 5)) < 0) {
    echo "listen() a échoué : raison: " . strerror($ret) . "\n";
}
do {
    if (($msgsock = accept_connect($sock)) < 0) {
        echo "accept_connect() a échoué : raison : " . strerror($msgsock) . "\n";
        break;
    }
    do {
        $buf = '';
        $ret = read($msgsock, $buf, 2048);
        if ($ret < 0) {
            echo "read() a échoué : raison : " . strerror($ret) . "\n";
            break 2;
        }
        if ($ret == 0) {
            break 2;
        }
        $buf = trim($buf);
        if ($buf == 'quit') {
            close($msgsock);
            break 2;
```

```
        }
        $talkback = "PHP: Vous avez dit ’$buf’.\n";
        write($msgsock, $talkback, strlen($talkback));
        echo "$buf\n";
    } while (TRUE);
    close($msgsock);
} while (TRUE);
close($sock);
?>
```

Exemple 2. Exemple avec les sockets : Client TCP/IP

Cet exemple est un client HTTP basique. Il se connecte à une page envoi les entêtes (requête HEAD), affiche le retour, et quitte.

```
<?php
error_reporting(E_ALL);
echo "<h2>TCP/IP Connection</h2>\n";
/* Demande le port du service WWW. */
$service_port = getservbyname(’www’, ’tcp’);
/* Demande l’IP du serveur de destination. */
$address = gethostbyname(’www.php.net’);
/* Crée la connexion TCP/IP. */
$socket = socket(AF_INET, SOCK_STREAM, 0);
if ($socket < 0) {
    echo "socket() a échoué : raison : " . strerror($socket) . "\n";
} else {
    "socket() réussi: " . strerror($socket) . "\n";
}
echo "Connexion à ’$address’ on port ’$service_port’...";
$result = connect($socket, $address, $service_port);
if ($result < 0) {
    echo "connect()  a échoué : raison : : ($result) " . strerror($result) . "\n";
} else {
    echo "OK.\n";
}
$in = "HEAD / HTTP/1.0\r\n\r\n";
$out = ”;
echo "Envoi des entêtes HTTP HEAD...";
write($socket, $in, strlen($in));
echo "OK.\n";
echo "Lecture de la réponse :\n\n";
while (read($socket, $out, 2048)) {
    echo $out;
}
echo "Fermeture de la socket...";
close($socket);
echo "OK.\n\n";
?>
```

accept_connect (4.0.2 - 4.0.6 only)

Accepte une connexion sur une socket.

```
int accept_connect (resource socket)
```

Une fois que la socket *socket* a été créé, avec la fonction **socket()**, liée à un nom avec **bind()**, et mise en attente de connexion avec **listen()**, **accept_connect()** accepte les connexions sur la socket *socket*. Une fois que la connexion est faite, un nouveau pointeur de socket est retourné, pour utilisation ultérieure. Si il n'y a pas de connexion en attente, **accept_connect()** se bloquera jusqu'à une connexion soit disponible. Si *socket* a été configurée comme non-bloquante, avec **socket_set_blocking()**, une erreur sera retournée.

Le pointeur de socket retourné par **accept_connect()** ne peut plus accepter de nouvelles connexion. La socket originale, *socket*, reste ouverte, et peut être réutilisée.

accept_connect() retourne une nouveau pointeur de socket en cas de succès, ou une erreur négative en cas d'erreur. Ce code peut être passé à la fonction **strerror()** pour obtenir un message d'erreur lisible.

Voir aussi **bind()**, **connect()**, **listen()**, **socket()** et **strerror()**.

bind (4.0.2 - 4.0.6 only)

Lie un nom à une socket.

```
int bind (resource socket, string address [, int protocol])
```

bind() lie le nom *address*, à la socket *socket*, qui doit être une socket valide, créée avec **socket()**.

address peut être une adresse IP numérique (e.g. `127.0.0.1`), si la socket est de la famille `AF_INET`; ou bien un chemin d'un domaine UNIX, si la socket est de la famille des `AF_UNIX`.

port sert uniquement dans le cas des sockets de type `AF_INET` et désigne le port de connexion sur l'hôte distant.

bind() retourne zéro en cas de succès, et une erreur négative en cas d'échec. Ce code peut être passé à la fonction **strerror()** pour obtenir un message d'erreur lisible.

Voir aussi **accept_connect()**, **connect()**, **listen()**, **socket()**, et **strerror()**.

close (4.0.2 - 4.0.6 only)

Ferme une socket.

```
bool close (resource socket)
```

close() ferme le fichier (ou la socket) *socket*.

Notez que **close()** ne doit pas être utilisée avec des pointeurs de fichiers créé par **fopen()**, **popen()**, **fsockopen()**, ou **pfsockopen()**; elle ne sert que pour les sockets créées avec **socket()** ou **accept_connect()**.

close() retourne `TRUE` en cas de succès, ou `FALSE` en cas d'erreur(i.e., *socket* est invalide).

Voir aussi **bind()**, **listen()**, **socket()** et **strerror()**.

connect (4.0.2 - 4.0.6 only)

Initie une connexion avec une socket.

```
int connect (resource socket, string address [, int port])
```

connect() initie une connexion avec la s ocket *socket*, qui doit être une socket valide, créée avec **socket()**.

address peut être une adresse IP numérique (e.g. 127.0.0.1), si la socket est de la famille AF_INET; ou bien un chemin d'un domaine UNIX, si la socket est de la famille des AF_UNIX.

port sert uniquement dans le cas des sockets de type AF_INET et désigne le port de connexion sur l'hôte distant.

connect() retourne zéro en cas de succès, et une erreur négative en cas d'échec. Ce code peut être passé à la fonction **strerror()** pour obtenir un message d'erreur lisible.

Voir aussi **bind()**, **listen()**, **socket()**, et **strerror()**.

listen (4.0.2 - 4.0.6 only)

Attend une connexion sur une socket.

```
int listen (resource socket, int backlog)
```

Une fois que la socket *socket* a été créée avec **socket()** et liée avec **bind()**, elle peut être mise en attente de connexion entrante. Un maximum de *backlog* connexion entrantes seront mises en attente de traitement.

listen() ne fonctionne qu'avec des sockets de type SOCK_STREAM et SOCK_SEQPACKET.

listen() retourne zéro en cas de succès, et une erreur négative en cas d'échec. Ce code peut être passé à la fonction **strerror()** pour obtenir un message d'erreur lisible.

Voir aussi **accept_connect()**, **bind()**, **connect()**, **socket()** et **strerror()**.

read (4.0.2 - 4.0.6 only)

Lit sur une socket

```
int read (resource socket_des, string &buffer, int length)
```

read() lit sur la socket *socket_des* créée avec **accept_connect()**, et place le résultat dans le buffer *&buffer*, *length* octets. Vous pouvez aussi utiliser \n, \t ou \0 pour terminer la lecture. Le nombre d'octets lus est retourné.

Voir aussi **accept_connect()**, **bind()**, **connect()**, **listen()**, **strerror()**, **socket_get_status()** et **write()**.

socket (4.0.2 - 4.0.6 only)

Crée une socket (point de communication).

```
resource socket (int domain, int type, int protocol)
```

socket() crée un point de communication, appelé socket, et retourne un pointeur de socket.

domain représente le domaine. Actuellement, ce peut être AF_INET et AF_UNIX.

type sélectionne le type de socket. Il peut prendre les valeurs suivantes : SOCK_STREAM, SOCK_DGRAM, SOCK_SEQPACKET, SOCK_RAW, SOCK_RDM, ou SOCK_PACKET.

protocol choisit le protocole.

Retourne un pointeur de socket valide en cas de succès, et une erreur négative en cas d'échec. Ce code peut être passé à la fonction **strerror()** pour obtenir un message d'erreur lisible.

Pour plus d'informations sur l'utilisation des **socket()**, ainsi que sur la pluspart des paramètres, reportez vous aux pages de manuel Unix socket (2).

Voir aussi **accept_connect()**, **bind()**, **connect()**, **listen()** et **strerror()**.

strerror (4.0.2 - 4.0.6 only)

Décrit une erreur de socket.

```
string strerror (int errno)
```

strerror() prend comme paramètre *errno* la valeur négative de retour d'une fonction de socket, et retourne l'explication correspondante au format texte. Cela facilite grandement la recherche d'erreur. Par exemple, au lieu d'être bloqué par une erreur '-111', et de devoir en rechercher la signification dans les fichiers systèmes, il suffit de la passer à **strerror()**, pour savoir ce qui s'est passé.

Exemple 1. Exemple avec strerror()

```
<?php
if (($socket = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
   echo "socket() a échoué : raison: " . strerror($socket) . "\n";
}
if (($ret = bind($socket, '127.0.0.1', 80)) < 0) {
   echo "bind() a échoué : raison: " . strerror($ret) . "\n";
}
?>
```

Le résultat de l'exemple ci dessus (en supposant que le script n'est pas exécuté avec les droits du root) :

```
        bind() a échoué : raison : Permission denied
```

Voir aussi **accept_connect()**, **bind()**, **connect()**, **listen()** et **socket()**.

write (4.0.2 - 4.0.6 only)

Ecrit sur une socket

```
int write (resource socket_des, string &buffer, int length)
```

write() écrit sur la socket *socket_des*, *length* octets issus du buffer *&buffer*.

Voir aussi **accept_connect()**, **bind()**, **connect()**, **listen()**, **read()**, **strerror()** et **socket_get_status()**.

LXXXIII. Chaîne de caractères

Ces fonctions permettent la manipulation de chaînes de caractères. Certaines sections plus spécialisées sont disponibles dès les sections sur les expressions régulières et dans la section URL.

Pour plus de détails sur le comportement des chaînes de caractères, notamment concernant les guillemets simples ou doubles, et les séquences d’échappement, reportez-vous à chaînes de caractères, dans le chapitre Types.

AddCSlashes (PHP 4 >= 4.0b4)

Ajoute des slash dans une chaîne, comme en langage C.

```
string addcslashes (string str, string charlist)
```

addcslashes() retourne une chaîne avec des antislash devant les caractères qui sont dans la liste *charlist*. Les caractères \n, \r etc... sont échappés. En langage C, les caractères avec un code ASCII inférieur à 32 ou supérieur à 126 sont convertis en représentation octale. Faites bien attention lorsque vous échappez des caractères alpha-numériques. Vous pouvez spécifier un intervalle dans *charlist* comme "\0..\37", qui échappera les caractères compris dans cet intervalle.

Exemple 1. Exemple avec addcslashes()

```
<?php
$escaped = addcslashes($no_echappe, "\0..\37!@\177..\377");
?>
```

> **Note : addcslashes()** a été ajouté en PHP 4.0b3-dev.

Voir aussi **stripcslashes()**, **stripslashes()**, **htmlspecialchars()** et **quotemeta()**.

AddSlashes (PHP 3, PHP 4 >= 4.0b1)

Ajoute un slash devant tous les caractères spéciaux.

```
string addslashes (string str)
```

addslashes() retourne une chaîne avec des antislash devant chaque caractère qui a en a besoin pour être inséré dans une requête de base de données. Ces caractères sont guillemets simples (’), guillemets doubles ("), antislash (\) et NULL (la valeur nulle).

Voir aussi **stripslashes()**, **htmlspecialchars()** et **quotemeta()**.

bin2hex (PHP 3>= 3.0.9, PHP 4 >= 4.0b1)

Convertit une valeur binaire en hexadécimale

```
string bin2hex (string str)
```

bin2hex() retourne une chaîne ASCII contenant la représentation hexadécimale de *str*. La conversion est faite avec le bit de poids fort en premier.

chop (PHP 3, PHP 4 >= 4.0b1)

Enlève les espaces de fin de chaîne.

```
string chop (string str)
```

chop() retourne l’argument sans les espaces de fin de chaîne.

Exemple 1. Exemple avec chop()

```
<?php
  $trimmed = chop($line);
?>
```

Note : chop() diffère de sa cousine Perl `chop()`, qui supprime le dernier caractère de la chaîne.

Voir aussi **trim()**, **ltrim()**, **rtrim()** et **chop()**.

chr (PHP 3, PHP 4 >= 4.0b1)

Retourne un caractère.

```
string chr (int ascii)
```

chr() retourne le caractère de code ASCII *ascii*.

Exemple 1. Exemple avec chr()

```
<?php
$str .= chr(27); /* ajoute un échappement à la fin de la chaîne $str */
/* Généralement, ceci est plus efficace  */
$str = sprintf("Cette chaîne se termine par un échappement : %c", 27);
?>
```

chr() est le contraire de **ord()**.

Voir aussi **sprintf()** avec le format de chaîne %c.

chunk_split (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Scinde une chaîne en plus petits morceaux.

```
string chunk_split (string string [, int chunklen [, string end]])
```

chunk_split() permet de scinder une chaîne en plus petit morceaux, comme dans le cas de la conversion en base64_encode pour se conformer à la RFC 2045. **chunk_split()** insère une fin de chaîne *end* (par défaut "\r\n"), tous les *chunklen* (par défaut 76) caractères. La chaîne retournée est une nouvelle chaîne, et l'original n'est pas modifié.

Exemple 1. Exemple avec chunk_split()

```
<?php
# formate $data avec la sémantique RFC 2045
$new_string = chunk_split(base64_encode($data));
?>
```

chunk_split() est nettement plus rapide que **ereg_replace()**.

Note : chunk_split() a été ajoutée en 3.0.6.

convert_cyr_string (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Convertit la chaîne d'un alphabet cyrillique vers un autre.

```
string convert_cyr_string (string str, string from, string to)
```

convert_cyr_string() convertit la chaîne donnée depuis un alphabet cyrillique vers un autre. Les arguments *from* et *to* sont des caractères qui représentent la source et la destination. Les valeurs acceptées :

- k - koi8-r
- w - windows-1251
- i - iso8859-5
- a - x-cp866
- d - x-cp866
- m - x-mac-cyrillic

count_chars (PHP 4 >= 4.0b4)

Retourne des informations sur les caractères utilisés dans une chaîne.

```
mixed count_chars (string string [, int mode])
```

count_chars() compte le nombre d'occurrences de chaque octet (0..255) dans la chaîne *string* et le retourne de différentes façons. L'option *mode* prend, par défaut, la valeur 0. Suivant le *mode*, **count_chars()** retourne une des réponses suivantes :

- 0 - Un tableau avec l'octet comme clé, et la fréquence comme valeur.
- 1 - Identique à 0, mais seules les fréquences non nulles sont listées.
- 2 - Identique à 0, mais seules les fréquences nulles sont listées.
- 3 - Une chaîne qui contient tous les octets utilisés.
- 4 - Une chaîne contenant tous les octets non utilisés.

Note : count_chars() a été ajoutée en PHP 4.0.

crc32 (PHP 4 >= 4.0.1)

Calcule le polynôme crc32 d'une chaîne

```
int crc32 (string str)
```

crc32() génère la somme de vérification de redondances cycliques (32 bits) de la chaîne *str*. Cette valeur sert généralement à vérifier l'intégrité de données transmises.

Voir aussi **md5()**.

crypt (PHP 3, PHP 4 >= 4.0b1)

Chiffre une chaîne avec un DES.

```
string crypt (string str [, string salt])
```

crypt() va coder une chaîne en utilisant la méthode de chiffrage du DES standard. Les arguments sont : la chaîne à chiffrer, et un grain de sel qui servira de base pour le chiffrement. Reportez-vous au manuel Unix pour plus de détails.

Si le grain de sel n'est pas fourni, il sera automatiquement généré par PHP.

Certains systèmes d'exploitation acceptent plus d'un type de chiffrement. En fait, le DES standard est parfois remplacé par un chiffrement MD5. Le type de chiffrement est alors choisi en fonction du grain de sel. A l'installation, PHP détermine les possibilités de cryptage et décidera d'accepter d'autres grains de sel pour d'autres types de chiffrement. Si le grain de sel n'est pas fourni, PHP génèrera alors un grain de 2 caractères, pour le DES standard, à moins que le système ne dispose de MD5 : dans ce cas, PHP génèrera un grain de sel pour MD, par défaut. PHP affecte la variable d'environnement CRYPT_SALT_LENGTH, à 2 s'il utilise le DES standard, et à 12 s'il utilise le MD5.

Si vous utilisez le grain de sel fourni, retenez bien que ce grain de sel est généré une seule fois. Si vous appelez **crypt()** récursivement, cela aura un impact sur l'apparence et finalement la sécurité de votre cryptage.

Le chiffrement standard fournit le grain de sel dans les deux premiers octets du résultat de la fonction **crypt()**.

Sur les systèmes qui supportent plusieurs méthodes de chiffrement, les variables d'environnement suivantes sont mises à 0 ou à 1, en fonction de la disponibilité de la méthode :

- CRYPT_STD_DES - DES Standard avec 2-octets de SALT
- CRYPT_EXT_DES - DES étendu avec 9-octets SALT
- CRYPT_MD5 - MD5 avec 12-octets SALT commençant à 1
- CRYPT_BLOWFISH - DES étendu avec 16-octets SALT commençant à 2

Il n'y a pas d'algorithme de décryptage, étant donné que **crypt()** est injective.

echo (unknown)

Affiche une ou plusieurs chaînes.

```
echo (string arg1, string [argn]...)
```

echo() affiche tous les paramètres.

echo() n'est pas une fonction à proprement parler, ce qui rend l'usage des parenthèses facultatifs. En fait, si vous voulez passer plus d'un paramètre, vous ne devez pas utiliser les parenthèses.

Exemple 1. Exemple avec echo()

```
<?php
echo "Bonjour Monde";
echo "Cet echo() se
répartit sur plusieurs lignes. Les nouvelles lignes
seront aussi affichées";
echo "Cet echo() se\nrépartit sur plusieurs lignes. Les nou-
velles lignes\nseront aussi affichées.";
echo "L'échappement de caractères est fait : \"comme ceci\".";
//Vous pouvez utiliser des variables avec echo
$foo = "foobar";
$bar = "barbaz";
echo "foo vaut &quot;$foo&quot;";
// foo vaut "foobar"
// Les guillemets simples évitent le remplacement des variables
echo 'foo is $foo'; // foo vaut $foo
```

```
// Si vous n'utilisez pas d'autres caractères,
// vous ne ferez qu'afficher une variable
echo $foo;          // foobar
echo $foo,$bar;     // foobarbarbaz
// comme echo() n'est pas une fonction, le code suivant est invalide
($some_var) ? echo('Oui'): echo('Non');
// Cependant, les lignes suivantes sont valides :
($some_var) ? print('Oui'): print('Non'); // print est une fonction
echo ($some_var) ? 'Oui': 'Non';
?>
```

Voir aussi : **print()**, **printf()** et **flush()**.

explode (PHP 3, PHP 4 >= 4.0b1)

Scinde une chaîne en morceaux, grâce à un délimiteur.

```
array explode (string separator, string string [, int limit])
```

explode() retourne un tableau qui contient les éléments de la chaîne *`string`*, séparés par *`separator`*. Si *`limit`* est fourni, le tableau retourné contiendra un maximum de *`limit`* éléments, et le dernier éléments contiendra le reste de la chaîne *`string`*. Si une chaîne vide est utilisée comme *`separator`*, alors **explode()** retournera FALSE. So *`separator`* contient une valeur qui n'est pas dans *`string`*, Alors **explode()** retournera la chaîne *`string`*.

Exemple 1. Exemple avec explode()

```
<?php
$pizza = "piece1 piece2 piece3 piece4 piece5 piece6";
$pieces = explode(" ", $pizza);
?>
```

Note : Le paramètre *`limit`* a été ajouté en PHP 4.0.1.

Note : Bien que **implode()** accepte, pour des raisons historiques, les arguments dans un sens ou l'autre, **explode()**, lui, ne le peut pas. Vous devez vous assurer que l'argument séparateur *`separator`* arrive avant l'argument de chaîne.

Voir aussi **preg_split()**, **spliti()**, **split()** et **implode()**.

get_html_translation_table (PHP 4 >= 4.0b4)

Retourne la table de traduction HTML

```
string get_html_translation_table  (int table [, int quote_style])
```

get_html_translation_table() retourne la table de traduction utilisée en interne par **htmlspecialchars()** et **htmlentities()**. Il y a deux nouvelles définitions : (*`html_entities`*, *`html_specialchars`*) qui vous permettent de spécifier vos propres tables.

Exemple 1. Exemple de table de traduction

```
<?php
$trans = get_html_translation_table(HTML_ENTITIES);
$str = "Hallo & <Frau> & Krämer";
$encoded = strtr($str, $trans);
?>
```

La variable `$encoded` va contenir désormais : "Hallo & <Frau> & Krämer".

array_flip() est alors très efficace pour inverser la direction de traduction :

```
<?php
$trans = array_flip($trans);
$original = strtr($str, $trans);
?>
```

Le contenu de `$original` sera : "`Hallo & <Frau> & Krämer`".

> **Note : get_html_translation_table()** a été ajoutée en PHP 4.0.

Voir aussi **htmlspecialchars()**, **htmlentities()**, **strtr()** et **array_flip()**.

get_meta_tags (PHP 3>= 3.0.4, PHP 4 >= 4.0b1)

Extrait toutes les balises meta d'un fichier

```
array get_meta_tags (string filename [, int use_include_path])
```

get_meta_tags() ouvre le fichier *`filename`* et l'analyse ligne par ligne, en recherchant les balises <meta>.

Exemple 1. Exemple avec les balises méta

```
<?php
<meta name="author" content="name">
<meta name="tags" content="php3 documentation">
</head> <!- parsing stops here ->
?>
```

(Faites bien attention aux fins de ligne. PHP utilise une fonction native pour analyser le fichier d'entrée, ce qui fait que les fichiers créés sous MacOS ne fonctionneront pas sous Unix).

Le nom d'une propriété devient sa clé, et la valeur devient la valeur dans le tableau associatif retourné, ce qui rend aisé la manipulation des informations. Les caractères spéciaux dans le nom de la propriété sont remplacés par des '_', le reste est converti en minuscules.

Mettre *`use_include_path`* à 1 forcera PHP à ouvrir les fichiers dans le chemin standard d'inclusion.

hebrev (PHP 3, PHP 4 >= 4.0b1)

Convertit un texte hébreux logique en texte visual

```
string hebrev (string hebrew_text [, int max_chars_per_line])
```

Le paramètre optionnel *`max_chars_per_line`* indique le nombre maximum de caractères par ligne qui seront générés. La fonction essaie d'éviter les césures de mots.

Voir aussi **hebrevc()**

hebrevc (PHP 3, PHP 4 >= 4.0b1)

Convertit un texte hébreux logique en texte visuel avec les nouvelles lignes de conversion.

```
string hebrevc (string hebrew_text [, int max_chars_per_line])
```

hebrevc() est similaire à **hebrev()**, au détail près qu'elle convertit les nouvelles lignes (\n) en "
\n". Le paramètre optionnel *max_chars_per_line* indique le nombre maximum de caractères par ligne qui seront générés. La fonction essaie d'éviter les césures de mots.

Voir aussi **hebrev()**

htmlentities (PHP 3, PHP 4 >= 4.0b1)

Convertit tous les caractères spéciaux en entité HTML.

```
string htmlentities (string string [, int quote_style])
```

htmlentities() est identique à **htmlspecialchars()** en tous points, sauf que tous les caractères qui ont une entité équivalente en HTML sont remplacés par ces entités. Comme **htmlspecialchars()**, elle prend un argument optionnel qui indique ce qui doit être fait avec les guillemets simples et doubles. ENT_COMPAT (par défaut) convertira les guillemets doubles, et ignorera les guillemets simples. ENT_QUOTES convertira les deux types de guillemets et ENT_NOQUOTES les ignorera tous les deux.

Actuellement, le jeu de caractères ISO-8859-1 est utilisé. Notez que l'argument optionnel a été ajouté PHP 3.0.17 et PHP 4.0.3.

Voir aussi **htmlspecialchars()** et **nl2br()**.

htmlspecialchars (PHP 3, PHP 4 >= 4.0b1)

Convertit tous les caractères spéciaux en entité HTML.

```
string htmlspecialchars (string string [, int quote_style])
```

Certains caractères ont une valeur avec HTML, et doivent être remplacés par des balises HTML pour conserver leur valeur. **htmlspecialchars()** retourne une chaîne dont tous les caractères sensibles ont été remplacés par leur équivalent.

htmlspecialchars() est utile pour empêcher un utilisateur de fournir un texte avec un sens HTML, comme dans un livre d'or.

htmlspecialchars() est pratique pour éviter que les textes fournis par les utilisateurs contiennent des balises HTML, comme dans le cas d'un livre d'or ou d'une tribune. **htmlspecialchars()** prend un argument optionnel qui indique ce qui doit être fait avec les guillemets simples et doubles. ENT_COMPAT (par défaut) convertira les guillemets doubles, et ignorera les guillemets simples. ENT_QUOTES convertira les deux types de guillemets et ENT_NOQUOTES les ignorera tous les deux.

Actuellement, PHP remplace les valeurs suivantes :

- '&' (et commercial) devient '&'
- '"' (guillemet double) devient '"' si ENT_NOQUOTES n'est pas actif
- "' (guillemet simple) devient ''' si ENT_QUOTES est actif
- '<' (inférieur à) devient '<'
- '>' (supérieur à) devient '>'

Exemple 1. Exemple avec htmlspecialchars()

```
<?php
$new = htmlspecialchars("<a href='test'>Test</A>", ENT_QUOTES);
?>
```

Notez bien que **htmlspecialchars()** ne fait aucun autre remplacement que ceux listés ci-dessus. Pour une traduction complète de toutes les balises, reportez-vous à **htmlentities()**. Notez que l'argument optionnel a été ajouté PHP 3.0.17 et PHP 4.0.3.

Voir aussi **htmlentities()** et **nl2br()**.

implode (PHP 3, PHP 4 >= 4.0b1)

Regroupe tous les éléments d'un tableau dans une chaîne, avec une chaîne de jointure.

```
string implode (string glue, array pieces)
```

implode() retourne une chaîne constituée de tous les éléments du tableau, pris dans l'ordre, transformés en chaîne, et séparés par *glue*.

Exemple 1. Exemple avec implode()

```
<?php
$colon_separated = implode(":", $array);
?>
```

> **Note :** Pour des raisons historiques, **implode()** accepte ces arguments dans l'un ou l'autre sens. Par cohérence avec la fonction **explode()**, il est plus clair d'utiliser l'ordre des arguments tel que documenté.

Voir aussi **explode()**, **join()** et **split()**.

join (PHP 3, PHP 4 >= 4.0b1)

Regroupe tous les éléments d'un tableau dans une chaîne, avec une chaîne de jointure.

```
string join (string glue, array pieces)
```

join() est un alias de **implode()**, et lui est identique en tous points.

Voir aussi **explode()**, **implode()** et **split()**.

levenshtein (PHP 3>= 3.0.17, PHP 4 >= 4.0.1)

Calcule la distance Levenshtein entre deux chaînes

```
int levenshtein (string str1, string str2)
int levenshtein (string str1, string str2, int cost_ins, int cost_rep, int cost_del)
int levenshtein (string str1, string str2, function cost)
```

levenshtein() retourne la distance Levenshtein entre les deux chaînes *`str1`* et *`str1`* ou -1 si un des arguments excède la limite de 255 caractères.

La distance Levenshtein est définie comme le nombre minimal de caractères qu'il faut remplacer, insérer ou effacer pour transformer la chaîne *`str1`* en *`str2`*. La complexité de l'algorithme est en `O(m*n)`, où `n` et `m` sont les longueurs respectives de *`str1`* et *`str2`* (ceci est plutôt un bon résultat, comparé à **similar_text()**, qui est en O(max(n,m)**3), mais cela reste coÛteux en terme de ressources).

Dans sa forme la plus simple, la fonction va prendre uniquement deux chaînes en paramètres, et calculer uniquement le nombre d'insertions, remplacements et effacements nécessaires pour transformer la chaîne *`str1`* en *`str2`*.

Une variante prend trois paramètres additionnels, qui définissent le coÛt des insertions, des remplacements et des effacement. C'est une version plus générale et plus souple que la version simple, mais qui n'est pas aussi efficace.

La deuxième variante n'est pas encore implémentée. Elle est encore plus générale, et plus souple, mais plus lente. Elle appellera une fonction utilisateur qui déterminera le coÛt de chaque opération.

La fonction utilisateur sera appelée avec les arguments suivants :

- opération a appliquer : 'I', 'R' or 'D'
- caractère courant de la chaîne *`str1`*
- caractère courant de la chaîne *`str2`*
- position courante de la chaîne *`str1`*
- position courante de la chaîne *`str2`*
- caractères restants dans la chaîne *`str1`*
- caractères restants dans la chaîne *`str2`*

La fonction utilisateur doit retourner un entier positif, qui décrira le coÛt de cette opération particulière. Elle peut ne prendre en compte que certains arguments, et non leur totalité.

L'utilisation d'une fonction utilisateur permet de prendre en compte la différence entre certains caractères, ou leur contexte pour déterminer le coÛt d'une opération d'insertion, remplacement ou effacement. Elle accroît la charge de calcul demandée au CPU, et annule l'optimisation des autres variantes.

Voir aussi **soundex()**, **similar_text()** et **metaphone()**.

localeconv (PHP 4 >= 4.0.5)

Lit le formatage numérique et monétaire

```
array localeconv (void)
```

localeconv() retourne un tableau associatif contenant les informations locales de formats monétaire et numérique utilisés par le serveur.

localeconv() retourne les informations à partir des données locales, comme définies par **setlocale()**. Le tableau associatif retourné contient les entrées suivantes :

Index	Description
decimal_point	Séparateur décimal
thousands_sep	Séparateur de milliers
grouping	Tableau contenant les groupages numériques
int_curr_symbol	Symbole monétaire international (i.e. FRF)
currency_symbol	Symbole monétaire local (i.e. F)
mon_decimal_point	Séparateur décimal monétaire
mon_thousands_sep	Séparateur de milliers monétaires
mon_grouping	Tableau contenant les groupages numériques monétaires

Index	Description
positive_sign	Signe des valeurs positives
negative_sign	Signe des valeurs négatives
int_frac_digits	Nombre de chiffres décimaux international
frac_digits	Nombre de chiffres décimaux locaux
p_cs_precedes	TRUE si currency_symbol précède une valeur positive, FALSE s'il lui succède
p_sep_by_space	TRUE si un espace sépare currency_symbol d'une valeur positive, FALSE sinon
n_cs_precedes	TRUE si currency_symbol précède une valeur négative, FALSE s'il lui succède
n_sep_by_space	TRUE si un espace sépare currency_symbol d'une valeur négative, FALSE sinon
p_sign_posn	0 Des parenthèses entourent la quantité et currency_symbol 1 Le signe précède la quantité et currency_symbol 2 Le signe suit la quantité et currency_symbol 3 Le signe précède immédiatement currency_symbol 4 Le signe suit immédiatement currency_symbol
n_sign_posn	0 Des parenthèses entourent la quantité et currency_symbol 1 Le signe précède la quantité et currency_symbol 2 Le signe suit la quantité et currency_symbol 3 Le signe précède immédiatement currency_symbol 4 Le signe suit immédiatement currency_symbol

Le champs de groupage contient un tableau qui définit comment les chiffres doivent être regroupés. Par exemple, ce champs pour le dollar américain contient un tableau de deux éléments (3 et 3). Les éléments sont classés de gauche à droite. Si un des éléments vaut CHAR_MAX, les groupages ne sont plus effectués. Si un éléments vaut 0, la valeur du précédent doit être utilisée.

Exemple 1. Exemple avec localeconv()

```
<?php
setlocale(LC_ALL, "en_US");
$locale_info = localeconv();
echo "<PRE>\n";
echo "--------------------------------\n";
echo "  Informations monétaires pour le serveur local: \n";
echo "--------------------------------\n\n";
echo "int_curr_symbol:   {$locale_info["int_curr_symbol"]}\n";
echo "currency_symbol:   {$locale_info["currency_symbol"]}\n";
echo "mon_decimal_point: {$locale_info["mon_decimal_point"]}\n";
echo "mon_thousands_sep: {$locale_info["mon_thousands_sep"]}\n";
echo "positive_sign:     {$locale_info["positive_sign"]}\n";
echo "negative_sign:     {$locale_info["negative_sign"]}\n";
echo "int_frac_digits:   {$locale_info["int_frac_digits"]}\n";
echo "frac_digits:       {$locale_info["frac_digits"]}\n";
echo "p_cs_precedes:     {$locale_info["p_cs_precedes"]}\n";
echo "p_sep_by_space:    {$locale_info["p_sep_by_space"]}\n";
```

```
echo "n_cs_precedes:     {$locale_info["n_cs_precedes"]}\n";
echo "n_sep_by_space:    {$locale_info["n_sep_by_space"]}\n";
echo "p_sign_posn:       {$locale_info["p_sign_posn"]}\n";
echo "n_sign_posn:       {$locale_info["n_sign_posn"]}\n";
echo "</PRE>\n";
?>
```

La constante CHAR_MAX est aussi définie ci-dessus.

Note : Ajouté en PHP 4.0.5

Voir aussi **setlocale()**.

ltrim (PHP 3, PHP 4 >= 4.0b1)

Enlève les espaces de début de chaîne.

```
string ltrim (string str)
```

ltrim() enlève les caractères blancs placés au début d'une chaîne et retourne la chaîne raccourcie. Les caractères blancs sont : "\n", "\r", "\t", "\v", "\0", et " ".

Voir aussi **chop()** et **trim()**.

md5 (PHP 3, PHP 4 >= 4.0b1)

Calcule un md5 avec la chaîne.

```
string md5 (string str)
```

Crypte la chaîne *str* en utilisant la méthode MD5 (voir RSA Data Security, Inc. MD5 Message-Digest Algorithm (http://www.faqs.org/rfcs/rfc1321.html).).

metaphone (PHP 4 >= 4.0b4)

Calcule la clé métaphone d'une chaîne de caractères.

```
string metaphone (string str)
```

metaphone() calcule la clé métaphone de la chaîne *str*.

Similairement à **soundex()**, métaphone crée une clé similaire pour des sons proches. C'est une fonction plus précise que **soundex()** car elle prend en compte les règles basiques de la prononciation en anglais. Les clés métaphones sont de longueur variable.

Le métaphone a été développée par Lawrence Philips <lphilips@verity.com>. Elle est décrite dans ["Practical Algorithms for Programmers", Binstock & Rex, Addison Wesley, 1995].

Note : C**metaphone()** a été ajoutée en PHP 4.0.

nl2br (PHP 3, PHP 4 >= 4.0b1)

Insère une césure HTML avant chaque nouvelle ligne.

```
string nl2br (string string)
```

nl2br() retourne la chaîne *string* dont toutes les lignes ont été remplaçées par '<BR />'.

A partir de la version PHP 4.0.5, **nl2br()** est désormais compatible XHTML. Toutes les versions antérieures retourneront la chaîne *string* avec '<br>' remplaçant les nouvelles lignes, au lieu de '<br />'.

Voir aussi **htmlspecialchars()** et **htmlentities()**.

ord (PHP 3, PHP 4 >= 4.0b1)

Retourne la valeur ASCII du caractère.

```
int ord (string string)
```

ord() retourne la valeur ASCII du premier caractère de la chaîne *string*. **ord()** est le contraire de **chr()**.

Exemple 1. Exemple avec ord()

```
<?php
if (ord($str) == 10) {
    echo "Le premier caractère de \$str est un retour chariot.\n";
}
?>
```

Voir aussi **chr()**.

parse_str (PHP 3, PHP 4 >= 4.0b1)

Analyse une chaîne, et en déduit des variables et leur valeur.

```
void parse_str (string str [, array arr])
```

Analyse la chaîne *str* comme si c'était une chaîne passée par URL, et affecte les variables qu'elle y trouve.

Exemple 1. Utilisation de parse_str()

```
<?php
$str = "first=valeur&second[]=ceci+marche&second[]=encore";
parse_str($str);
echo $first;     /* affiche "valeur" */
echo $second[0]; /* affiche "ceci marche" */
echo $second[1]; /* affiche "encore" */
?>
```

print (unknown)

Affiche une chaîne.

```
print (string arg)
```

print() affiche *arg*.

Voir aussi **echo()**, **printf()** et **flush()**.

printf (PHP 3, PHP 4 >= 4.0b1)

Affiche une chaîne formatée.

```
int printf (string format [, mixed args...])
```

printf() affiche les arguments en fonction du *format*. Ce format est décrit en détails dans la documentation de **sprintf()**.

Voir aussi **print()**, **sprintf()**, **sscanf()**, **fscanf()** et **flush()**.

quoted_printable_decode (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Décode une chaîne

```
string quoted_printable_decode  (string str)
```

quoted_printable_decode() retourne une chaîne 8-bit résultant du décodage de la chaîne *str*.
quoted_printable_decode() est similaire à **imap_qprint()**, hormis le fait qu'elle ne requiert pas le module IMAP.

QuoteMeta (PHP 3, PHP 4 >= 4.0b1)

Ajoute un antislash devant tous les caractères méta

```
string quotemeta (string str)
```

quotemeta() retourne une version de la chaîne *str*, avec un antislash (\) devant tous les caractères de la liste ci-dessous :

```
. \ + * ? [ ^ ] ( $ )
```

.

Voir aussi **addslashes()**, **htmlentities()**, **htmlspecialchars()**, **nl2br()** et **stripslashes()**.

rtrim (PHP 3, PHP 4 >= 4.0b1)

Efface les espaces de fin de chaîne.

```
string rtrim (string str)
```

rtrim() la chaîne *str*, débarassée de ses espaces terminaux, y compris les nouvelles lignes. **rtrim()** est un alias de **chop()**.

Exemple 1. Exemple avec rtrim()

```
<?php
$trimmed = rtrim($line);
?>
```

Voir aussi **trim()**, **ltrim()** et **rtrim()**.

sscanf (PHP 4 >= 4.0.1)

Analyse une fonction en fonction d'un format

```
mixed sscanf (string str, string format [, string var1...])
```

sscanf() est le complémentaire de **printf()**. **sscanf()** lit les données de la chaîne *str* et interprète son contenu en fonction du format *format*. Si seulement deux paramètres sont passés à **sscanf()**, les valeurs obtenues seront retournées sous forme d'un tableau.

Exemple 1. Exemple avec sscanf()

```
<?php
// lecture d'un numéro de série
$serial = sscanf("SN/2350001","SN/%d");
// et la date de fabrication
$mandate = "January 01 2000";
list($month, $day, $year) = sscanf($mandate,"%s %d %d");
echo "Le produit $serial a été fabriqué le: $year-".substr($month,0,3)."-$day\n";
?>
```

Si les paramètres optionnels sont passés, **sscanf()** retournera le nombre de valeurs assignées. Les options doivent être passées par référence.

Exemple 2. Utilisation des options avec sscanf()

```
<?php
// Lecture des informations d'auteur, et génération
// d'une entrée DocBook
$auth = "24\tVictor Hugo";
$n = sscanf($auth,"%d\t%s %s", &$id, &$first, &$last);
echo "<auteur id='$id'>
    <Prénom>$first</firstname>
    <Nom>$last</surname>
</auteur>\n";
?>
```

Voir aussi **fscanf()**, **printf()** et **sprintf()**.

setlocale (PHP 3, PHP 4 >= 4.0b1)

Change les informations locales.

```
string setlocale (mixed category, string locale)
```

category est une chaîne ou une constante qui spécifie la catégorie de fonctions qui va être affectée par les informations locales :

- LC_ALL : toutes les fonctions ci-dessous
- LC_COLLATE : pour les comparaisons de chaînes (voir **strcoll()**)
- LC_CTYPE : pour la classification de caractères et les conversions, par exemple **strtoupper()**
- LC_MONETARY : pour localeconv() - (en cours d'implémentation)
- LC_NUMERIC : pour les séparateurs décimaux
- LC_TIME : pour le format des dates et heures date avec **strftime()**

Si *locale* est une chaîne vide (""), les noms locaux prendront la valeur des variables d'environnement de même nom, ou à partir de "LANG".

Si *locale* vaut zéro ou "0", la valeur reste inchangée, mais l'état courant est retourné.

setlocale() retourne la valeur courante, ou FALSE si la fonctionnalité n'est pas encore implémentée pour la plate-forme. Une catégorie invalide provoque une alerte.

similar_text (PHP 3>= 3.0.7, PHP 4)

Calcule la similarité de deux chaînes.

```
int similar_text (string first, string second [, double percent])
```

similar_text() calcule la similarité entre deux chaînes, comme décrit par Oliver [1993]. Notez que cette implémentation n'utilise pas une pile, comme dans le pseudo-code d'Oliver, mais un appel récursif qui accélère parfois l'exécution. Notez aussi que la complexité de cet algorithme est en o(N**3) avec N la taille de la plus grande chaîne.

En passant une référence comme troisième argument, **similar_text()** va calculer le pourcentage de similarité. Il retourne le nombre de caractères correspondant l'un à l'autre, d'une chaîne à l'autre.

soundex (PHP 3, PHP 4 >= 4.0b1)

Calcule la valeur soundex d'une chaîne.

```
string soundex (string str)
```

soundex() calcule la valeur soundex de *str*.

Une valeur Soundex est telle que deux mots prononcés de la même façon auront des valeurs Soundex identiques. Cela permet d'effectuer des recherches dans les bases de données, si vous connaissez la prononciation mais pas l'orthographe. **soundex()** retourne une chaîne de 4 caractères, commençant par une lettre.

soundex() particulière a été décrite par Donald Knuth dans "The Art Of Computer Programming, vol. 3: Sorting And Searching", Addison-Wesley (1973), pp. 391-392.

Attention : le soundex dépend de la langue, et le soundex PHP est optimisé pour l'anglais. Des versions françaises existent sous forme de script.

Exemple 1. Exemple avec Soundex

```
<?php
soundex("Euler") == soundex("Ellery") == 'E460';
soundex("Gauss") == soundex("Ghosh") == 'G200';
soundex("Knuth") == soundex("Kant") == 'H416';
soundex("Lloyd") == soundex("Ladd") == 'L300';
soundex("Lukasiewicz") == soundex("Lissajous") == 'L222';
```

```
?>
```

sprintf (PHP 3, PHP 4 >= 4.0b1)

Retourne une chaîne formatée.

```
string sprintf (string format [, mixed args...])
```

sprintf() retourne une chaîne formatée avec le format *format*.

La chaîne de format est composée de 0 ou plus directives : généralement des caractères qui sont recopiés tels quels (hormis %), et des spécifications, chacune d'elle disposant de son propre paramètre. Cela s'applique à **sprintf()** et **printf()**.

Chaque conversion consiste en un signe pourcentage (%), suivi d'un ou plusieurs éléments parmi ceux-ci :

1. Une option de remplissage, qui indique quel caractère sera utilisé pour le remplissage, et la taille finale de la chaîne. Le caractère de remplissage peut être un espace ou le caractère zéro (0).). La valeur par défaut est l'espace. Une autre valeur peut être spécifiée en la préfixant par un guillemet simple ('). Voir les exemples plus loin.
2. Un argument optionnel *alignment spécifier* qui indique que le résultat doit être justifié à droite ou à gauche. Par défaut, il est justifié à gauche. Le caractère - signifie : justification à droite.
3. Argument optionnel, *width spécifier* indique le nombre minimum de caractères que la conversion devrait retourner.
4. Argument optionnel, *precision spécifier* indique le nombre de chiffres utilisé pour afficher un nombre à virgule flottante. Cette option n'a d'effet que sur les nombres à virgule de type double (Une autre fonction pratique pour formater les nombres est : **number_format()**).
5. *type specifier* indique le type de données passées en argument. Les types possibles sont :
   % - un signe pourcentage : aucun argument nécessaire.
   b - l'argument est traité comme un entier, et représenté comme un nombre binaire.
   c - l'argument est traité comme un entier, et représenté comme un nombre ascii.
   d - l'argument est traité comme un entier, et représenté comme un nombre décimal.
   u - l'argument est traité comme un entier, et représenté comme un nombre décimal non signé.
   f - l'argument est traité comme un double, et représenté comme un nombre à virgule flottante.
   o - l'argument est traité comme un entier, et représenté comme un nombre octal.
   s - l'argument est traité tel quel, et représenté comme une chaîne.
   x - l'argument est traité comme un entier, et représenté comme un nombre hexadécimal (en minuscules).
   X - l'argument est traité comme un entier, et représenté comme un nombre hexadécimal (en majuscules).

A partir de PHP 4.0.6, le paramètre *format* supportera aussi la numérotation des arguments, et leur échange. Par exemple :

Exemple 1. Echange d'arguments : cas habituel

```
<?php
$format = "Il y a %d singes dans le %s";
printf($format,$num,$location);
?>
```

Cela pourra afficher "Il y a 5 singes dans le baobab". Mais imaginons un instant que nous créons cette chaîne à partir d'un fichier séparé, car nous voulons internationaliser le message. On voudra notamment écrire librement :

Exemple 2. Echange d'arguments : cas problématique

```
<?php
$format = "Le %s contient %d singes";
printf($format,$num,$location);
?>
```

Maintenant, on a un problème. L'ordre d'utilisation des variables dans la chaîne de formatage n'est pas celui d'appel de la fonction **sprintf()**. L'idéal serait de pouvoir garder l'ordre des arguments, quel que soit l'ordre des variables fournies. Il faudrait donc indiquer dans la chaîne de formatage dans quel ordre utiliser les valeurs. On pourrait écrire ceci à la place:

Exemple 3. Echange d'arguments : solution

```
<?php
$format = "Le %2\$s contient %1\$d singes";
printf($format,$num,$location);
?>
```

Et vous pouvez désormais répéter les variables sans ajouter de nouvel argument. Par exemple :

Exemple 4. Echange d'arguments : répétition

```
<?php
$format = "Le %2\$s contient %1\$d singes.  C'est un beau %2\$s, avec %1\$d signes dessus.";
printf($format,$num,$location);
?>
```

Voir aussi **printf()**, **sscanf()**, **fscanf()** et **number_format()**.

Exemple 5. Exemple avec sprintf(): complété avec des zéros

```
<?php
$isodate = sprintf("%04d-%02d-%02d", $year, $month, $day);
?>
```

Exemple 6. Exemple avec sprintf(): format monétaire

```
<?php
$money1 = 68.75;
$money2 = 54.35;
$money = $money1 + $money2;
// echo $money affichera "123.1";
$formatted = sprintf("%01.2f", $money);
// echo $formatted affichera "123.10"
?>
```

strncasecmp (PHP 4 >= 4.0.2)

Compare en binaire des chaînes de caractères

```
int strncasecmp (string str1, string str2, int len)
```

strncasecmp() est similaire à **strcasecmp()**, à la différence près qu'elle permet de limiter le nombre de caractères utilisés pour comparer *str1* et *str1*, avec le paramètre *len*. Si une des chaînes est plus courte que *len*, alors la longueur de cette chaîne sera utilisée pour effectuer la comparaison.

strncasecmp() retourne < 0 si *str1* est plus petit que *str2*; > 0 si *str1* est plus grand que *str2*, et 0 si elles sont égales.

Voir aussi **ereg()**, **strcasecmp()**, **strcmp()**, **substr()**, **stristr()** et **strstr()**.

strcasecmp (PHP 3>= 3.0.2, PHP 4 >= 4.0b1)

Compare en binaire des chaînes, insensible à la casse.

```
int strcasecmp (string str1, string str2)
```

strcasecmp() retourne < 0 si *str1* est plus petit que *str2*; > 0 si *str1* est plus grand que *str2*, et 0 s'ils sont égaux.

Exemple 1. Exemple avec strcasecmp()

```
<?php
$var1 = "Bonjour";
$var2 = "bonjour";
if ( !strcasecmp($var1,$var2) ) {
    echo '$var1 est égal à $var2, à la casse près.';
}
?>
```

Voir aussi **ereg()**, **strcmp()**, **substr()**, **stristr()**, **strncasecmp()** et **strstr()**.

strchr (PHP 3, PHP 4 >= 4.0b1)

Renvoie la chaîne à partir de la première occurrence

```
string strchr (string haystack, string needle)
```

strchr() est un alias de **strstr()**, et lui est identique en tous points.

strcmp (PHP 3, PHP 4 >= 4.0b1)

Compare en binaire des chaînes.

```
int strcmp (string str1, string str2)
```

strcmp() retourne < 0 si *str1* est plus petit que *str2*; > 0 si *str1* est plus grand que *str2*, et 0 s'ils sont égaux.

Notez bien que la comparaison est sensible à la casse.

Voir aussi **ereg()**, **strcasecmp()**, **substr()**, **stristr()**, **strncmp()**, **strncasecmp()** et **strstr()**.

strcoll (PHP 4 >= 4.0.5)

Compare des chaînes localisées

```
int strcoll (string str1, string str2)
```

strcoll() retourne < 0 si *str1* est plus petit que *str2*; > 0 si *str1* est plus grand que *str2*, et 0 si elles sont égales. **strcoll()** utilise la configuration locale pour effectuer les comparaisons. Si la configuration locale est : C ou POSIX, **strcoll()** est équivalente à **strcmp()**.

Notez que cette comparaison est sensible à la casse, et que contrairement à **strcmp()**, **strcoll()** n'est pas binaire.

> **Note :** Ajoutée en PHP 4.0.5.

Voir aussi **ereg()**, **strcmp()**, **strcasecmp()**, **substr()**, **stristr()**, **strncasecmp()**, **strncmp()**, **strstr()** et **setlocale()**.

strcspn (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Recherche la longueur du premier segment de chaîne qui ne corresponde pas au masque donné.

```
int strcspn (string str1, string str2)
```

strcspn() retourne la longueur du premier segment de la chaîne *str1* qui *ne* contiennent *aucun* des caractères de la chaîne *str2*.

Voir aussi **strspn()**.

strip_tags (PHP 3>= 3.0.8, PHP 4)

Enlève les balises HTML et PHP.

```
string strip_tags (string str [, string allowable_tags])
```

strip_tags() recherche et supprime toutes les balises HTML et PHP d'une chaîne. En cas de balises non fermées, ou de balises mal formées, elle génère une erreur. **strip_tags()** utilise le même système que la fonction **fgetss()**.

Vous pouvez utiliser l'option *allowable_tags* pour spéficier les balises qui seront ignorées.

Note : *allowable_tags* a été ajouté en PHP 3.0.13, et PHP 4.0B3.

StripCSlashes (PHP 4 >= 4.0b4)

Déquote une chaîne quotée avec addcslashes()

```
string stripcslashes (string str)
```

stripcslashes() retourne une chaîne dont les anti-slash ont été supprimés. **stripcslashes()** reconnaît les \n, \r..., et les représentations octales et hexadécimales utilisées en C.

Note : stripcslashes() a été ajouté en PHP 4.0b3-dev.

Voir aussi **addcslashes()**.

StripSlashes (PHP 3, PHP 4 >= 4.0b1)

Enlève les slashs ajoutés par la fonction addslashes()

```
string stripslashes (string str)
```

stripslashes() retourne une chaîne dont tous les slashs ont été supprimés. (`\'` devient `'`) et ainsi de suite). Les doubles antislash sont remplacés par des simples antislash.

Voir aussi **addslashes()**.

stristr (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

strstr(), insensible à la casse.

```
string stristr (string haystack, string needle)
```

stristr() retourne tous les éléments de *`haystack`* à partir de la première occurrence de *`needle`*, jusqu'à la fin. *`needle`* et *`haystack`* sont examinés sans tenir compte de la casse.

Si *`needle`* n'est pas trouvé, retourne FALSE.

Si *`needle`* n'est pas une chaîne, elle est convertie en entier, puis est utilisée comme si elle était passée à **chr()**.

Voir aussi **strchr()**, **strrchr()**, **substr()** et **ereg()**.

strlen (PHP 3, PHP 4 >= 4.0b1)

Retourne la longueur de la chaîne.

```
int strlen (string str)
```

strlen() retourne la longueur de la chaîne *`string`*, c'est-à-dire le nombre de caractères.

strnatcmp (PHP 4 >= 4.0RC2)

Compare des chaînes par ordre "naturel"

```
int strnatcmp (string str1, string str2)
```

strnatcmp() implémente un algorithme de comparaison qui traite les chaînes alpha-numériques comme un être humain : c'est ce qui est appelé l'"ordre naturel". Un exemple de la différence de traitement entre un tel algorithme et un algorithme de comparaison de chaînes (comme lorsqu'on utilise **strcmp()**) est illustré ci-dessous :

```
<?php
$arr1 = $arr2 = array("img12.png","img10.png","img2.png","img1.png");
echo "Comparaison standard de chaînes\n";
usort($arr1,"strcmp");
print_r($arr1);
echo "\nComparaison de chaînes par ordre naturel\n";
usort($arr2,"strnatcmp");
print_r($arr2);
?>
```

L'exemple précédent affiche ceci :

```
Comparaison standard de chaînes
Array
(
    [0] => img1.png
    [1] => img10.png
```

```
    [2] => img12.png
    [3] => img2.png
)
Comparaison de chaînes par ordre naturel
Array
(
    [0] => img1.png
    [1] => img2.png
    [2] => img10.png
    [3] => img12.png
)
```

Pour plus d'informations, reportez-vous à Martin Pool Natural Order String Comparison (http://www.linuxcare.com.au/projects/natsort/).

Comme les autres fonctions de comparaison de chaînes, elle retourne une valeur < 0 si *`str1`* est plus petite que *`str2`*; > 0 si *`str1`* est plus grande que *`str2`*, et 0 si elles sont égales.

Notez que ces comparaisons sont sensibles à la casse.

Voir aussi **ereg()**, **strcasecmp()**, **substr()**, **stristr()**, **strcmp()**, **strncmp()**, **strnatcasecmp()**, **strstr()**, **natsort()**, **strncasecmp()** et **natcasesort()**.

strnatcasecmp (PHP 4 >= 4.0RC2)

Compare des chaînes par ordre "naturel" insensible à la casse

```
int strnatcasecmp (string str1, string str2)
```

strnatcasecmp() implémente un algorithme de comparaison qui traite les chaînes alpha-numériques comme un être humain : c'est ce qui est appelé l'"ordre naturel". Pour plus d'informations, reportez-vous à Martin Pool Natural Order String Comparison (http://www.linuxcare.com.au/projects/natsort/).

Comme les autres fonctions de comparaisons de chaînes, elle retourne une valeur < 0 si *`str1`* est plus petite que *`str2`*; > 0 si *`str1`* est plus grande que *`str2`*, et 0 si elles sont égales.

Voir aussi **ereg()**, **strcasecmp()**, **substr()**, **stristr()**, **strcmp()**, **strncmp()**, **strnatcmp()**, **strncasecmp()** et **strstr()**.

strncmp (PHP 4 >= 4.0b4)

Compare en binaire les premiers caractères

```
int strncmp (string str1, string str2, int len)
```

strncmp() est similaire à **strcmp()**, à la différence près que vous pouvez spécifier le nombre limite de caractères (*`len`*) utilisés pour faire la comparaison. Si l'une des chaînes est plus courte que *`len`*, alors cette longueur sera utilisée pour faire la comparaison.

Comme les autres fonctions de comparaisons de chaînes, elle retourne une valeur < 0 si *`str1`* est plus petite que *`str2`*; > 0 si *`str1`* est plus grande que *`str2`*, et 0 si elles sont égales.

Notez que la comparaison est sensible à la casse.

Voir aussi **ereg()**, **strcasecmp()**, **substr()**, **stristr()**, **strcmp()**, **strncasecmp()** et **strstr()**.

str_pad (PHP 4 >= 4.0.1)

Complète une chaîne avec une autre

```
string str_pad (string input, int pad_length [, string pad_string [, int pad_type]])
```

str_pad() complète la chaîne *input* à droite, à gauche ou dans les deux directions, avec *pad_string* jusqu'à la taille de *pad_length*. Si *pad_string* n'est pas fourni, *input* est complété avec des espaces. Sinon, il est complété avec *pad_string*.

pad_type peut prendre les valeurs de STR_PAD_RIGHT, STR_PAD_LEFT, ou STR_PAD_BOTH. Si *pad_type* n'est pas spécifiée, cela vaut STR_PAD_RIGHT.

Si *pad_length* est négative ou inférieure à la taille courante de la chaîne *input*, aucun complément n'est ajouté.

Exemple 1. Exemple avec str_pad()

```
<?php
$input = "Paris";
print str_pad($input, 10);                      // produces "Paris     "
print str_pad($input, 10, "-=", STR_PAD_LEFT);  // produces "-=-=-Paris"
print str_pad($input, 10, "_", STR_PAD_BOTH);   // produces "__Paris___"
?>
```

strpos (PHP 3, PHP 4 >= 4.0b1)

Recherche la première occurrence d'un caractère dans une chaîne.

```
int strpos (string haystack, string needle [, int offset])
```

strpos() retourne la position numérique de la première occurrence de *needle* dans la chaîne *haystack*. Contrairement à **strrpos()**, *needle* peut être une chaîne.

Si *needle* n'est pas trouvée, retourne FALSE.

> **Note :** Il est facile de confondre la valeur de retour "caractère trouvé à la position 0" et "caractère introuvable". Voici comment faire la différence :
>
> ```
> <?php
>   // PHP 4.0b3 et plus récent :
>   $pos = strpos($machaine, "b");
>   if ($pos === FALSE) { // note: trois signes égal
>     // non trouvé
>   }
>   // versions plus anciennes que 4.0b3:
>   $pos = strpos("b", $machaine);
>   if (is_string($pos) && !$pos) {
>     // non trouvé
>   }
> ?>
> ```

Si *needle* n'est pas une chaîne, elle est convertie en entier, et utilisée comme la valeur ASCII d'un caractère.

L'argument optionnel *offset* permet de préciser le caractère à partir duquel chercher, dans *haystack*. La position doit être relative au début de la chaîne *haystack*.

Voir aussi **strrpos()**, **strrchr()**, **strchr()**, **substr()**, **stristr()** et **strstr()**.

strrchr (PHP 3, PHP 4 >= 4.0b1)

Recherche la partie terminale d'une chaîne après un caractère donné

```
string strrchr (string haystack, string needle)
```

strrchr() retourne la partie de la chaîne *haystack* qui commence à la dernière occurrence de *needle* et va jusqu'à la fin de la chaîne *haystack*.

strrchr() retourne FALSE si *needle* n'est pas trouvé.

Si *needle* contient plus d'un caractère, les autres sont ignorés.

Si *needle* n'est pas une chaîne, il est converti en un entier, et utilisé comme valeur ordinale.

Exemple 1. Exemple avec strrchr()

```
<?php
// lit le dernier répertoire de $PATH
$dir = substr(strrchr($PATH, ":"), 1);
// lit tout après la dernière ligne
$text = "Line 1\nLine 2\nLine 3";
$last = substr(strrchr($text, 10), 1 );
?>
```

Voir aussi **substr()**, **stristr()** et **strstr()**.

str_repeat (PHP 4 >= 4.0b4)

Répète une chaîne.

```
string str_repeat (string input, int multiplier)
```

str_repeat() retourne *input_str* répétée *multiplier* fois. *multiplier* doit être supérieur à 0.

Exemple 1. Exemple avec str_repeat()

```
<?php
echo str_repeat("-=", 10);
?>
```

Cet exemple affichera "-=-=-=-=-=-=-=-=-=-=".

> **Note : str_repeat()** a été ajoutée en PHP 4.0.

strrev (PHP 3, PHP 4 >= 4.0b1)

Inverse l'ordre des caractères d'une chaîne.

```
string strrev (string string)
```

strrev() retourne *string*, après avoir changé l'ordre des caractères.

strrpos (PHP 3, PHP 4 >= 4.0b1)

Recherche la dernière occurrence d'un caractère dans une chaîne.

```
int strrpos (string haystack, char needle)
```

strrpos() retourne la position numérique de la dernière occurrence de *needle* dans la chaîne *haystack*. **strrpos()** ne peut accepter qu'un seul caractère.

Si *needle* n'est pas trouvé, retourne FALSE.

> **Note :** Il est facile de confondre la valeur de retour "caractère trouvé à la position 0" et "caractère introuvable". Voici comment faire la différence :
>
> ```
> <?php
> // PHP 4.0b3 et plus récent :
> $pos = strpos("b", $mystring);
> if ($pos === FALSE) { // note: trois égal signes
>     // non trouvé
> }
> // versions plus anciennes que 4.0b3:
> $pos = strpos("b", $mystring);
> if (is_string($pos) && !$pos) {
>     // non trouvé
> }
> ?>
> ```

Si *needle* n'est pas une chaîne, elle est convertie en entier, et utilisée comme la valeur ASCII d'un caractère.

Voir aussi **strpos()**, **strrchr()**, **substr()**, **stristr()** et **strstr()**.

strspn (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Retourne la longueur du premier segment qui vérifie le masque.

```
int strspn (string str1, string str2)
```

strspn() retourne la longueur du premier segment de *str1* qui est constitué entièrement de caractères dans la chaîne *str2*.

```
<?php
strspn("42 est une réponse, quelle est la question...", "1234567890");
?>
```

Cet exemple affichera "2", car "42" est la plus longue chaîne contenant des chiffres dans la chaîne de questions.

Voir aussi **strcspn()**.

strstr (PHP 3, PHP 4 >= 4.0b1)

Renvoie la chaîne à partir de la première occurrence

```
string strstr (string haystack, string needle)
```

strstr() retourne toute la chaîne *haystack* à partir de la première occurrence de *needle*, jusqu'à la fin.

Si *needle* n'est pas trouvé, retourne FALSE.

> **Note : strstr()** est sensible à la casse. Si besoin est, utilisez **stristr()**.

Si *needle* n'est pas une chaîne, elle est convertie en entier, et utilisée comme valeur ordinale d'un caractère.

Exemple 1. Exemple avec strstr()

```
<?php
$email = 'sterling@designmultimedia.com';
$domain = strstr($email, '@');
print $domain;
// affiche "@designmultimedia.com"
?>
```

Voir aussi **stristr()**, **strrchr()**, **substr()** et **ereg()**.

strtok (PHP 3, PHP 4 >= 4.0b1)

Morcelle une chaîne

```
string strtok (string arg1, string arg2)
```

strtok() est utilisée pour morceler une chaîne. Pour cela, si vous avez une chaîne du type "ceci est une chaîne exemple", vous pouvez la morceler en mots, en utilisant ' ' comme délimiteur.

Exemple 1. Exemple avec strtok()

```
<?php
$string = "ceci est une chaîne exemple";
$tok = strtok($string," ");
while ($tok) {
    echo "Mot=$tok<br>";
    $tok = strtok(" ");
}
?>
```

Notez que seul, le premier appel à **strtok()** utilise l'argument chaîne. Après, chaque appel à strtok ne requiert que le délimiteur à utiliser. Pour recommencer, vous pouvez simplement appeler **strtok()** avec un nouvel argument, pour l'initialiser. Notez que vous pouvez mettre des délimiteurs multiples. La chaîne sera morcelée à chaque fois qu'on rencontrera un des délimiteurs.

Soyez prudents avec les délimiteurs qui sont égaux à "0". Cette valeur sera confondue avec FALSE.

Voir aussi **split()** et **explode()**.

strtolower (PHP 3, PHP 4 >= 4.0b1)

Met tous les caractères en minuscules.

```
string strtolower (string str)
```

strtolower() retourne *string* avec tous les caractères alphabétiques en minuscule.

Notez que le caractère 'alphabétique' est déterminé par la table de caractères locale. Par exemple, dans la table des caractères par défaut du "C", des caractères tels que a-umlaut (ä) ne seront pas convertis.

Exemple 1. Exemple avec strtolower()

```
<?php
$str = "Marie A Un Petit Agneau, Et Elle L'Adore";
$str = strtolower($str);
print $str;
# Affiche : marie a un petit agneau, et elle l'adore
?>
```

Voir aussi **strtoupper()** et **ucfirst()**.

strtoupper (PHP 3, PHP 4 >= 4.0b1)

Met tous les caractères en majuscules.

```
string strtoupper (string string)
```

strtoupper() retourne *string* avec tous ses caractères alphabétiques mis en majuscule.

Notez que le caractère 'alphabétique' est déterminé par la table de caractères locale. Par exemple, dans la table des caractères par défaut du "C", des caractères tels que a-umlaut (ä) ne seront pas convertis.

Exemple 1. Exemple avec strtoupper()

```
<?php
$str = "Marie A Un Petit Agneau, Et Elle L'Adore";
$str = strtoupper($str);
print $str; # Affiche : MARIE A UN PETIT AGNEAU, ET ELLE L'ADORE
?>
```

Voir aussi **strtolower()** et **ucfirst()**.

str_replace (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Remplace toutes les occurrences d'une chaîne par une autre.

```
mixed str_replace (mixed search, mixed replace, mixed subject)
```

str_replace() remplace toutes les occurrences de *search* dans *subject* par la chaîne *replace*. Si vous n'avez pas besoin de règles de remplacement sophistiquées, utilisez **str_replace()** de préférence à **ereg_replace()** et **preg_replace()**.

En PHP 4.0.5 et plus récent, chaque paramètre de **str_replace()** peut être un tableau.

Si *subject* est un tableau, alors le remplacement est effectué pour chaque valeur de *subject*, et la valeur retournée sera un tableau.

Si *search* et *replace* sont des tableaux, alors **str_replace()** prend une valeur dans chaque tableau, et s'en sert pour chercher et remplacer dans *subject*. Si *replace* contient moins de valeurs que *search*, des chaînes vides seront utilisées pour compléter le tableau *replace*. Si *search* est un tableau et *replace* est une chaîne, alors la même chaîne de remplacement sera utilisée pour chaque valeur de *search*. Le contraire n'aurait pas beaucoup de sens.

Exemple 1. Exemple avec str_replace()

```
<?php
$bodytag = str_replace("%body%", "black", "<body text=%body%>");
?>
```

str_replace() n'altère pas les données binaires.

> **Note : str_replace()** a été ajoutée en PHP 3.0.6, mais était erronée jusqu'à PHP 3.0.8.

Voir aussi **ereg_replace()**, **preg_replace()** et **strtr()**.

strtr (PHP 3, PHP 4 >= 4.0b1)

Remplace toutes les occurrences d'un caractère par un autre.

```
string strtr (string str, string from, string to)
```

strtr() travaille sur *str*, remplaçant chaque occurrence de chaque caractère de la chaîne *from* correspondant à la chaîne *to* et retourne le résultat.

Si *from* et *to* sont de longueurs différentes, les caractères en trop sont ignorés.

Exemple 1. Exemple avec strtr()

```
<?php
$addr = "Le gâteau au maïs aigü";
$addr = strtr($addr, "âïü", "aiu");
print $addr;
// Affiche : "Le gateau au mais aigu"
// Note : ne cherchez pas la recette...
?>
```

strtr() peut aussi être appelée avec deux arguments. Dans ce cas, elle se comporte différemment : *from* doit être un tableau associatif contenant des paires de chaînes, qui seront remplacées dans la chaîne source. **strtr()** recherchera toujours la chaîne la plus longue, et la remplacera en premier. Elle ne remplacera jamais une portion de chaîne qu'elle a déjà remplacé.

Exemples:

```
<?php
$trans = array("bonjour" => "salut", "salut" => "bonjour");
echo strtr("bonjour à tous, j'ai dit salut", $trans)."\n";
?>
```

Cet exemple affichera : "salut à tous, j'ai dit bonjour",

> **Note :** Travailler avec deux arguments a été ajouté en PHP 4.0.

Voir aussi **ereg_replace()**.

substr (PHP 3, PHP 4 >= 4.0b1)

Retourne une partie de la chaîne.

```
string substr (string string, int start [, int length])
```

substr() retourne une portion de *string*, spécifiée avec le début *start* et la longueur *length*.

Si *start* est positif, la chaîne retournée commencera au caractère *start* de la chaîne *string*. Par exemple, dans la chaîne 'abcdef', le caractère à la position 0 est 'a', le caractère à la position 2 est 'c', et ainsi de suite. Par exemple:

```
<?php
  $reste = substr("abcdef", 1);    // retourne "bcdef"
  $reste = substr("abcdef", 1, 3); // retourne "bcd"
?>
```

Si *start* est négatif, la chaîne retournée commencera au caractère *start* de la chaîne *string*, en partant de la fin. Par exemple:

```
<?php
  $reste = substr("abcdef", -1);    // retourne "f"
  $reste = substr("abcdef", -2);    // retourne "ef"
  $reste = substr("abcdef", -3, 1); // retourne "d"
?>
```

Si *length* est donné et positive, la chaîne retournée aura la longueur *length*. Si *length* est donnée et négative, la chaîne retournée aura la longueur *length*, en partant de la fin.

```
<?php
  $reste = substr("abcdef", 1, -1);
  // retourne "bcde"
?>
```

Voir aussi **strrchr()** et **ereg()**.

substr_count (PHP 4 >= 4.0RC2)

Compte le nombre de sous-chaînes

```
int substr_count (string haystrack, string needle)
```

substr_count() retourne le nombre de fois que *needle* apparaît dans *haystack*.

Exemple 1. Exemple substr_count()

```
<?php
print substr_count("Ceci est un test", "es"); // affiche 2
?>
```

substr_replace (PHP 4 >= 4.0b4)

Remplace dans une sous partie de chaîne

```
string substr_replace (string string, string replacement, int start [, int length])
```

substr_replace() effectue un remplacement dans la portion de *string* délimitée par le caractère *start* et de longueur optionnelle *length*. Le remplacement est fait avec la chaîne *replacement*. Le résultat est retourné.

Si *start* est positif, le remplacement commencera au caractère *start*, dans la chaîne *string*.

Si *start* est négative, le remplacement commencera au caractère *start* en partant de la fin de la chaîne *string*.

Si *length* est donné et positif, la chaîne retournée aura la longueur *length*. Si *length* est donné et négatif, la chaîne retournée aura la longueur *length*, en partant de la fin. Par défaut, il prendra la valeur de strlen(*string*); c'est-à-dire qu'il remplacera jusqu'à la fin de la chaîne *string*.

Exemple 1. Exemple avec substr_replace()

```
<?php
$var = 'ABCDEFGH:/MNRPQR/';
echo "Original: $var<hr>\n";
/* Ces deux exemples remplacent tout $var avec 'bob'. */
echo substr_replace($var, 'bob', 0)."<br>\n";
echo substr_replace($var, 'bob', 0, strlen($var))."<br>\n";
/* Insère 'bob' à gauche, du début de $var. */
echo substr_replace($var, 'bob', 0, 0)."<br>\n";
/* Ces deux exemples remplacent 'MNRPQR' dans $var avec 'bob'. */
echo substr_replace($var, 'bob', 10, -1)."<br>\n";
echo substr_replace($var, 'bob', -7, -1)."<br>\n";
/* Efface 'MNRPQR' dans $var. */
echo substr_replace($var, '', 10, -1)."<br>\n";
?>
```

Voir aussi **str_replace()** et **substr()**.

> **Note : substr_replace()** a été ajoutée en PHP 4.0.

trim (PHP 3, PHP 4 >= 4.0b1)

Enlève les espaces de début et de fin de chaîne

```
string trim (string str)
```

trim() retire les espaces blancs de début et de fin de chaîne, et retourne la chaîne nettoyée. Les espaces blancs sont : "\n", "\r", "\t", "\v", "\0", et " " (espace).

Voir aussi **chop()** et **ltrim()**.

ucfirst (PHP 3, PHP 4 >= 4.0b1)

Force le premier caractère d'une chaîne en majuscule.

```
string ucfirst (string str)
```

ucfirst() met le premier caractère d'une chaîne *str* en majuscules, si ce caractère est alphabétique.

Notez que le caractère 'alphabétique' est déterminé par la table de caractères locale. Par exemple, dans la table des caractères par défaut du "C", des caractères tels que a-umlaut (ä) ne seront pas convertis.

Exemple 1. Exemple avec ucfirst()

```
<?php
$text = 'marie a un petit agneau, et l'adore.';
$text = ucfirst($text); // $text vaut : Marie a un petit agneau, et l'adore
?>
```

Voir aussi **strtoupper()** et **strtolower()**.

ucwords (PHP 3>= 3.0.3, PHP 4 >= 4.0b1)

Force le premier caractère de chaque mot d'une chaîne en majuscule

```
string ucwords (string str)
```

ucwords() met le premier caractère de chaque mot de la chaîne *str* si ce caractère est une lettre.

Exemple 1. Exemple avec ucwords()

```
<?php
$text = "marie a un petit agneau, et l'adore.";
$text = ucwords($text); // $text vaut : Marie A Un Petit Agneau, Et l'Adore.
?>
```

> **Note :** La définition d'un mot est : une chaîne de caractères immédiatement après un caractère blanc (c'est-à-dire : espace, form-feed, nouvelle ligne, retour chariot, tabulation horizontale, et tabulation verticale).

Voir aussi **strtoupper()**, **strtolower()** et **ucfirst()**.

wordwrap (PHP 4 >= 4.0.2)

Ajoute une césure à une chaîne tous les n caractères

```
string wordwrap (string str [, int width [, string break [, int cut]]])
```

wordwrap() ajoute la césure *str* au numéro de colonne *width*. La ligne est césurée avec la chaîne *break*.

wordwrap() ajoute la césure automatiquement à la ligne 75 et utilise '\n' (nouvelle ligne) si *width* ou *break* sont omis.

Si *cut* vaut 1, la chaîne sera toujours coupée à la taille spécifiée. Dans ce cas, les mots qui dépasseront, seront coupés : voyez le second exemple.

> **Note :** Le paramètre *cut* a été ajouté dans PHP 4.0.3.

Exemple 1. Exemple wordwrap()

```
<?php
$text = "Maître corbeau jura, mais un peu tard, qu’on ne l’y prendrait plus.";
$newtext = wordwrap( $text, 20 );
echo "$newtext\n";
?>
```

Cet exemple va afficher :

```
Maître corbeau
 jura, mais un peu t
ard, qu’on ne l’y pr
endrait plus.
```

Exemple 2. Exemple avec wordwrap()

```
<?php
$text = "Un très très long mooooooooooooooooooooooot.";
$newtext = wordwrap( $text, 8, "\n", 1);
echo "$newtext\n";
?>
```

Cet exemple va afficher

```
Un très
très
long
mooooooo
oooooooo
oooooooo
oot.
```

Voir aussi **nl2br()**.

LXXXIV. Sybase

sybase_affected_rows (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Retourne le nombre de lignes affectées par la dernière requête.

```
int sybase_affected_rows ([resource link_identifier])
```

sybase_affected_rows() retourne le nombre de lignes affectées par la dernière requête.

sybase_affected_rows() retourne le nombre de lignes affectées par la dernière requête INSERT, UPDATE ou DELETE sur le serveur associé à l'identifiant de connexion *link_identifier*. Si le lien n'est pas précisé, le dernier lien ouvert est utilisé.

Cette commande ne sert à rien sur les requête SELECT : uniquement sur des requêtes qui modifient les lignes. Pour connaître le nombre de lignes retournées par un SELECT, utilisez **sybase_num_rows()**.

> **Note : sybase_affected_rows()** est disponible avec l'interface CT vers Sybase, mais pas avec la librairie DB.

sybase_close (PHP 3, PHP 4 >= 4.0b1)

Ferme une connexion Sybase.

```
bool sybase_close (resource link_identifier)
```

sybase_close() retourne TRUE en cas de succès et FALSE en cas d'erreur.

sybase_close() termine la connexion avec le serveur Sybase associé à l'identifiant de connexion *link_identifier*.

Notez qu'il n'est pas utile de fermer les connexions non persistantes, car elles seront terminées à la fin du script.

sybase_close() ne ferme pas les connexions persistantes générées par **sybase_pconnect()**.

Voir aussi **sybase_connect()** et **sybase_pconnect()**.

sybase_connect (PHP 3, PHP 4 >= 4.0b1)

Ouvre une connexion à un serveur Sybase.

```
resource sybase_connect (string servername, string username, string password)
```

sybase_connect() retourne un identifiant positif de lien Sybase en cas de succès, et FALSE en cas d'erreur.

sybase_connect() établit une connexion à un serveur Sybase. Le nom de serveur *servername* doit être valide, défini dans le fichier d'interface.

Si un deuxième appel à **sybase_connect()** est fait avec les mêmes arguments, une nouvelle connexion ne sera pas établie, mais ce sera l'identifiant de la connexion déjà ouverte qui sera retourné.

La connexion sera fermée dès la fin du script, à moins qu'elle ne soit pas explicitement fermée avec **sybase_close()**.

Voir aussi **sybase_pconnect()** et **sybase_close()**.

sybase_data_seek (PHP 3, PHP 4 >= 4.0b1)

Déplace le pointeur interne de lignes.

```
bool sybase_data_seek (resource result_identifier, int row_number)
```

sybase_data_seek() retourne TRUE en cas de succès, et FALSE en cas d'échec.

sybase_data_seek() déplace le pointeur interne de ligne du résutalt Sybase associé à *result_identifier* jusqu'à la ligne *row_number*. Le prochain appel à **sybase_fetch_row()** sans préciser la ligne, retournera la ligne *row_number*.

Voir aussi **sybase_data_seek()**.

sybase_fetch_array (PHP 3, PHP 4 >= 4.0b1)

Retourne une ligne sous la forme d'un tableau.

```
array sybase_fetch_array (resource result)
```

sybase_fetch_array() retourne un tableau qui contient la ligne demandée, ou FALSE s'il ne reste plus de ligne.

sybase_fetch_array() est une version évoluée de **sybase_fetch_row()**. En plus d'enregistrer les données dans un tableau à index numérique, cette fonction peut aussi les enregistrer dans un tableau associatif, en utilisant les nom des champs comme clés.

Il est très important de noter que**sybase_fetch_array()** N'est PAS nettement plus lent que **sybase_fetch_row()**, tandis qu'elle fourni un confort d'utilisation notable.

Pour plus de détails : **sybase_fetch_row()**.

sybase_fetch_field (PHP 3, PHP 4 >= 4.0b1)

Lit les informations d'un champs.

```
object sybase_fetch_field (resource result, int [field_offset])
```

sybase_fetch_field() retourne un objet contenant les informations du champs.

sybase_fetch_field() sert à obtenir des informations à propos des champs dans le résultat *result*. Si l'offset du champs n'est pas précisé, le champs suivant est traité.

Les propriétés des objets sont :

- name - column name. nom de la colonne. Si la colonne est un résultat de fonction, le nom de cette fonction devient computed#N, oú #N est un numéro de série.
- column_source - la table d'origine de la colonne.
- max_length - taille maximale de la colonne
- numeric - 1 si la colonne est de type numérique.
- type - type de données de la colonne

Voir aussi **sybase_field_seek()**.

sybase_fetch_object (PHP 3, PHP 4 >= 4.0b1)

Retourne une ligne sous la forme d'un objet.

```
int sybase_fetch_object (resource result)
```

sybase_fetch_object() retourne un objet qui contient la ligne demandée, en cas de succès, et FALSE en cas d'erreur.

sybase_fetch_object() est similaire à **sybase_fetch_array()**, avec une différence : c'est un objet qui est retourné à la place d'un tableau. Indirectement, cela signifie que vous ne pourrez accéder aux valeurs que par les propriétés, et non plus avec des offsets (les nombres sont interdits comme nom de propriété).

Au niveau de la vitesse, cette fonction est identique à **sybase_fetch_array()**, et presque aussi rapide que **sybase_fetch_row()** (la différence est insignifiante).

Voir aussi **sybase_fetch-array()** et **sybase_fetch-row()**.

sybase_fetch_row (PHP 3, PHP 4 >= 4.0b1)

Retourne une ligne sous la forme d'un tableau énuméré.

```
array sybase_fetch_row (resource result)
```

sybase_fetch_row() retourne un tableau qui contient la ligne demandée, en cas de succès, et FALSE en cas d'erreur.

sybase_fetch_row() lit une ligne dans le résultat associé à l'identifiant de résultat *result*. La ligne retournée est sous la forme d'un tableau. Chaque champs est enregistré dans un index du tableau, les index commencant à 0.

Les prochains appels à **sybase_fetch_row()** retourneront la ligne suivante du résultat, ou FALSE, si il ne reste plus de lignes.

Voir aussi **sybase_fetch_array()**, **sybase_fetch_object()**, **sybase_data_seek()** et **sybase_result()**.

sybase_field_seek (PHP 3, PHP 4 >= 4.0b1)

Modifie l'index d'un champs.

```
int sybase_field_seek (resource result, int field_offset)
```

sybase_field_seek() modifie l'index d'un champs. Le prochain appel à la fonction **sybase_fetch_field()** sans préciser l'index du champs retournera ce champs.

Voir aussi **sybase_fetch_field()**.

sybase_free_result (PHP 3, PHP 4 >= 4.0b1)

Libère un résultat de la mémoire.

```
bool sybase_free_result (int result)
```

sybase_free_result() n'est vraiment utile que si vous risquez d'utiliser trop de mémoire durant votre script. La mémoire occupée par les résultats est automatiquement libérée à la fin du script. Mais, si vous êtes sÛr de ne pas avoir besoin du résultat ultérieurement.

sybase_get_last_message (PHP 3, PHP 4 >= 4.0b1)

Retourne le dernier message du serveur

```
string sybase_get_last_message (void)
```

sybase_get_last_message() retourne le dernier message rapporté par le serveur.

sybase_min_client_severity (PHP 3, PHP 4 >= 4.0b1)

Fixe la sévérité minimale du client

```
void sybase_min_client_severity (int severity)
```

sybase_min_client_severity() fixe la sévérité minimale du client.

Note : sybase_min_client_severity() est disponible avec l'interface CT vers Sybase, mais pas avec la librairie DB.

Voir aussi **sybase_min_server_severity()**.

sybase_min_error_severity (PHP 3, PHP 4 >= 4.0b1)

Fixe la sévérité minimale du client pour les erreurs

```
void sybase_min_error_severity (int severity)
```

sybase_min_error_severity() fixe la sévérité minimale du client pour les erreurs.

Voir aussi **sybase_min_message_severity()**.

sybase_min_message_severity (PHP 3, PHP 4 >= 4.0b1)

Fixe la sévérité minimale du client pour les messages

```
void sybase_min_message_severity (int severity)
```

sybase_min_message_severity() fixe la sévérité minimale du client pour les messages.

Voir aussi **sybase_min_error_severity()**.

sybase_min_server_severity (PHP 3, PHP 4 >= 4.0b1)

Fixe la sévérité minimale du client pour le serveur

```
void sybase_min_server_severity (int severity)
```

sybase_min_server_severity() fixe la sévérité minimale du client pour le serveur.

Note : sybase_min_server_severity() est disponible avec l'interface CT vers Sybase, mais pas avec la librairie DB.

Voir aussi **sybase_min_client_severity()**.

sybase_num_fields (PHP 3, PHP 4 >= 4.0b1)

Retourne le nombre de champs dans un résultat.

```
int sybase_num_fields (resource result)
```

sybase_num_fields() retourne le nombre de champs du résultat *result*.

Voir aussi **sybase_query()**, **sybase_fetch_field()** et **sybase_num_rows()**.

sybase_num_rows (PHP 3, PHP 4 >= 4.0b1)

Retourne le nombre de lignes dans un résultat.

```
int sybase_num_rows (resource result)
```

sybase_num_rows() retourne le nombre de lignes du résultat *result*.

Voir aussi **sybase_query()** et **sybase_fetch_row()**.

sybase_pconnect (PHP 3, PHP 4 >= 4.0b1)

Ouvre une connexion persistante à un serveur Sybase.

```
resource sybase_pconnect (string servername, string username, string password)
```

sybase_pconnect() retourne un identifiant de connexion positif en cas de succès, et FALSE en cas d'erreur.

sybase_connect() se comporte comme **sybase_connect()** avec deux différence majeures :

Premièrement, lors de la connexion, la fonction va chercher une connexion (persistante) déjà ouverte, avec le même hôte, nom de compte et mot de passe. Si une telle connexion est trouvée, un identifiant de cette connexion est retourné, plutôt que d'en ouvrir une nouvelle.

Deuxièmement, la connexion au serveur SyBase ne sera pas terminée lors de la fin du script. Au contraire, le lien sera maintenu pour des connexions ultérieures. **sybase_close()** ne fermera pas un lien crée par **sybase_pconnect()**.

Ce type de liens est dit 'persistent'.

sybase_query (PHP 3, PHP 4 >= 4.0b1)

Envoie une requête à une base Sybase.

```
int sybase_query (string query, int link_identifier)
```

sybase_query() retourne un identifiant de résultat positif en cas de succès, et FALSE sinon.

sybase_query() envoie une requête à la base de données courante, sur le serveur associé à l'identifiant de connexion. Si l'identifiant de connexion n'est pas précisé, la fonction essaiera d'utiliser la dernière connexion ouverte. Si aucune connexion n'a été ouverte, la fonction va tenter d'ouvrir une connexion avec la fonction **sybase_connect()**.

Voir aussi **sybase_select_db()** et **sybase_connect()**.

sybase_result (PHP 3, PHP 4 >= 4.0b1)

Lit une valeur dans un résultat.

```
string sybase_result (resource result, int i, mixed field)
```

sybase_result() retourne le contenu d'une cellule. L'argument *field* peut être l'index du champs, ou bien le nom du champs, ou encore, le nom de la table " point " le nom du champs. Si la colonne a été aliasée ('SELECT foo AS bar FROM...'), utilisez l'alias à la place du nom de la colonne.

Lorsque vous travaillez sur des résultats de grande taille, vous devriez utiliser les autres fonctions qui lisent une ligne entière (voir plus loin). Etant donné que ces fonctions lisent une ligne entière, elles sont BEAUCOUP plus rapide que **sybase_result()**. De plus, l'utilisation d'index numérique est beaucoup plus rapide que les noms des champs, ou les noms des tables et des champs.

Fonctions de substitution, à haute efficacité : **sybase_fetch_row()**, **sybase_fetch_array()** et **sybase_fetch_object()**.

sybase_select_db (PHP 3, PHP 4 >= 4.0b1)

Sélectionne une base de données Sybase.

```
bool sybase_select_db (string database_name, resource link_identifier)
```

sybase_select_db() retourne TRUE en cas de succès, et FALSE en cas d'erreur.

sybase_select_db() change la base de données courante et active sur le serveur associé avec l'identifiant de connexion *link_identifier*. Si *link_identifier* n'est pas précisé, le dernier lien ouvert est utilisé. Si aucun lien n'a été ouvert, la fonction va tenter d'en établir un en appelant **sybase_connect()**.

Tous les prochains appels à **sybase_query()** seront faites sur la bas de données courante et active.

Voir aussi **sybase_connect()**, **sybase_pconnect()** et **sybase_query()**.

LXXXV. URL

base64_decode (PHP 3, PHP 4 >= 4.0b1)

Décode une chaîne en MIME base64

```
string base64_decode (string encoded_data)
```

base64_decode() décode *encoded_data* et retourne les données décodées. Les informations initiales peuvent être binaires.

Voir aussi : **base64_encode()** et la RFC-2045 section 6.8.

base64_encode (PHP 3, PHP 4 >= 4.0b1)

Encode une chaîne en MIME base64.

```
string base64_encode (string data)
```

base64_encode() retourne *data* encodé en base64. Cet encodage est fait pour permettre aux informations binaires d'être manipulées par les systèmes qui ne gèrent pas correctement les 8 bits, comme par exemple, les corps de mail.

Une chaîne encodée Base64 prend, grosso modo, 33% de plus que les données initiales.

Voir aussi: **base64_decode()**, **chunk_split()**, et la RFC-2045 section 6.8.

parse_url (PHP 3, PHP 4 >= 4.0b1)

Analyse une URL et retourne ses composants.

```
array parse_url (string url)
```

parse_url() retourne un tableau associatif contenant les composants de l'URL. Les composants recherchés sont : "scheme", "host", "port", "user", "pass", "path", "query", et "fragment".

rawurldecode (PHP 3, PHP 4 >= 4.0b1)

Décode une chaîne URL.

```
string rawurldecode (string str)
```

rawurldecode() retourne une chaîne dont les séquences de caractères %xy, avec xy deux valeurs hexadécimales, auront été remplacées par le caractère ASCII correspondant. Par exemple, la chaîne

```
foo%20bar%40baz
```

devient

```
foo bar@baz
```

.

Voir aussi **rawurlencode()**, **urldecode()** et **urlencode()**.

rawurlencode (PHP 3, PHP 4 >= 4.0b1)

Encode une chaîne en URL, selon la RFC1738.

```
string rawurlencode (string str)
```

rawurlencode() retourne une chaîne dont tous les caractères non-alpha-numériques (hormis

`-_.`

) auront été remplacés par des séquences `%xy` (%), avec xy deux valeurs hexadécimales. Ce codage est conforme à la RFC1738 qui évite que les caractères spéciaux soient interprétés comme des délimiteurs, et pour protéger les URL lors du transfert (contrairement à certains systèmes email). Par exemple, si vous voulez mettre un mot de passe dans une URL de ftp :

Exemple 1. Exemple avec rawurlencode()

```
<?php
echo '<A HREF="ftp://user:', rawurlencode ('foo @+%/'), '@ftp.my.com/x.txt">';
?>
```

Ou, si vous transmettez un chemin dans une URL

Exemple 2. Exemple avec rawurlencode()

```
<?php
echo '<A HREF="http://x.com/department_list_script/', rawurlencode ('sales et marke-
ting/Miami'), '">';
?>
```

Voir aussi **rawurldecode()**, **urldecode()** et **urlencode()**.

urldecode (PHP 3, PHP 4 >= 4.0b1)

Décode une chaîne encodée URL.

```
string urldecode (string str)
```

urldecode() décode toutes les séquences `%##` et les remplace par leur valeur. La chaîne ainsi décodée est retournée.

Exemple 1. Exemple avec urldecode()

```
<?php
$a = split ('&', $querystring);
$i = 0;
while ($i < count ($a)) {
    $b = split ('=', $a [$i]);
    echo 'Value for parameter ', htmlspecialchars (urldecode ($b [0])),
         ' is ', htmlspecialchars (urldecode ($b [1])), "<BR>";
    $i++;
}
?>
```

Voir aussi **urlencode()**, **rawurlencode()** et **rawurldecode()**.

urlencode (PHP 3, PHP 4 >= 4.0b1)

Encode une chaîne en URL.

```
string urlencode (string str)
```

urlencode() retourne une chaîne dont les caractères non alpha-numériques (hormis `-_.`) sont remplacés par des séquences commencant par un caractère pourcentage (%), suivi de deux chiffres hexadécimaux. Les espaces sont remplacés par des signes plus (+). Ce codage est celui qui est utilisé pour poster des informations dans les formulaires HTML. Le type MIME est `application/x-www-form-urlencoded`. Ce codage est différent de celui spécifié dans la RFC1738 (voir **rawurlencode()**) : pour des raisons historiques, les espaces sont remplacés par des signes plus (+). **urlencode()** est pratique pour transmettre des informations via une URL. C'est aussi un moyen de passer des informations d'une page à l'autre.

Exemple 1. Exemple avec urlencode()

```
<?php
echo '<A HREF="moncgi?foo=', urlencode ($userinput), '">';
?>
```

Voir aussi **urldecode()**.

Note: Faîtes bien attention aux variables qui ressemblent à des entitées HTML, comme par exemple &amp, &copy et &pound, qui sont analysées par le client web et remplacée par leur valeur, au lieu de passer le nom de variable désiré. C'est un vrai problème qui a été montré par le W3C depuis longtemps. La référence est ici : http://www.w3.org/TR/html4/appendix/notes.html#h-B.2.2. PHP supporte le remplacement de séparateur d'arguments par un point-virgule, comme recommandé par le W3C, grâce à la directive arg_separator .ini. Malheureusement, la plus part des clients web n'envoie pas leur données de formulaire avec des point-virgule. Une solution plus portable est d'utiliser `&amp;` à la place de & comme séparateur. Vous n'avez alors pas à changer la directive arg_separator. Laissez la à &, mais encodez vos URL avec **htmlentities()**.

Exemple 2. Exemple avec urlencode() et htmlentities()

```
<?php
echo '<A HREF="moncgi?foo=', htmlentities (urlencode ($userinput) ), '">';
?>
```

Voir aussi **urldecode()**, **htmlentities()**, **rawurldecode()** et **rawurlencode()**.

LXXXVI. Variables

doubleval (PHP 3, PHP 4 >= 4.0b1)

Retourne la valeur numérique (double) de la variable.

```
double doubleval (mixed var)
```

doubleval() retourne la valeur numérique (double) de la variable *var*.

var peut être de type scalaire. Vous ne pouvez pas utiliser la fonction **doubleval()** avec un tableau ou un objet.

```
<?php
$var = '122.34343Le';
$double_valeur_de_var = doubleval($var);
print $double_valeur_de_var; // affiche 122.34343
?>
```

Voir aussi **intval()**, **strval()**, **settype()** et Transtypage.

empty (unknown)

Détermine si une variable est affectée.

```
int empty (mixed var)
```

empty() retourne la valeur FALSE si la variable *var* est affectée ou bien a une valeur différente de 0; la valeur TRUE dans les autres cas.

```
<?php
$var = 0;
if (empty($var)) { // retourne TRUE
  print 'soit $var vaut 0, soit il n'est pas défini';
}
if (!isset($var)) { // retourne FALSE
  print '$var n'est pas définie';
}
?>
```

Notez que cette fonction n'a pas de sens si elle est utilisée sur autre chose qu'une variable. i.e. `empty (addslashes ($name))` n'a pas de sens, car cela revient à vérifier une entité qui n'est pas une variable.

Voir aussi **isset()** et **unset()**.

gettype (PHP 3, PHP 4 >= 4.0b1)

Retourne le type de la variable.

```
string gettype (mixed var)
```

gettype() retourne le type de la variable PHP *var*.

Les chaînes de caractères que peut retourner la fonction sont les suivantes :

- "boolean"
- "integer"

- "double"
- "string"
- "array"
- "object"
- "resource"
- "user function"
- "unknown type"

Voir aussi **settype()**.

get_defined_vars (PHP 4 >= 4.0.4)

Liste toutes les variables définies

```
array get_defined_vars (void )
```

get_defined_vars() retourne un tableau multidimensionnel contenant la liste de toutes les variables définies, qu'elles soient des variables d'environnement, de serveurs ou définies par l'utilisateur.

```
<?php
$b = array(1,1,2,3,5,8);
$arr = get_defined_vars();
// affiche $b
print_r($arr["b"]);
// affiche le chemin jusqu'à l'interpréteur CGI PHP (si PHP est utilisé en CGI)
// i.e. /usr/local/bin/php
echo $arr["_"];
// affiche la ligne de commande, s'il y en a une
print_r($arr["argv"]);
// affiche toutes les variables serveurs
print_r($arr["HTTP_SERVER_VARS"]);
// affiche toutes les clés disponibles dans les tableaux de variables
print_r(array_keys(get_defined_vars()));
?>
```

Voir aussi **get_defined_functions()**.

get_resource_type (PHP 4 >= 4.0.2)

Retourne le type de ressource

```
string get_resource_type (resource handle)
```

get_resource_type() retourne une chaîne représentant le type de ressources de *handle*. Si le paramètre n'est pas une ressource valide, une erreur est générée.

```
<?php
$c = mysql_connect();
echo get_resource_type($c)."\n";
// affiche : mysql link
// (lien mysql)
$fp = fopen("foo","w");
echo get_resource_type($fp)."\n";
```

```
// affiche : file
// (fichier)
$doc = new_xmldoc("1.0");
echo get_resource_type($doc->doc)."\n";
// affiche : domxml document
// (document domxml)
?>
```

intval (PHP 3, PHP 4 >= 4.0b1)

Retourne la valeur numérique (integer) de la variable.

```
int intval (mixed var [, int base])
```

intval() retourne la valeur numérique entière (integer) de la variable *var*, en convertissant la valeur dans la base spécifiée (par défaut en base 10).

var peut être de type scalaire. Vous ne pouvez pas utiliser la fonction **intval()** avec un tableau ou un objet.

Voir aussi **doubleval()**, **strval()**, **settype()** et Transtypage.

is_array (PHP 3, PHP 4 >= 4.0b1)

Détermine si une variable est un tableau.

```
bool is_array (mixed var)
```

is_array() renvoie la valeur TRUE si la variable *var* est un tableau, FALSE sinon.

Voir aussi **is_double()**, **is_float()**, **is_int()**, **is_integer()**, **is_real()**, **is_string()**, **is_long()**, et **is_object()**.

is_bool (PHP 4 >= 4.0b4)

Détermine si une variable est un tableau booléen

```
bool is_bool  (mixed var)
```

is_bool() retourne TRUE si *var* est un booléen.

Voir aussi **is_array()**, **is_double()**, **is_float()**, **is_int()**, **is_integer()**, **is_real()**, **is_string()**, **is_long()**, et **is_object()**.

is_double (PHP 3, PHP 4 >= 4.0b1)

Détermine si une variable est de type double.

```
bool is_double (mixed var)
```

is_double() renvoie TRUE si la variable *var* est du type "double", FALSE sinon.

Voir aussi **is_array()**, **is_bool()**, **is_float()**, **is_int()**, **is_integer()**, **is_real()**, **is_string()**, **is_long()**, et **is_object()**.

is_float (PHP 3, PHP 4 >= 4.0b1)

Détermine si une variable est de type float.

```
bool is_float (mixed var)
```

Cette fonction est un alias de la fonction **is_double()**.

Voir aussi **is_double()**, **is_bool()**, **is_real()**, **is_int()**, **is_integer()**, **is_string()**, **is_object()**, **is_array()**, et **is_long()**.

is_int (PHP 3, PHP 4 >= 4.0b1)

Détermine si une variable est de type integer.

```
bool is_int (mixed var)
```

is_int() est un alias de la fonction **is_long()**.

Voir aussi **is_bool()**, **is_double()**, **is_float()**, **is_integer()**, **is_string()**, **is_real()**, **is_object()**, **is_array()**, et **is_long()**.

is_integer (PHP 3, PHP 4 >= 4.0b1)

Détermine si une variable est de type int.

```
bool is_integer (mixed var)
```

Cette fonction est un alias de la fonction **is_long()**.

Voir aussi **is_bool()**, **is_double()**, **is_float()**, **is_int()**, **is_string()**, **is_real()**, **is_object()**, **is_array()**, et **is_long()**.

is_long (PHP 3, PHP 4 >= 4.0b1)

Détermine si une variable est de type integer.

```
bool is_long (mixed var)
```

is_long() renvoie TRUE si la variable *var* est du type entier long (long), FALSE sinon.

Voir aussi **is_bool()**, **is_double()**, **is_float()**, **is_int()**, **is_real()**, **is_string()**, **is_object()**, **is_array()**, et **is_integer()**.

is_null (PHP 4 >= 4.0.4)

Indique si une variable est NULL

```
bool is_null (mixed var)
```

is_null() retourne TRUE, si *var* est NULL, et FALSE.

Voir aussi : **is_bool()**, **is_double()**, **is_numeric()**, **is_float()**, **is_int()**, **is_real()**, **is_string()**, **is_object()**, **is_array()** et **is_integer()**.

is_numeric (PHP 4 >= 4.0RC1)

Détermine si une variable est un type numérique

```
bool is_numeric (mixed var)
```

is_numeric() retourne TRUE si *var* est un nombre, ou une chaîne numérique, ou FALSE sinon.

Voir aussi **is_bool()**, **is_double()**, **is_float()**, **is_int()**, **is_real()**, **is_string()**, **is_object()**, **is_array()**, et **is_integer()**.

is_object (PHP 3, PHP 4 >= 4.0b1)

Détermine si une variable est de type object.

```
bool is_object (mixed var)
```

is_object() renvoie TRUE si la variable *var* est un objet, FALSE sinon.

Voir aussi **is_bool()**, **is_long()**, **is_int()**, **is_integer()**, **is_float()**, **is_double()**, **is_real()**, **is_string()**, et **is_array()**.

is_real (PHP 3, PHP 4 >= 4.0b1)

Détermine si une variable est de type real.

```
bool is_real (mixed var)
```

Cette fonction est un alias de la fonction **is_double()**.

Voir aussi **is_bool()**, **is_long()**, **is_int()**, **is_integer()**, **is_float()**, **is_double()**, **is_object()**, **is_string()**, et **is_array()**.

is_resource (PHP 4 >= 4.0b4)

Détermine si une variable est une ressource

```
bool is_resource  (mixed var)
```

is_resource() retourne TRUE si la variable *var* est une ressource PHP, sinon FALSE.

Les ressources peuvent être des pointeurs de fichiers, des identifiants de résultats SQL, qui sont allouées et libérées en interne, par PHP, et qui peuvent demander un peut de nettoyage lorsqu'elle sont devenues inutiles, mais pas encore supprimées.

is_scalar (PHP 4 >= 4.0.5)

Indique si une variable est un scalaire

```
bool is_scalar  (mixed var)
```

is_scalar() retourne TRUE si la variable *var* est scalaire, et FALSE sinon.

Les variables scalaires sont celles qui contiennent des entiers, des nombres à virgules flottantes, des chaînes de caractères ou des booléens. Par exemple :

```
<?php
function show_var($var) {
    if (is_scalar($var))
        echo $var;
    else
        var_dump($var);
}
$pi = 3.1416;
$proteines = array("hemoglobine", "cytochrome c oxidase", "ferredoxine");
show_var($pi);
// affiche : 3.1416
show_var($proteines)
// affiche:
// array(3) {
//   [0]=>
//   string(10) "hemoglobine"
//   [1]=>
//   string(20) "cytochrome c oxidase"
//   [2]=>
//   string(10) "ferredoxine"
// }
?>
```

Note : is_scalar() a été ajoutée en version PHP 4.05.

Voir aussi : **is_bool()**, **is_double()**, **is_numeric()**, **is_float()**, **is_int()**, **is_real()**, **is_string()**, **is_object()**, **is_array()** et **is_integer()**.

is_string (PHP 3, PHP 4 >= 4.0b1)

Détermine si une variable est de type string.

```
bool is_string (mixed var)
```

is_string() renvoie TRUE si la variable *var* est du type chaine de caractères (string), FALSE sinon.

Voir aussi **is_long()**, **is_int()**, **is_integer()**, **is_float()**, **is_double()**, **is_real()**, **is_object()**, et **is_array()**.

isset (unknown)

Détermine si une variable est affectée.

```
int isset (mixed var)
```

isset() renvoie TRUE si la variable *var* est définie, FALSE sinon.

Si une variable a été désaffectée avec la fonction **unset()**, la fonction **isset()** renverra FALSE.

```
<?php
$a = "test";
echo isset ($a); // TRUE
unset($a);
echo isset ($a); // FALSE
?>
```

Voir aussi **empty()** et **unset()**.

print_r (PHP 4 >= 4.0b1)

Affiche des informations lisibles pour une variable.

```
void print_r (mixed expression)
```

Cette fonction affiche des informations à propos d'une variable, de manière à ce qu'elle soit lisible. Pour une chaîne, un entier ou un double, la valeur sera elle même sera affichée. Pour les tableaux, les valeurs seront présentées dans un format qui montre les clés et les valeurs. Une notation similaire est disponible pour les objets.

Comparer **print_r()** et **var_dump()**.

```
<?php
$a = array (1, 2, array ("a", "b", "c"));
print_r ($a);
?>
```

serialize (PHP 3>= 3.0.5, PHP 4 >= 4.0b1)

Linéarise une variable

```
string serialize (mixed value)
```

serialize() retourne une chaîne contenant une représentation linéaire de *value*, pour stockage.

C'est une technique pratique pour stocker ou passer des valeurs de PHP entre scripts, sans perdre ni leur structure, ni leur type.

Pour récupérer une variable linéarisée, et retrouver une variable, utilisez **unserialize()**. **serialize()** acceptent les types integer, double, string, array (multidimensionnels) et object (les propriétés des objets seront linéarisées, mais pas les méthodes).

Exemple 1. Exemple avec serialize()

```
<?php
// $session_data contient un tableau multi-dimensionnel , avec les
// informations de session de l'utilisateur courant. On utilise serialize()
// pour les stocker dans une base de données
$conn = odbc_connect ("webdb", "php", "chicken");
$stmt = odbc_prepare ($conn,
      "UPDATE sessions SET data = ? WHERE id = ?");
$sqldata = array (serialize($session_data), $PHP_AUTH_USER);
if (!odbc_execute ($stmt, &$sqldata)) {
    $stmt = odbc_prepare($conn,
     "INSERT INTO sessions (id, data) VALUES(?, ?)");
    if (!odbc_execute($stmt, &$sqldata)) {
    /* Grosse bourde! Souffre et potasse! */
    }
}
?>
```

settype (PHP 3, PHP 4 >= 4.0b1)

Affecte un type à une variable.

```
int settype (string var, string type)
```

settype() modifie le type de la variable *var* en *type*.

Les valeurs possibles pour le paramètre *type* sont :

- "integer"
- "double"
- "string"
- "array"
- "object"

settype() renvoie TRUE en cas de succès, FALSE sinon.

Voir aussi **gettype()**.

strval (PHP 3, PHP 4 >= 4.0b1)

Retourne la valeur de la variable, au format chaîne.

```
string strval (mixed var)
```

strval() retourne la valeur de la variable *var*, au format chaîne de caractères.

var peut être un scalaire. Vous ne pouvez pas utiliser la fonction **strval()** avec des tableaux ou des objets.

Voir aussi **doubleval()**, **intval()**, **settype()** et Transtypage.

unserialize (PHP 3>= 3.0.5, PHP 4 >= 4.0b1)

Crée une variable PHP à partir d'une valeur linéarisée

```
mixed unserialize (string str)
```

unserialize() prend une variable linéarisée (voir **serialize()**) et la convertit en variable PHP. La valeur convertie est retournée par la fonction, et peut être de type integer, double, string, array ou object. Les objets linéarisés perdent leurs méthodes.

Exemple 1. Exemple avec unserialize()

```
<?php
// Ici, on utilise unserialize() pour charger les données de sessions
// depuis la base de données, dans $session_data. Cet exemple complète
// celui fourni avec serialize().
$conn = odbc_connect ("webdb", "php", "chicken");
$stmt = odbc_prepare ($conn, "SELECT data FROM sessions WHERE id = ?");
$sqldata = array ($PHP_AUTH_USER);
if (!odbc_execute ($stmt, &$sqldata) || !odbc_fetch_into ($stmt, &$tmp)) {
    // si la préparation ou la lecture échoue, on crée un tableau vide
    $session_data = array();
} else {
```

```
    // les données sauvées sont dasn $tmp[0].
    $session_data = unserialize ($tmp[0]);
    if (!is_array ($session_data)) {
    // Erreur... initialisation à tableau vide
    $session_data = array();
    }
}
?>
```

unset (unknown)

Détruit une variable

```
void unset (mixed var [, mixed var [, ...]])
```

unset() détruit les variables *var*. Notez qu'en PHP 3, **unset()** retournait toujours TRUE (en fait, la valeur entière 1). **unset()** n'est plus une véritable fonction : c'est une structure du langage, ce qui fait qu'elle ne retourne pas de valeur. Lire la valeur retournée par **unset()** (dans une variable, par exemple), retourne une erreur d'analyse.

Exemple 1. Exemple avec unset()

```
<?php
// Destruction d'une seule variable
unset ($foo);
// Destruction d'un élément de tableau
unset ($bar['quux']);
// Destruction de plusieurs variables
unset ($foo1, $foo2, $foo3);
?>
```

Le comportement de **unset()** à l'intérieur d'une fonction peut varier suivant le type de variable que vous voulez détruire.

Si une variable globale est détruite avec **unset()** depuis une fonction, seule la variable locale sera détruite. Le variable globale gardera la valeur acquise avant l'appel à **unset()**.

```
<?php
function destroy_foo() {
    global $foo;
    unset($foo);
}
$foo = 'bar';
destroy_foo();
echo $foo;
?>
```

L'exemple ci dessus affichera :

bar

Si une variable qui est passée par référence est détruite à l'intérieur d'une fonction, seule la variable locale sera détruite. La variable globale conservera la dernière valeur qu'elle avait avant l'appel de **unset()**.

```
<?php
function foo(&$bar) {
```

```
    unset($bar);
    $bar = "bla";
}
$bar = 'truc';
echo "$bar\n";
foo($bar);
echo "$bar\n";
?>
```

L'exemple ci dessus va afficher :

```
truc
truc
```

Si une variable statique est détruite à l'intérieure d'une fonction **unset()** détruira la référence à la variable statique, plutôt que la variable statique elle même.

```
<?php
function foo() {
    static $a;
    $a++;
       echo "$a\n";
    unset($a);
}
foo();
foo();
foo();
?>
```

L'affichage du script ci-dessus donnera :

```
1
2
3
```

Si vous voulez détruire une variable globale, depuis une fonction, vous pouvez utiliser le tableau *$GLOBALS* :

```
<?php
function foo() {
    unset($GLOBALS['bar']);
}
$bar = "truc";
foo();
?>
```

Note : unset() est une structure du langage et non pas une fonction.

Voir aussi **isset()** et **empty()**.

var_dump (PHP 3>= 3.0.5, PHP 4 >= 4.0b1)

Dumpe les informations d’une variable.

```
void var_dump (mixed expression)
```

Cette fonction retourne les informations structurées d’une variable, y compris son type et sa valeur. Les tableaux sont explorés recursivement, avec des indentations, pour mettre en valeur leur structure.

Comparez **var_dump()** et **print_r()**.

```
<pre>
<?php
    $a = array (1, 2, array ("a", "b", "c"));
    var_dump ($a);
?>
</pre>
```

LXXXVII. WDDX

Ces fonctions doivent fonctionner avec l'aide de WDDX (http://www.openwddx.org/).

Pour utiliser WDDX, you devez installer la librairie EXPAT (qui est fournie avec la distribution d'Apache 1.3.7 ou plus récent), et recompiler PHP avec `--with-xml` et `--enable-wddx`.

Notez bien que toutes les fonctions qui enregistrent des données, utilisent le premier élément d'un tableau pour savoir si ce tableau doit être enregistré sous la forme d'un tableau, ou d'une structure. Si le premier élément a une clé de type chaîne, le tableau sera enregistré sous la forme d'une structure, et sinon, sous la forme d'un tableau.

Exemple 1. Enregistrer une valeur simple

```
<?php
print wddx_serialize_value("Exemple de paquet de PHP à WDDX ", "Paquet PHP");
?>
```

Cet exemple va produire le résultat suivant :

```
<wddxPacket version='0.9'><header comment='Paquet PHP' ><data>
<string>Exemple de paquet de PHP à WDDX</string></data></wddxPacket>
```

Exemple 2. Utilisation de paquets incrémentaux

```
<?php
$pi = 3.1415926;
$packet_id = wddx_packet_start("PHP");
wddx_add_vars($packet_id, "pi");
/* Supposons que $villes provient d'une base de données */
$cities = array("Paris", "Marseilles", "Lyon");
wddx_add_vars($packet_id, " villes ");
$packet = wddx_packet_end($packet_id);
print $packet;
?>
```

Cet exemple donnera :

```
<wddxPacket version='0.9'><header comment='PHP' ><data><struct>
<var name='pi'><number>3.1415926</number></var><var name='cities'>
<array length='3'><string>Paris</string><string>Marseilles</string>
<string>Lyon</string></array></var></struct></data></wddxPacket>
```

wddx_serialize_value (PHP 3>= 3.0.7, PHP 4)

Enregistrer une valeur dans un paquet WDDX

```
string wddx_serialize_value (mixed var [, string comment])
```

wddx_serialize_value() sert à créer un paquet WDDX à partir d'une seule valeur. Cette fonction prend la valeur de *var*, et un argument optionnel *comment* qui apparaîtra dans l'entête du paquet, et retourne un paquet WDDX.

wddx_serialize_vars (PHP 3>= 3.0.7, PHP 4)

Enregistrer plusieurs valeurs dans un paquet WDDX

```
string wddx_serialize_vars (mixed var_name [, mixed ...])
```

wddx_serialize_vars() sert à créer un paquet WDDX avec une structure qui contient la représentation des variables passées en arguments.

wddx_serialize_vars() prend un nombre variable d'arguments, chacun d'entre eux pouvant être une chaîne contenant le nom d'une variable, ou un tableau de chaîne de nom de variable, ou même d'autres tableaux.

Exemple 1. wddx_serialize_vars()

```
<?php
$a = 1;
$b = 5.5;
$c = array("bleu", "orange", "violet");
$d = "colors";
$clvars = array("c", "d");
print wddx_serialize_vars("a", "b", $clvars);
?>
```

L'exemple ci-dessus donnera : `<wddxPacket version='0.9'><header ><data><struct><var name='a'><number>1</number></var> <var name='b'><number>5.5</number></var><var name='c'><array length='3'> <string>bleu</string><string>orange</string><string>violet</string></array></var> <var name='d'><string>colors</string></var></struct></data></wddxPacket>`

wddx_packet_start (PHP 3>= 3.0.7, PHP 4)

Commencer un nouveau paquet WDDX avec une structure

```
resource wddx_packet_start ([string comment])
```

wddx_packet_start() sert à créer un nouveau paquet WDDX, pour pouvoir y faire des ajouts incrémentaux de variables. Cette fonction prend un argument optionel *comment* et retourne un identifiant de paquet, qui servira à d'autres fonctions. Elle va automatiquement créer une définition de structure dans le paquet, pour accueillir des variables.

wddx_packet_end (PHP 3>= 3.0.7, PHP 4)

Clos un paquet WDDX.

```
string wddx_packet_end (resource packet_id)
```

wddx_packet_end() clos un paquet WDDX repéré par son identifiant *packet_id*.

wddx_add_vars (PHP 3>= 3.0.7, PHP 4)

Ajouter des variables à un paquet WDDX.

```
wddx_add_vars (resource packet_id, mixed name_var [, mixed ...])
```

wddx_add_vars() sert à enregistrer les variables passées en argument, et les ajouter au paquet repéré par son identifiant *packet_id*. Les variables enregistrées sont spécifiées de la même façon que pour **wddx_serialize_vars()**.

wddx_deserialize (PHP 3>= 3.0.7, PHP 4)

Lire un paquet WDDX.

```
mixed wddx_deserialize (string packet)
```

wddx_deserialize() prend la chaîne *packet* et la lit. Cette fonction retourne un résultat qui peut être une chaîne, un nombre ou un tableau. Notez que les structures sont lues sous la forme de tableaux associatifs.

LXXXVIII. Analyseur syntaxique XML

Introduction

A propos de XML

Le langage XML (eXtensible Markup Language (Langage à Balises Etendu)) est un format structuré de données pour les échanges sur le web. C'est un standard défini par le consortium World Wide Web (W3C). Plus d'informations à propos du XML et des technologies afférentes sont accessibles (en anglais) http://www.w3.org/XML/.

Installation

Cette extension de PHP utilise expat, disponible à http://www.jclark.com/xml/. Le fichier Makefile livré avec expat ne construit pas par défaut de librairie : il faut utiliser la ligne suivante :

```
libexpat.a: $(OBJS)
ar -rc $@ $(OBJS)
ranlib $@
```

Les sources RPM de expat sont disponibles à http://www.guardian.no/~ssb/phpxml.html.

Notez que si vous utilisez Apache-1.3.7 ou plus récent, vous disposez déjà de la librairie expat. Configurez simplement PHP avec `--with-xml` (sans aucun autre information) et la librairie expat d'Apache sera automatiquement utilisée.

Sous UNIX, lancez la configuration de PHP avec l'option `--with-xml`, la librairie expat étant installée là où votre compilateur peut la trouver. Si vous compilez PHP comme module de PHP 1.3.9 ou plus récent, PHP utilisera automatiquement le module expat livré avec Apache. Il vous faudra peut être fixer les valeurs des variables d'environnement CPPFLAGS et LDFLAGS, si vous avez fait une installation exotique.

Compilez PHP. *Tada!* C'est fait !

A propos de cette extension :

Cette extension PHP supporte la librairie expat de James Clark sous PHP. Cela vous permettra d'analyser mais pas de valider les documents XML. Il supporte trois types de codage différents, disponibles aussi sous PHP: `US-ASCII`, `ISO-8859-1` et `UTF-8`. `UTF-16` n'est pas supporté.

Cette extension vous permet de créer des analyseurs XML puis de définir des *points d'entrée* pour chaque événement XML. Les analyseurs XML disposent de quelques paramétrages.

Les gestionnaires d'évènements XML sont:

Tableau 1. Les gestionnaires d'évènements XML

Fonction PHP de configuration du gestionnaire	Description de l'événement
xml_set_element_handler()	Un événement est généré à chaque fois que l'analyseur XML rencontre une balise de début ou de fin. Deux gestionnaires sont disponibles : un pour le début, et un pour la fin.
xml_set_character_data_handler()	"Character data" correspond grosso modo à tout ce qui n'est pas une balise XML, y compris les espaces entre les balises. Notez bien que l'analyseur XML n'ajoute ou n'efface aucun espace, et que c'est à l'application (c'est-à-dire vous) de décider de la signification de ces espaces.
xml_set_processing_instruction_handler()	Les programmeurs PHP sont habitués aux instructions exécutables (processing instructions ou PIs). `<?php ?>` est une instruction exécutable où php est appelé programme cible. Ces instructions sont gérées de manière spécifiques, (sauf le programme cible, qui est réservé à XML).

Fonction PHP de configuration du gestionnaire	Description de l'événement
xml_set_default_handler()	Tout ce qui n'a pas trouvé de gestionnaire est transmis au gestionnaire par défaut. Vous retrouverez par exemple, les déclarations de type de document dans ce gestionnaire.
xml_set_unparsed_entity_decl_handler()	Ce gestionnaire est appelé pour gérer les déclaration des entités non analysés.
xml_set_notation_decl_handler()	Ce gestionnaire est appelé pour gérer les notations.
xml_set_external_entity_ref_handler()	Ce gestionnaire est appelé lorsque l'analyseur XML trouve une référence à un fichier externe. Cela peut être un fichier, ou une URL. Reportez-vous à entité externe pour un exemple.

Problèmes de casse

Les fonctions de gestion des balises peuvent rencontrer des balises en minuscule, majuscule ou encore dans un mélange des deux. En XML, la procédure standard est d' "identifier les séquences de caractère qui ne sont pas reconnues comme majuscule, et de les remplacer par leur équivalent majuscule". En d'autres termes, XML met toutes lettres en majuscules.

Par défaut, tous les noms des éléments qui sont transmis aux fonctions de gestion sont mises en majuscule. Ce comportement est contrôlé par l'analyseur XML, et peut être lu et modifié avec les fonctions respectives **xml_parser_get_option()** et **xml_parser_set_option()**, respectivement.

Codes d'erreurs

Les constantes suivantes sont définies comme des codes d'erreurs XML : (retournée par **xml_parse()**)

XML_ERROR_NONE
XML_ERROR_NO_MEMORY
XML_ERROR_SYNTAX
XML_ERROR_NO_ELEMENTS
XML_ERROR_INVALID_TOKEN
XML_ERROR_UNCLOSED_TOKEN
XML_ERROR_PARTIAL_CHAR
XML_ERROR_TAG_MISMATCH
XML_ERROR_DUPLICATE_ATTRIBUTE
XML_ERROR_JUNK_AFTER_DOC_ELEMENT
XML_ERROR_PARAM_ENTITY_REF
XML_ERROR_UNDEFINED_ENTITY
XML_ERROR_RECURSIVE_ENTITY_REF
XML_ERROR_ASYNC_ENTITY
XML_ERROR_BAD_CHAR_REF
XML_ERROR_BINARY_ENTITY_REF
XML_ERROR_ATTRIBUTE_EXTERNAL_ENTITY_REF
XML_ERROR_MISPLACED_XML_PI
XML_ERROR_UNKNOWN_ENCODING
XML_ERROR_INCORRECT_ENCODING
XML_ERROR_UNCLOSED_CDATA_SECTION
XML_ERROR_EXTERNAL_ENTITY_HANDLING

Codage des caractères

L'extension XML de PHP supporte les caractères Unicode (http://www.unicode.org/) grâce à différents codages. Il y a deux types de codages de caractères : le codage à la source et le codage à la cible. PHP utilise le `UTF-8` comme représentation interne.

L'encodage à la source est effectué lors de l'analyse du fichier par XML. Lors de la création d'un analyseur XML), un type de codage à la source doit être spécifié (et il ne pourra plus être modifié jusqu'à la destruction de l'analyseur). Les codages supportés sont : `ISO-8859-1`, `US-ASCII` et `UTF-8`. Les deux derniers sont des codages à un seul octet, c'est-à-dire que les caractères sont représentés sur un seul octet. `UTF-8` peut représenter des caractères composés par un

nombre variable de bits (jusqu'à 21), allant de 1 à quatre octets. Le codage par défaut utilisé par PHP `ISO-8859-1`.

Le codage à la cible est effectué lorsque PHP transfert les données aux gestionnaires XML. Lorsqu'un analyseur est créé, le codage à la cible est spécifié de la même façon que le codage à la source, mais il peut être modifié à tout moment. Le codage à la cible affectera les balises, tout comme les données brutes, et les noms des instructions exécutables.

Si l'analyseur XML rencontre un caractère qu'il ne connaît pas (hors limite, par exemple), il retournera une erreur.

Si PHP rencontre un caractère dans le document XML analysé, qu'il ne peut pas représenter dans le codage à la cible choisi, le caractère sera remplacé par un point d'interrogation (cette attitude est susceptible de changer ultérieurement).

Quelques exemples

Voici une liste d'exemple de code PHP qui analyse un document XML.

Exemple de structure XML

Ce premier exemple affiche la structure de l'élément de début dans un document avec indentation.

Exemple 1. Afficher une structure XML

```
<?php
$file = "data.xml";
$depth = array();
function startElement($parser, $name, $attrs) {
    global $depth;
    for ($i = 0; $i < $depth[$parser]; $i++) {
        print "  ";
    }
    print "$name\n";
    $depth[$parser]++;
}
function endElement($parser, $name) {
    global $depth;
    $depth[$parser]--;
}
$xml_parser = xml_parser_create();
xml_set_element_handler($xml_parser, "startElement", "endElement");
if (!($fp = fopen($file, "r"))) {
    die("could not open XML input");
}
while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("XML error: %s at line %d",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}
xml_parser_free($xml_parser);
?>
```

XML Transtypage XML -> HTML

Exemple 2. XML Transtypage XML -> HTML

Cet exemple remplace les balises XML d'un document par des balises HTML. Les éléments inconnus seront ignorés. Bien entendu, cet exemple sera appliqué à un type précis de fichiers XML.

```
<?php
$file = "data.xml";
```

```
$map_array = array(
    "BOLD"     => "B",
    "EMPHASIS" => "I",
    "LITERAL"  => "TT"
);
function startElement($parser, $name, $attrs) {
    global $map_array;
    if ($htmltag = $map_array[$name]) {
        print "<$htmltag>";
    }
}
function endElement($parser, $name) {
    global $map_array;
    if ($htmltag = $map_array[$name]) {
        print "</$htmltag>";
    }
}
function characterData($parser, $data) {
    print $data;
}
$xml_parser = xml_parser_create();
// use case-folding so we are sure to find the tag in $map_array
xml_parser_set_option($xml_parser, XML_OPTION_CASE_FOLDING, TRUE);
xml_set_element_handler($xml_parser, "startElement", "endElement");
xml_set_character_data_handler($xml_parser, "characterData");
if (!($fp = fopen($file, "r"))) {
    die("could not open XML input");
}
while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("XML error: %s at line %d",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}
xml_parser_free($xml_parser);
?>
```

XML Entité externe

Cet exemple exploite les références externes de XML : il est possible d'utiliser un gestionnaire d'entité externe pour inclure et analyser les documents, tous comme les instructions exécutables peuvent servir à inclure et analyser d'autres documents, et aussi fournir une indication de confiance (voir plus bas).

Le document XML qui est utilisé dans cet exemple est fourni plus loin dans l'exemple (`xmltest.xml` et `xmltest2.xml`).

Exemple 3. Entité externe

```
<?php
$file = "xmltest.xml";
function trustedFile($file) {
    // only trust local files owned by ourselves
    if (!eregi("^([a-z]+)://", $file)
        && fileowner($file) == getmyuid()) {
            return TRUE;
    }
    return FALSE;
}
function startElement($parser, $name, $attribs) {
    print "&lt;<font color=\"#0000cc\">$name</font>";
    if (sizeof($attribs)) {
        while (list($k, $v) = each($attribs)) {
```

```
            print " <font color=\"#009900\">$k</font>=\"<font
                   color=\"#990000\">$v</font>\"";
        }
    }
    print "&gt;";
}
function endElement($parser, $name) {
    print "&lt;/<font color=\"#0000cc\">$name</font>&gt;";
}
function characterData($parser, $data) {
    print "<B>$data</B>";
}
function PIHandler($parser, $target, $data) {
    switch (strtolower($target)) {
        case "php":
            global $parser_file;
            // If the parsed document is "trusted", we say it is safe
            // to execute PHP code inside it.  If not, display the code
            // instead.
            if (trustedFile($parser_file[$parser])) {
                eval($data);
            } else {
                printf("Code PHP peu sÛr : <B>%s</B>",
                        htmlspecialchars($data));
            }
            break;
    }
}
function defaultHandler($parser, $data) {
    if (substr($data, 0, 1) == "&" && substr($data, -1, 1) == ";") {
        printf('<font color="#aa00aa">%s</font>',
                htmlspecialchars($data));
    } else {
        printf('<font size="-1">%s</font>',
                htmlspecialchars($data));
    }
}
function externalEntityRefHandler($parser, $openEntityNames, $base, $systemId,
                                  $publicId) {
    if ($systemId) {
        if (!list($parser, $fp) = new_xml_parser($systemId)) {
            printf("Could not open entity %s at %s\n", $openEntityNames,
                   $systemId);
            return FALSE;
        }
        while ($data = fread($fp, 4096)) {
            if (!xml_parse($parser, $data, feof($fp))) {
                printf("XML error: %s at line %d while parsing entity %s\n",
                       xml_error_string(xml_get_error_code($parser)),
                       xml_get_current_line_number($parser), $openEntityNames);
                xml_parser_free($parser);
                return FALSE;
            }
        }
        xml_parser_free($parser);
        return TRUE;
    }
    return FALSE;
}
function new_xml_parser($file) {
    global $parser_file;
    $xml_parser = xml_parser_create();
    xml_parser_set_option($xml_parser, XML_OPTION_CASE_FOLDING, 1);
    xml_set_element_handler($xml_parser, "startElement", "endElement");
    xml_set_character_data_handler($xml_parser, "characterData");
    xml_set_processing_instruction_handler($xml_parser, "PIHandler");
```

```
    xml_set_default_handler($xml_parser, "defaultHandler");
    xml_set_external_entity_ref_handler($xml_parser, "externalEntityRefHandler");
    if (!($fp = @fopen($file, "r"))) {
        return FALSE;
    }
    if (!is_array($parser_file)) {
        settype($parser_file, "array");
    }
    $parser_file[$xml_parser] = $file;
    return array($xml_parser, $fp);
}
if (!(list($xml_parser, $fp) = new_xml_parser($file))) {
    die("could not open XML input");
}
print "<pre>";
while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("XML error: %s at line %d\n",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}
print "</pre>";
print "parse complete\n";
xml_parser_free($xml_parser);
?>
```

Exemple 4. xmltest.xml

```
<?xml version='1.0'?>
<!DOCTYPE chapter SYSTEM "/just/a/test.dtd" [
<!ENTITY plainEntity "FOO entity">
<!ENTITY systemEntity SYSTEM "xmltest2.xml">
]>
<chapter>
 <TITLE>Title &plainEntity;</TITLE>
 <para>
  <informaltable>
   <tgroup cols="3">
    <tbody>
     <row><entry>a1</entry><entry morerows="1">b1</entry><entry>c1</entry></row>
     <row><entry>a2</entry><entry>c2</entry></row>
     <row><entry>a3</entry><entry>b3</entry><entry>c3</entry></row>
    </tbody>
   </tgroup>
  </informaltable>
 </para>
 &systemEntity;
 <sect1 id="about">
  <title>About this Document</title>
  <para>
   <!- this is a comment -->
   <?php print 'Hi!  This is PHP version '.phpversion(); ?>
  </para>
 </sect1>
</chapter>
```

Ce fichier est inclus depuis `xmltest.xml`:

Exemple 5. xmltest2.xml

```
<?xml version="1.0" ?>
<!DOCTYPE foo [
<!ENTITY testEnt "test entity">
?>
<foo>
   <element attrib="value"?>
   &testEnt;
   <?php print "This is some more PHP code being executed."; ?>
</foo>
```

xml_parser_create (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Création d'un analyseur XML.

```
resource xml_parser_create ([string encoding])
```

encoding (optional)

Le codage de caractère de l'analyseur : les codages suivants sont supportés :
`ISO-8859-1` (par défaut)
`US-ASCII`
`UTF-8`

xml_parser_create() crée un analyseur XML et retourne une référence sur cet analyseur pour qu'il puisse être utilisé ultérieurement par d'autres fonctions XML. **xml_parser_create()** retourne `FALSE` en cas d'erreur.

xml_set_object (PHP 4 >= 4.0b4)

Utilise un analyseur XML à l'intérieur d'un objet.

```
void xml_set_object (resource parser, object &object)
```

xml_set_object() rend l'analyseur *parser* utilisable depuis un objet. Toutes les méthodes de callback, affectées par **xml_set_element_handler()**, seront les méthodes de cet objet.

```
<?php
class xml  {
var $parser;
function xml() {
    $this->parser = xml_parser_create();
    xml_set_object($this->parser,&$this);
    xml_set_element_handler($this->parser,"tag_open","tag_close");
    xml_set_character_data_handler($this->parser,"cdata");
}
function parse($data) {
    xml_parse($this->parser,$data);
}
function tag_open($parser,$tag,$attributes) {
    var_dump($parser,$tag,$attributes);
}
function cdata($parser,$cdata) {
    var_dump($parser,$cdata);
}
function tag_close($parser,$tag) {
    var_dump($parser,$tag);
}
} // Fin de la classe xml
$xml_parser = new xml();
$xml_parser->parse("<A ID=\"bonjour\">PHP</?>");
?>
```

xml_set_element_handler (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Affecte les gestionnaires de début et de fin.

```
int xml_set_element_handler  (resource parser, string startElementHandler, string
endElementHandler)
```

xml_set_element_handler() affecte les gestionnaires de début et de fin de l'analyseur XML *parser*. *startElementHandler* et *endElementHandler* sont des chaînes qui contiennent les noms de fonctions qui existent lorsque **xml_parse()** est appelé pour créer *parser*.

La fonction *startElementHandler* doit accepter trois paramètres:

```
startElementHandler (resource parser, string name, array attribs)
```

parser

Le premier paramètre, *parser*, est une référence sur l'analyseur XML qui appelle cette fonction.

name

Le deuxième paramètre, *name*, contient le nom de l'élément qui a provoqué l'appel du gestionnaire. Si l'analyseur gère la casse, cet élément sera en majuscule.

attribs

Le troisième paramètre, *attribs*, contient un tableau associatif avec les attributs de l'éléments (s'il en existe). Les clés de ce tableau seront les noms des attributs, et les valeurs seront les valeurs correspondantes des attributs. Les noms des attributs seront mis en majuscule si l'analyseur gère la casse. Les valeurs des attributs seront intouchées.

L'ordre original des attributs peut être retrouvé en passant en revue le tableau *attribs*, avec la fonction **each()**. La première clé sera la première clé du tableau.

La fonction *endElementHandler* doit accepter deux paramètres:

```
endElementHandler (resource parser, string name)
```

parser

Le premier paramètre, *parser*, est une référence sur l'analyseur XML qui appelle cette fonction.

name

Le second paramètre, *name*, contient le nom de l'élément qui a provoqué l'appel du gestionnaire. Si l'analyseur gère la casse, cet élément sera en majuscule.

Si un gestionnaire reçoit une chaîne vide, ou FALSE, c'est qu'il est en train d'être désactivé.

xml_set_element_handler() retourne TRUE si le gestionnaire est actif, et FALSE sinon, ou si *parser* n'est pas un analyseur.

Il n'est pas pour l'instant possible d'utiliser des objets pour servir de gestionnaire. Reportez-vous à **xml_set_object()** pour utiliser l'analyseur XML depuis un objet.

xml_set_character_data_handler (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Affecte les gestionnaires de caractère bruts.

```
int xml_set_character_data_handler  (resource parser, string handler)
```

Affecte les gestionnaires de début et de fin de l'analyseur XML *parser*. *handler* est une chaîne qui contient le nom d'une fonction qui existe lorsque **xml_parse()** est appelé pour créer *parser*.

La fonction *handler* doit accepter deux paramètres:

```
handler (resource parser, string data)
```

parser

Le premier paramètre, *parser*, est une référence sur l'analyseur XML qui appelle cette fonction.

data

Le second paramètre, *data*, contient les caractères sous la forme d'une chaîne.

Si un gestionnaire reçoit une chaîne vide ou FALSE, c'est qu'il est en train d'être désactivé.

xml_set_character_data_handler() retourne TRUE si le gestionnaire est actif, et FALSE sinon, ou si *parser* n'est pas un analyseur.

Il n'est pas pour l'instant possible d'utiliser des objets pour servir de gestionnaire. Reportez-vous à **xml_set_object()** pour utiliser l'analyseur XML depuis un objet.

xml_set_processing_instruction_handler (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Affecte les gestionnaires d'instructions exécutables.

```
int xml_set_processing_instruction_handler  (resource parser, string handler)
```

Affecte les gestionnaires d'instructions exécutables de l'analyseur XML *parser*. *handler* est une chaîne qui contient le nom d'une fonction qui existe lorsque **xml_parse()** est appelé pour créer *parser*.

Une instruction exécutable a la forme suivante :

```
<?
       target
       data
```

Vous pouvez mettre du code PHP entre ces balises, mais soyez conscient d'une des limitations des instructions exécutables de XML : la balise de fin d'instruction exécutable (?>) ne peut être échappée, ce qui fait que cette séquence NE DOIT JAMAIS apparaître dans le code PHP placé dans le document PHP. Si un tel texte apparaît, la balise de fin d'instruction exécutable sera reconnue, et le reste du code sera considéré comme des données brutes (et donc, pas exécutées).

La fonction *handler* doit accepter trois paramètres:

```
handler (resource parser, string target, string data)
```

parser

Le premier paramètre, *parser*, est une référence sur l'analyseur XML qui appelle cette fonction.

target

Le second paramètre, *target*, contient l'application cible.

data

Le troisième paramètre, *data*, contient le code sous la forme d'une chaîne.

Si un gestionnaire reçoit une chaîne vide, ou FALSE, c'est qu'il est en train d'être désactivé.

xml_set_processing_instruction_handler() retourne TRUE si le gestionnaire est actif, et FALSE sinon, ou si *parser* n'est pas un analyseur.

Il n'est pas pour l'instant possible d'utiliser des objets pour servir de gestionnaire. Reportez-vous à **xml_set_object()** pour utiliser l'analyseur XML depuis un objet.

xml_set_default_handler (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Affecte le gestionnaire par défaut.

```
int xml_set_default_handler  (resource parser, string handler)
```

Affecte le gestionnaire par défaut de l'analyseur XML *parser*. *handler* est une chaîne qui contient le nom d'une fonction qui existe lorsque **xml_parse()** est appelé pour créer *parser*.

La fonction *handler* doit accepter deux paramètres:

```
handler (resource parser, string data)
```

parser

Le premier paramètre, *parser*, est une référence sur l'analyseur XML qui appelle cette fonction.

data

Le second paramètre, *data*, contient les caractères sous la forme d'une chaîne. Cela peut être une déclaration XML, un type de document, une entité ou d'autre données pour qui aucun gestionnaire n'est prévu.

Si un gestionnaire reçoit une chaîne vide ou FALSE, c'est qu'il est en train d'être désactivé.

xml_set_default_handler() retourne TRUE si le gestionnaire est actif, et FALSE sinon, ou si *parser* n'est pas un analyseur.

Il n'est pas pour l'instant possible d'utiliser des objets pour servir de gestionnaire. Reportez-vous à **xml_set_object()** pour utiliser l'analyseur XML depuis un objet.

xml_set_unparsed_entity_decl_handler (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Affecte les gestionnaires d'entité non déclaré.

```
int xml_set_unparsed_entity_decl_handler  (resource parser, string handler)
```

Affecte les gestionnaires d'entité non déclaré de l'analyseur XML *parser*. *handler* est une chaîne qui contient le nom d'une fonction qui existe lorsque **xml_parse()** est appelé pour créer *parser*.

Ce gestionnaire sera appelé si l'analyseur XML rencontre une déclaration d'entité externe avec une déclaration de NDATA, comme suit :

```
<!ENTITY name {publicId | systemId}
        NDATA notationName
```

Reportez-vous à la section des spécifications XML 1.0 (http://www.w3.org/TR/1998/REC-xml-19980210#sec-external-ent) pour connaître les notations des entités externes.

La fonction *handler* doit accepter six paramètres:

```
handler (resource parser, string entityName, string base, string systemId, string
publicId, string notationName)
```

parser

Le premier paramètre, *parser*, est une référence sur l'analyseur XML qui appelle cette fonction.

entityName

Le nom de l'entité qui va être définie

base

La meilleure base de résolution de l'identifiant système de cette entité externe. Actuellement, ce paramètre est toujours une chaîne vide.

systemId

Identifiant système pour cet entité externe.

publicId

Identifiant public pour cet entité externe.

notationName

Nom de la notation de cette entité. (Voir **xml_set_notation_decl_handler()**).

Si un gestionnaire reçoit une chaîne vide ou FALSE, c'est qu'il est en train d'être désactivé.

xml_set_unparsed_entity_decl_handler() retourne TRUE si le gestionnaire est actif, et FALSE sinon, ou si *parser* n'est pas un analyseur.

Il n'est pas pour l'instant possible d'utiliser des objets pour servir de gestionnaire. Reportez-vous à **xml_set_object()** pour utiliser l'analyseur XML depuis un objet.

xml_set_notation_decl_handler (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Affecte les gestionnaires de notation.

```
int xml_set_notation_decl_handler  (resource parser, string handler)
```

Affecte les gestionnaires de début et de fin de l'analyseur XML *parser*. *handler* est une chaîne qui contient le nom d'une fonction qui existe lorsque **xml_parse()** est appelé pour créer *parser*.

Une notation est une partie du DTD du document, qui a le format suivant :

```
<!NOTATION name
{ systemId | publicId?>
```

Reportez-vous à la section des spécifications XML 1.0 (http://www.w3.org/TR/1998/REC-xml-19980210#sec-external-ent) pour connaître les notations des entités externes.

La fonction *handler* doit accepter cinq paramètres:

```
handler (resource parser, string notationName, string base, string systemId, string
publicId)
```

parser

Le premier paramètre, *parser*, est une référence sur l'analyseur XML qui appelle cette fonction.

notationName

Le nom de la notation,*name*, comme précisé dans le format de notation ci-dessus.

base

La meilleure base de résolution de l'identifiant système de cette entité externe. Actuellement, ce paramètre est toujours une chaîne vide.

systemId

Identifiant système pour cet entité externe.

publicId

Identifiant public pour cet entité externe.

Si un gestionnaire reçoit une chaîne vide ou FALSE, c'est qu'il est en train d'être désactivé.

xml_set_notation_decl_handler() retourne TRUE si le gestionnaire est actif, et FALSE sinon ou si *parser* n'est pas un analyseur.

Il n'est pas pour l'instant possible d'utiliser des objets pour servir de gestionnaires. Reportez-vous à **xml_set_object()** pour utiliser l'analyseur XML depuis un objet.

xml_set_external_entity_ref_handler (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Modifie le gestionnaire de référence externes.

```
int xml_set_external_entity_ref_handler  (resource parser, string handler)
```

Fixe le gestionnaire d'entité externe de l'analyseur XML *parser*. *handler* et *endElementHandler* sont des chaînes qui contiennent les noms de fonction qui existent lorsque **xml_parse()** est appelé pour créer le *parser*.

La fonction *handler* doit accepter 5 paramètres, et retourner un entier. Si la valeur retourné par le gestionnaire est FALSE (comme par exemple si aucune valeur n'est retournée), l'analyseur XML s'arrêtera, et la fonction **xml_get_error_code()** retournera XML_ERROR_EXTERNAL_ENTITY_HANDLING.

```
int handler (resource parser, string openEntityNames, string base, string systemId, string
publicId)
```

parser

Le premier paramètre,*parser*, est une référence sur l'analyseur XML qui appelle cette fonction.

openEntityNames

Le deuxième paramètre, *openEntityNames*, est la liste de noms d'entité, séparés par des espaces. Ces entités sont accessibles à l'analyse par cet entité (y compris le nom de l'entité referencé).

base

La meilleure base de résolution de l'identifiant système de cet entité externe. Actuellement, ce paramètre est toujours une chaîne vide.

systemId

Identifiant système pour cet entité externe.

publicId

Le cinquième paramètre, *publicId*, est l'identifiant public, comme spécifié dans la déclaration d'entité, ou un chaîne vide, si aucune déclaration n'a été spécifiée. L'espace dans l'identifiant public sera normalisé comme spécifié dans les spécifications XML.

Si un gestionnaire reçoit une chaîne vide, ou FALSE, c'est qu'il est en train d'être désactivé.

xml_set_external_entity_ref_handler() retourne TRUE si le gestionnaire est actif, et FALSE sinon ou si *parser* n'est pas un analyseur.

Il n'est pas pour l'instant possible d'utiliser des objets pour servir de gestionnaire. Reportez-vous à **xml_set_object()** pour utiliser l'analyseur XML depuis un objet.

xml_parse (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Commence l'analyse d'un fichier XML.

```
int xml_parse (resource parser, string data [, int isFinal])
```

parser

une référence sur l'analyseur XML à utiliser.

data

Une partie des données à analyser. Un document peut être analysé morceau par morceau, en appelant **xml_parse()** plusieurs fois, tant que le paramètre *isFinal* est mis à TRUE pour le dernier morceau.

isFinal (optional)

S'il vaut TRUE, *data* est la dernière partie à analyser.

Lorsqu'un document XML est analysé, les gestionnaires d'événements sont appelés aussi souvent que nécessaire, et retournent TRUE ou FALSE.

TRUE est retourné lorsque l'analyse a été concluante, et FALSE en cas d'échec, ou si *parser* n'est pas un analyseur valide. Lors d'un échec d'analyse, la cause de l'erreur peut être obtenue grâce aux fonctions **xml_get_error_code()**, **xml_error_string()**, **xml_get_current_line_number()**, **xml_get_current_column_number()** et **xml_get_current_byte_index()**.

xml_get_error_code (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Retourne le nombre courant de colonne d'un analyseur XML.

```
int xml_get_error_code (resource parser)
```

xml_get_error_code() retourne FALSE si *parser* n'est pas valide, ou sinon, retourne le numéro de colonne courante de la ligne courante de l'analyseur, qui correspond à la position d'analyse courante de l'analyseur XML.

xml_error_string (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Lit le message d'erreur de l'analyseur XML.

```
string xml_error_string (int code)
```

code

Un message d'erreur, issu de **xml_get_error_code()**.

xml_error_string() retourne la chaîne avec un message textuel, décrivant l'erreur *code*, ou FALSE si aucune description n'a été trouvée.

xml_get_current_line_number (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Retourne le numéro de ligne courant d'un analyseur XML.

```
int xml_get_current_line_number (resource parser)
```

parser

Une référence sur un analyseur XML valide.

xml_get_current_line_number() retourne FALSE si *parser* n'est pas valide, ou sinon, retourne le numéro de la ligne en cours d'analyse.

xml_get_current_column_number (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Retourne le nombre courant de colonne d'un analyseur XML.

```
int xml_get_current_column_number (resource parser)
```

parser

Une référence sur un analyseur XML valide.

xml_get_current_column_number() retourne FALSE si *parser* n'est pas valide, ou sinon, retourne le numéro de colonne courante de la ligne courante de l'analyseur, qui correspond à la position d'analyse courante de l'analyseur XML.

xml_get_current_byte_index (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Retourne l'index de l'octet courant d'un analyseur XML.

```
int xml_get_current_byte_index (resource parser)
```

parser

Une référence sur un analyseur XML valide.

xml_get_current_byte_index() retourne FALSE si *parser* n'est pas valide, ou sinon, retourne l'index de l'octet d'analyse courante de l'analyseur XML.

xml_parse_into_struct (PHP 3>= 3.0.8, PHP 4 >= 4.0b1)

Analyse une structure XML

```
int xml_parse_into_struct (resource parser, string data, array &values, array &index)
```

xml_parse_into_struct() analyse le fichier XML *data*, et le place dans deux tableaux : le premier *index* contient des pointeurs sur la position des valeurs correspondantes dans le tableau *values* array. Ces deux paramètres sont passés par références.

Ci-dessous, vous trouverez un exemple qui illustre la structure des deux tableaux générés par la fonction. On utilise une balise simple `note`, placée dans une autre balise `para`. On analyse le tout, et on affiche la structure générée :

```
<?php
$simple = "<para><note>simple note</note></para>";
$p = xml_parser_create();
xml_parse_into_struct($p,$simple,$vals,$index);
xml_parser_free($p);
echo "Tableau d'index\n";
print_r($index);
echo "\nTableau de valeurs\n";
print_r($vals);
?>
```

Lors de l'éxécution du code, l'affichage sera :

```
Tableau d'index
Array
(
    [PARA] => Array
        (
            [0] => 0
            [1] => 2
        )
    [NOTE] => Array
        (
            [0] => 1
        )
)
Tableau de valeurs
Array
(
    [0] => Array
        (
            [tag] => PARA
            [type] => open
            [level] => 1
        )
    [1] => Array
        (
            [tag] => NOTE
            [type] => complete
            [level] => 2
            [value] => simple note
        )
    [2] => Array
        (
            [tag] => PARA
            [type] => close
            [level] => 1
        )
)
```

L'analyse événementielle (comme celle de expat), peut se révéler complexe lorsque le document XML est complexe. **xml_parse_into_struct()** ne génère pas d'objet de type DOM, mais il génère plutôt des structures qui peuvent être parcourues à la façon d'un arbre. Considérons le fichier suivant, qui représente une petite base de données XML :

Exemple 1. moldb.xml - Petite base de données moléculaire

```
<?xml version="1.0"?>
<moldb>
    <molecule>
        <name>Alanine</name>
        <symbol>ala</symbol>
        <code>A</code>
        <type>hydrophobic</type>
    </molecule>
    <molecule>
        <name>Lysine</name>
        <symbol>lys</symbol>
        <code>K</code>
        <type>charged</type>
    </molecule>
</moldb>
```

Et maitenant, un code qui analyse le document, et génère les objet ad hoc :

Exemple 2. parsemoldb.php - analyse moldb.xml et crée un tableau d'objet moléculaires

```
<?php
class AminoAcid {
    var $name;  // nom de l'amino acide
    var $symbol;    // symbole en trois lettres
    var $code;  // code en une lettre
    var $type;  // hydrophobe, chargé ou neutre
    function AminoAcid ($aa) {
        foreach ($aa as $k->$v)
            $this->$k = $aa[$k];
    }
}
function readDatabase($filename) {
    // read the xml database of aminoacids
    $data = implode("",file($filename));
    $parser = xml_parser_create();
    xml_parser_set_option($parser,XML_OPTION_CASE_FOLDING,0);
    xml_parser_set_option($parser,XML_OPTION_SKIP_WHITE,1);
    xml_parse_into_struct($parser,$data,&$values,&$tags);
    xml_parser_free($parser);
    // parcourt les structures
    foreach ($tags as $key->$val) {
        if ($key == "molecule") {
            $molranges = $val;
            // chaque paire contigue sont les définitions supérieures
            // et inférieures de la molécule
            for ($i=0; $i < count($molranges); $i+=2) {
                    $offset = $molranges[$i] + 1;
                $len = $molranges[$i + 1] - $offset;
                $tdb[] = parseMol(array_slice($values, $offset, $len));
            }
        } else {
            continue;
        }
    }
    return $tdb;
}
function parseMol($mvalues) {
    for ($i=0; $i < count($mvalues); $i++)
        $mol[$mvalues[$i]["tag"]] = $mvalues[$i]["value"];
    return new AminoAcid($mol);
}
$db = readDatabase("moldb.xml");
echo "** Database of AminoAcid objects:\n";
```

```
print_r($db);
?>
```

Après exécution de `parsemoldb.php`, la variable `$db` contient un tableau d'objets `AminoAcid`, et l'affichage le confirme :

```
** Database of AminoAcid objects:
Array
(
    [0] => aminoacid Object
        (
            [name] => Alanine
            [symbol] => ala
            [code] => A
            [type] => hydrophobic
        )
    [1] => aminoacid Object
        (
            [name] => Lysine
            [symbol] => lys
            [code] => K
            [type] => charged
        )
)
```

xml_parser_free (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Détruit un analyseur XML.

```
boolean xml_parser_free (resource parser)
```

parser

Une référence sur un analyseur XML.

xml_parser_free() retourne `FALSE` si *`parser`* n'est pas une référence valide, ou sinon, détruit l'analyseur et retourne `TRUE`.

xml_parser_set_option (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Affecte les options d'un analyseur XML.

```
int xml_parser_set_option (resource parser, int option, mixed value)
```

parser

Une référence vers un analyseur XML.

option

L'option à modifier. Voir ci-dessous :

value

La nouvelle valeur de l'option.

xml_parser_set_option() retourne FALSE si *parser* n'est pas une référence valide sur un analyseur XML, ou si l'option n'a pas pu être modifiée. Sinon, l'option est effectivement modifiée, et la fonction retourne TRUE.

Les options suivantes sont disponibles :

Tableau 1. options d'analyseur XML

Option	Type de données	Description
XML_OPTION_CASE_FOLDING	entier	Contrôle la gestion de la casse des balises de cet analyseur XML. Par défaut, activé.
XML_OPTION_TARGET_ENCODING	string	Modifie le codage à la cible utilisé par cet analyseur XML. Par défaut, c'est celui qui a été spécifié lors de l'appel de **xml_parser_create()**. Les codages supportés sont ISO-8859-1, US-ASCII et UTF-8.

xml_parser_get_option (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Lit les options d'un analyseur XML.

```
mixed xml_parser_get_option  (resource parser, int option)
```

parser

Une référence sur un analyseur XML valide.

option

L'option demandée. Reportez-vous à **xml_parser_set_option()** pour avoir la liste des options disponibles.

xml_parser_get_option() retourne FALSE si *parser* n'est pas valide, ou sinon, retourne la valeur de l'option demandée.

Reportez-vous à **xml_parser_set_option()** pour avoir la liste des options disponibles.

utf8_decode (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Convertit une chaîne UTF-8 en ISO-8859.

```
string utf8_decode (string data)
```

utf8_decode() décode la chaîne *data*, en supposant qu'elle est au format UTF-8, et la convertit au format ISO-8859-1.

Voir aussi **utf8_encode()**pour plus de détails sur le codage UTF-8.

utf8_encode (PHP 3>= 3.0.6, PHP 4 >= 4.0b1)

Convertit une chaîne ISO-8859-1 en UTF-8.

```
string utf8_encode (string data)
```

utf8_encode() code la chaîne data au format `UTF-8`, et retourne la version codée. `UTF-8` est un mécanisme standardisé utilisé par Unicode pour coder les caractère de grande taille dans des flots d'octets. `UTF-8` est transparent pour les caractères ASCII, il est auto-synchronisé (c'est à dire qu'un programme peut toujours savoir dans un flot d'octet où un caractère commence), et peut être utilisé pour faire des comparaisons de chaînes standard, comme pour le tri. PHP utilise l'`UTF-8` pour coder les caractères jusqu'à 4 octets comme ceci :

Tableau 1. UTF-8 encoding

octets	bits	représentation
1	7	0bbbbbbb
2	11	110bbbbb 10bbbbbb
3	16	1110bbbb 10bbbbbb 10bbbbbb
4	21	11110bbb 10bbbbbb 10bbbbbb 10bbbbbb

Chaque *b* représente un bit qui peut être utilisé pour enregistrer un caractère.

LXXXIX. XSLT

Avertissement

Ce module est *EXPERIMENTAL*. Cela signifie que le comportement de ces fonctions, leurs noms et concrètement, TOUT ce qui est documenté ici peut changer dans un futur proche, SANS PREAVIS! Soyez-en conscient, et utiliser ce module à vos risques et périls.

Introduction

A propos de XSLT et Sablotron

XSLT (Extensible Stylesheet Language (XSL) Transformations) est un langage de transformation des documents XML en d'autres documents XML. C'est un standard défini par le consortium World Wide Web (W3C). Les informations sur le XLST et ses technologies sont disponibles à http://www.w3.org/TR/xslt.

Installation

Cette extension utilise Sablotron et expat, qui sont toutes les deux disponibles à http://www.gingerall.com/. Les sources comme les exécutables sont proposés.

Sous UNIX, lancez **configure** avec l'option `--with-sablot`. La librairie Sablotron doit être installée là où le compilateur peut la trouver.

A propos de Sablotron

Cette extension PHP implémente le support de Sablotron, par Ginger Alliance. Cette librairie vous permet de transformer des documents XML en d'autres documents XML, mais aussi en HTML ou encore n'importe quel format à balise. Elle fournit un mécanisme basique et portable de templates, séparant le contenu de l'interface d'un site web.

xslt_closelog (PHP 4 >= 4.0.3)

Efface le fichier d'historique

```
boolean xslt_closelog (resource xh)
```

xh

Une référence valide sur un analyseur XSLT.

xslt_closelog() retourne FALSE si *xh* n'est pas un analyse XSLT valide, ou bien si la fermeture du fichier d'historique a échoué. Sinon, retourne TRUE.

xslt_create (PHP 4 >= 4.0.3)

Crée un nouvel analyseur XSLT.

```
resource xslt_create (void)
```

xslt_create() retourne un identifiant d'analyseur XSLT. Il sera nécessaire pour les appels ultérieurs aux fonctions.

xslt_errno (PHP 4 >= 4.0.3)

Retourne le numéro d'erreur courant

```
int xslt_errno ([resource xh])
```

xslt_errno() le numéro courant d'erreur, pour l'analyseur *xh*. Si *xh* n'est pas fourni, le dernier numéro d'erreur est retourné.

xslt_error (PHP 4 >= 4.0.3)

Retourne le message d'erreur courant

```
mixed xslt_error ([resource xh])
```

xslt_errno() le message d'erreur courant, pour l'analyseur *xh*. Si *xh* n'est pas fourni, le dernier numéro d'erreur est retourné.

xslt_fetch_result (PHP 4 >= 4.0.3)

Lit un résultat

```
string xslt_fetch_result (resource xh [, string result_name])
```

xslt_fetch_result() lit le résultat disponible dans le buffer de l'analyseur *xh*. Si *result_name* n'est pas fourni, le résultat "/_result" sera lu.

xslt_free (PHP 4 >= 4.0.3)

Détruit un analyseur XSLT

```
void xslt_free (resource xh)
```

xslt_free() détruit l'analyseur XSLT *xh*.

xslt_openlog (PHP 4 >= 4.0.3)

Modifie le fichier d'historique

```
boolean xslt_openlog (resource xh, string logfile [, int loglevel])
```

xslt_openlog() remplace le fichier d'historique courant par *logfile*, pour l'analyseur XSLT *xh*.

xslt_output_begintransform (PHP 4 >= 4.0.3)

Commence la transformation XSLT

```
void xslt_output_begintransform  (string xslt_filename)
```

xslt_output_begintransform() commence à retourner la transformée de vos données. En l'appel de **xslt_output_begintransform()** et celui de **xslt_output_endtransform()**, toutes les données seront transformées par la feuille de style XSLT *xslt_filename*.

Exemple 1. Transformation avec une feuille de style XSLT (avec DOM-XML)

```
<?php
$xsl_file = "article.xsl";
xslt_output_begintransform($xsl_file);
$doc = new_xmldoc('1.0');
$article = $doc->new_root('article');
$article->new_child('title', 'Histoire du Tyrol méridional');
$article->new_child('author', 'Sterling Hughes');
$article->new_child('body', 'Juste après la première guerre mondiale, l'Italie
                             obtiend le Tyrol méridional aux dépends de l'Autriche.
                             Et depuis cette époque, rien d'interessant n'est arrivé.');
echo $doc->dumpmem();
xslt_output_endtransform();
?>
```

xslt_output_endtransform (PHP 4 >= 4.0.3)

Termine une transformation XSLT

```
void xslt_output_endtransform  (void)
```

xslt_output_endtransform() termine la transformation commencée avec **xslt_output_begintransform()**. Vous devez appeler **xslt_output_endtransform()** pour pouvoir accéder aux résultats.

xslt_process (PHP 4 >= 4.0.3)

Transforme des données XML

```
boolean xslt_process  (string xsl_data, string xml_data, string result)
```

xslt_process() prend la chaîne string *xsl_data* comme feuille de style XSLT, et des données XML dans *xml_data*. Le résultat de la transformation sera placé dans *result*. **xslt_process()** retourne TRUE en cas de succès, et FALSE sinon. Vous pourrez lire les erreurs survenues grâce aux fonctions **xslt_errno()** et **xslt_error()**.

Exemple 1. Utilisation de xslt_process() pour transformer trois

```
<?php
$xslData = '<xsl:stylesheet
  version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="article">
    <table border="1" cellpadding="2" cellspacing="1">
        <tr>
            <td width="20%">
              
            </title>
            <td width="80%">
                <h2><xsl:value-of select="title"></h2>
                <h3><xsl:value-of select="author"></h3>
                <br>
                <xsl:value-of select="body">
            </td>
        </tr>
    </table>
</xsl:template>
</xsl:stylesheet>';
$xmlData = '
<?xml version="1.0">
<article>
    <title>Learning German</title>
    <author>Sterling Hughes</author>
    <body>
      Essential phrases:
      <br>
      <br>
      K&#246;nnen Sie mir sagen, wo die Toilette ist?<br>
      Ein grosses Bier, bitte!<br>
      Noch eins, bitte.<br>
    </body>
</article>';
if (xslt_process($xslData, $xmlData, $result))
{
    echo "Voici un brillant article sur l'apprentissage du ";
    echo " français: ";
    echo "<br>\n<br>";
    echo $result;
}
else
{
    echo "Une erreur est survenue durant le traitement XSL...\n";
    echo "\tErreur numéro : " . xslt_errno() . "\n";
    echo "\tMessage d'erreur : " . xslt_error() . "\n";
    exit;
}
?>
```

xslt_run (PHP 4 >= 4.0.3)

Applique une feuille de style à un fichier

```
boolean xslt_run (resource xh, string xslt_file, string xml_data_file [, string result [,
array xslt_params [, array xslt_args]]])
```

xslt_run() applique au fichier *xml_data_file* la feuille de style *xslt_file*. La feuille de style peut utiliser les paramètres optionnels *xslt_params* et l'analyseur XSLT est démarré avec *xslt_args*. Le résultat est placé dans le buffer nommé *result* (par défaut, dans "/_result").

xslt_set_sax_handler (PHP 4 >= 4.0.3)

Modifie les gestionnaires SAX de l'analyseur XSLT

```
boolean xslt_set_sax_handler (resource xh, array handlers)
```

xslt_set_sax_handler() remplace les gestionnaires SAX de l'analyseur XSLT *xh* par *handlers*.

xslt_transform (PHP 4 >= 4.0.3)

Exécute une transformation XSLT

```
boolean xslt_transform  (string xsl, string xml, string result, string params, string args,
string resultBuffer)
```

xslt_transform() fournit une interface avec les API avancées de sablotron, sans vous obliger à utiliser les ressources.

XC. YAZ

Introduction

Cette extension offre à PHP l'interface avec les produits YAZ, qui implémentent le protocole Z39.50. Avec cette extension, vous pouvez facilement implémenter un client Z39.50 qui analyse ou scanne des serveurs Z39.50 en paralléle.

YAZ est disponible à http://www.indexdata.dk/yaz/. Vous pouvez trouvez des informations, des scripts d'exemples, etc... pour cette extension à http://www.indexdata.dk/phpyaz/.

Le module masque l'essentiel de la complexité de Z39.50, ce qui le rend très facile à utiliser. Il supporte les connexions peristantes de manière similaire à celle supportés par les serveurs SQL : cela signifie qu'une connexion est partagée entre plusieurs scripts PHP, ce qui évite les opérations de connexions.

Installation

Compilez YAZ et installez le. Compilez PHP avec vos modules et ajoutez l'option `--with-yaz`. Les instructions sont :

```
gunzip -c yaz-1.6.tar.gz|tar xf -
gunzip -c php-4.0.X.tar.gz|tar xf -
cd yaz-1.6
./configure --prefix=/usr
make
make install
cd ../php-4.0.X
./configure --with-yaz=/usr/bin
make
make install
```

Exemple

PHP/YAZ conserve les connexions aux serveurs. Un entier positif représente l'ID d'une connexion particulière.

Le script ci-dessous montre comment effectuer une recherche paralléle. Lorsqu'il est appelé sans paramètre, ce script affiche la requête. Sinon, il effectue la recherche sur les serveurs.

Exemple 1. Recherche paralléle utilisant YAZ

```
<?php
$num_hosts = count ($host);
if (empty($term) || count($host) == 0) {
    echo '<form method="get">
    <input type="checkbox"
    name="host[]" value="bagel.indexdata.dk/gils">
        GILS test
    <input type="checkbox"
    name="host[]" value="localhost:9999/Default">
        local test
    <input type="checkbox" checked="1"
    name="host[]" value="z3950.bell-labs.com/books">
        BELL Labs Library
    <br>
    RPN Query:
    <input type="text" size="30" name="term">
    <input type="submit" name="action" value="Search">
    ';
} else {
    echo 'Vous avez recherché '.htmlspecialchars($term).'<br>';
    for ($i = 0; $i > $num_hosts; $i++) {
```

```
        $id[] = yaz_connect($host[$i]);
        yaz_syntax($id[$i],"sutrs");
        yaz_search($id[$i],"rpn",$term);
    }
    yaz_wait();
    for ($i = 0; $i < $num_hosts; $i++) {
        echo '<hr>'.$host[$i].":";
        $error = yaz_error($id[$i]);
        if (!empty($error)) {
            echo "Erreur: $error";
        } else {
            $hits = yaz_hits($id[$i]);
            echo "Nombre de résultats : $hits";
        }
        echo '<dl>';
        for ($p = 1; $p <= 10; $p++) {
            $rec = yaz_record($id[$i],$p,"string");
            if (empty($rec)) continue;
            echo "<dt><B>$p</B></dt><dd>";
            echo ereg_replace("\n", "<br>\n",$rec);
            echo "</dd>";
        }
        echo '</dl>';
    }
}
?>
```

yaz_addinfo (PHP 4 >= 4.0.1)

Retourne plus de détails après une erreur

```
int yaz_addinfo (int id)
```

yaz_addinfo() retourne plus de détails après la dernière erreur survenue. Une chaîne vide est retournée si la dernière opération a été réussie, ou bien si aucune autre information n'est disponible.

yaz_close (PHP 4 >= 4.0.1)

Ferme une connexion YAZ

```
int yaz_close (int id)
```

yaz_close() ferme une connexion à un hôte YAZ. L'application ne pourra plus utiliser l'identifiant de connexion *id*. *id* est un identifiant d'hôte, retourné par **yaz_connect()**.

yaz_connect (PHP 4 >= 4.0.1)

Prépare une connexion à un hôte YAZ

```
int yaz_connect (string zurl [, string authentication])
```

yaz_connect() retourne un identifiant positif en cas de succès, et FALSE sinon.

yaz_connect() prépare une connexion à un serveur Z39.50. *zurl* est de la forme "host[:port][/database]". Si port est omis, 210 est utilisé. Si database est omis, default est utilisé. **yaz_connect()** n'est pas bloquante, et ne tente pas d'établir une socket. En fait, elle ne fait que préparer la connexion pour exécution ultérieure par **yaz_wait()**.

yaz_errno (PHP 4 >= 4.0.1)

Retourne le numéro d'erreur

```
int yaz_errno (int id)
```

yaz_errno() retourne le numéro d'erreur de la dernière requête. Une valeur positive est retournée si le serveur a retourné un diagnostic. La valeur 0 est retournée si aucune erreur n'est survenue. Une valeur négative indique une erreur sans diagnostic (connexion perdue,...).

yaz_errno() doit être appelée après chaque requête. (après la fin de **yaz_wait()**), pour savoir si la transaction a réussi ou échoué.

yaz_error (PHP 4 >= 4.0.1)

Retourne une description de l'erreur

```
string yaz_error (int id)
```

yaz_error() retourne un message d'erreur pour la dernière requête. Une chaîne vide est retournée si la dernière requête a réussi.

yaz_error() retourne un message en anglais, qui correspond au numéro d'erreur retourné par **yaz_errno()**.

yaz_hits (PHP 4 >= 4.0.1)

Retourne le nombre de résultat de la dernière recherche

```
int yaz_hits (int id)
```

yaz_hits() retourne le nombre de résultat de la dernière recherche.

yaz_element (PHP 4 >= 4.0.1)

Spécifie le type d'éléments à lire

```
int yaz_element (int id, string elementset)
```

yaz_element() est à utiliser en conjonction avec **yaz_search()** et **yaz_present()** pour spécifier le type d'éléments à lire. La majorité des serveurs supporte F (full, tous), et B (brief, bref).

yaz_element() retourne TRUE en cas de succès, et FALSE sinon.

yaz_database (PHP 4 >= 4.0.6)

Spécifie la base d'une session

```
int yaz_database (int id, string databases)
```

yaz_database() spécifie *databases*, la ou les bases utilisées lors des recherches, lectures, etc, en remplaçant celles spécifiées lors de la fonction **yaz_connect()**. Pour indiquer plusieurs bases de données, séparez les noms par des +.

yaz_database() vous permet d'utiliser différents jeux de bases durant une session.

yaz_database() retourne TRUE en cas de succès, ou FALSE en cas d'erreur.

yaz_present (PHP 4 >= 4.0.5)

Prépare à la lecture (Z39.50 present).

```
int yaz_present (void)
```

yaz_present() prépare PHP à la lecture des résultats, après une recherche. **yaz_range()** doit être appelée avant celle-ci pour spécifier la plage de résultat à lire.

yaz_range (PHP 4 >= 4.0.1)

Spécifie le nombre maximal de résultat à lire

```
int yaz_range (int id, int start, int number)
```

yaz_range() est utilisée conjointement à **yaz_search()**, pour spécifier le nombre maximal *number* de résultat à lire, ainsi que la position de début de lecture avec *start*. Si **yaz_range()** n'est pas utilisée, *start* vaudra 1 et *number* vaudra 10.

yaz_range() retourne TRUE en cas de succès; FALSE en cas d'erreur.

yaz_record (PHP 4 >= 4.0.1)

Retourne un résultat

```
int yaz_record (int id, int pos, string type)
```

yaz_record() retourne un résultatà la position *pos*, ou une chaîicirc;ne vide si aucun résultat n'est disponible à la position *pos*.

yaz_record() recherche une ligne dans le résultat, à la position spécifiée. Si aucune ligne n'existe à la position donnée, une chaîicirc;ne vide est retournée. L'argument *type* spécifie la forme du résultat retourné : si *type* vaut "string", la ligne est retournée sous la forme d'une chaîicirc;ne, prête à l'affichage. Si *type* vaut "array", la ligne sera retournée sous la forme d'un tableau structuré.

yaz_search (PHP 4 >= 4.0.1)

Prépares une recherche

```
int yaz_search (int id, string type, string query)
```

yaz_search() prépare une recherche sur le serveur identifié par *id*. *type* représente le type de requête : seul RPN est supporté actuellement, et dans ce cas, le troisième argument est un préfixe de notation de requête utilisé par YAZ. Comme pour **yaz_connect()**, **yaz_search()** n'est pas bloquante, et ne fait que préparer la recherche pour exécution ultérieure, avec **yaz_wait()**.

Les requêtes RPN sont des représentation textuelles des requêtes de type Type-1, comme définit dans le standard Z39.50. Cependant, dans la représentation textuelle utilisée par YAZ, une notation à préfixage est utilisée, c'est-à-dire que l'opérateur précédent l'opérande. La chaîne de requête est une séquence de mots réservés, où les espaces sont ignorés, à moins qu'ils n'ai été mis entre guillemets doubles. Les mots réservés qui commencent par un arobase (@) sont considérés comme des opérateurs et traités comme tels.

Tableau 1. Opérateurs RPN

Syntaxe	Description
`@and query1 query2`	intersection des requêtes query1 et query2
`@or query1 query2`	union des requêtes query1 et query2
`@not query1 query2`	requêtes "query1 et non(query2)"
`@set name`	nomme le résultat
`@attrset set query`	spécifie le jeu d'attributs de la requête. Cette construction n'est autorisée qu'une seule fois, au début d'une requête.
`@attr set type=value query`	Applique les attributs à une requête. Le type et la valeur sont des entiers indiquant les types et valeurs des attributs, dans cet ordre. Le jeu, si fourni, spécifie le jeu d'attribut utilisé.

Les requêtes suivantes illustrent des requêtes valides :

```
ordinateur
```

Recherche les documents qui contiennent le mot "ordinateur". Aucun attribut n'est spécifié.

```
"serveur rapide"
```

Recherche les documents qui contiennent les mots "serveur rapide"

```
@attr 1=4 php
```

L'attribut est de type 1 (Bib-1 use), sa valeur est 4 (Title, titre) : cette requête recherche les documents où le mot "php" est dans le titre.

```
@attrset gils @and @attr 1=4 php @attr 1=1003 "Rasmus Lerdorf"
```

Cette requête utilise tout le jeu d'attributs GILS. Elle recherche les documents dont le titre contient "php", et qui contiennent le nom "Rasmus Lerdorf" comme auteur.

yaz_syntax (PHP 4 >= 4.0.1)

Spécifie la syntaxe de lecture des lignes

```
int yaz_syntax (int id, string syntax)
```

yaz_syntax() est utilisée conjointement avec **yaz_search()** pour spécifier la méthode de lecture des lignes. La syntaxe est spécifiée comme un OID (Identifiant d'Objet), en notation brute, séparée par des points (i.e. `1.2.840.10003.5.10`), ou bien avec une des valeurs prédéfinies : sutrs, usmarc, grs1, xml, etc... **yaz_syntax()** doit être utilisée en conjonction avec **yaz_search()** et **yaz_present()** pour spécifier la méthode de lecture des résultats.

yaz_scan (PHP 4 >= 4.0.5)

Prépare un scan

```
int yaz_scan (int id, string type, string startterm [, array flags])
```

yaz_scan() prépare une requête "Z39.50 Scan Request". *`id`* spécifie l'hôte cible. Le point de départ est donné avec *`startterm`*. La forme de spécification du point de départ est donné par *`type`*. Actuellement, le type `rpn` est supporté. Le paramètre optionnel *`flags`* donne des informations supplémentaires pour contrôler le comportement de la requête de scan. Actuellement, trois index sont lus dans ce paramètre : `number` (nombre de termes requis), `position` (position préférée du terme) et `stepSize` (taille du pas préférée). Pour reéllement envoyer la requête de recherche à l'hôte, et recevoir la réponse, **yaz_wait()** doit être appelée. A la fin de **yaz_wait()**, **yaz_error()** et **yaz_scan_result()** auront les résultats.

La syntaxe de *`startterm`* est similaire aux requêtes RPN, décrites dans **yaz_search()**. *`startterm`* est constitué de zéro ou plus spécifications, avec les opérateurs `@attr`, suivi par exactement un token.

Exemple 1. Fonction PHP qui scanne les titres

```
function scan_titles($id, $starterm) {
  yaz_scan($id,"rpn", "@attr 1=4 " . $starterm);
  yaz_wait();
  $errno = yaz_errno($id);
  if ($errno == 0) {
    $ar = yaz_scan_result($id,&$options);
    echo 'Scan ok; ';
    $ar = yaz_scan_result($id, &$options);
    while(list($key,$val)=each($options)) {
       echo "$key = $val  ";
    }
    echo '<br><table><tr><td>';
    while(list($key,list($k, $term, $tcount))=each($ar)) {
      if (empty($k)) continue;
      echo "<tr><td>$term</td><td>";
```

```
            echo $tcount;
            echo "</td></tr>";
          }
          echo '</table>';
        } else {
          echo "Echec du scan. Erreur: " . yaz_error($id) . "<br>";
        }
      }
```

yaz_scan_result (PHP 4 >= 4.0.5)

Retourne le résultat d'un scan

```
array yaz_scan_result (int id [, array & result])
```

yaz_scan_result() retourne un tableau contenant les termes recu de l'hôte *id*, lors de la dernière requête de scan. Le tableau commence à l'index 0. Chaque valeur est une paire, où le premier élément est le terme, et le second est un compte de résultat. Si *result* est fourni, il sera rempli avec les informations générales de la requête de scan : `number` (numbre d'entrée retournées), `stepsize` (taille du pas), `position` (position du terme), `status` (Statut du Scan).

yaz_ccl_conf (PHP 4 >= 4.0.5)

Configure l'analyseur CCL

```
int yaz_ccl_conf (int id, array config)
```

yaz_ccl_conf() configure l'analyseur CCL de requete de l'hôte *id*, avec les définitions de points d'accès (CCL qualifiers) et leur équivalent en RPN. Pour cabler une requête spécifique vers un appel RPN, utilisez **yaz_ccl_parse()**. Chaque index du tableau *config* est un nom de champs CCL et la valeur correspondante contient une chaîne spécifiant le code RPN. Ce code est une séquence de paires "attribut-type, attribute-value". Les "attribut-type" et "attribut-value" sont séparé par le signe égal (=). Chaque paire est séparé par un espace ("=").

Exemple 1. Exemple de configuration CCL

Dans l'exemple ci-dessous, l'analyseur CCL est configuré pour supporter trois champs CCL : `ti`, `au` et `isbn`. Chaque champs correspond à leur équivalent équivalent BIB-1. On suppose que chaque variable `$id` est un hôte de destination.

```
<?php
      $field["ti"] = "1=4";
      $field["au"] = "1=1";
      $field["isbn"] = "1=7";
      yaz_ccl_conf($id,$field);
?<
```

yaz_ccl_parse (PHP 4 >= 4.0.5)

Appelle l'analyseur CCL

```
int yaz_ccl_parse (int id, string query, array &result)
```

yaz_ccl_parse() appelle l'analyseyr CCL. Il convertit une requête CCL FIND en une requête RPN qui peut être passée à **yaz_search()** pour effectuer une recherche. Pour définir un champs CCL valide, utilisez la fonction **yaz_ccl_conf()** avant celle-ci. Si la requête *query* a pu être convertie en RPN, **yaz_ccl_parse()** retourne TRUE, et l'index rpn du tableau *result* contient une requête RPN valide. Si la requête n'a pas pu être convertie, (pour n'importe quelle raison, comme syntaxe invalide, champs inconnu...), **yaz_ccl_parse()** retourne FALSE. Trois index sont alors créés dans le tableau de résultat : errorcode (code d'erreur CCL, un entier), errorstring (message d'erreur CCL), et errorpos position estimée de l'erreur dans la requête (entier, position en nombre de caractères).

yaz_itemorder (PHP 4 >= 4.0.5)

Prépare une requête Z39.50 Item Order avec le package ILL-Request

```
int yaz_itemorder (array args)
```

yaz_itemorder() prépare une requête de type "Extended Services" en utilisant le "Profile" avec "Use of Z39.50 Item Order Extended Service to Transport ILL (Profile/1)" (Note du Traducteur : mailez moi de l'aide!). Reportez-vous ici (http://www.nlc-bnc.ca/iso/ill/stanprf.htm) ou aux spécification (http://www.nlc-bnc.ca/iso/ill/document/standard/z-ill-1a.pdf). Le paramètre *args* doit être un tableau associatif, contenant les informations "Item Order" à envoyer. L'index du tableau est le nom ASN.1 correspondant au tag path. Par exemple, le numéro ISBN sous l'Item-ID est la clé item-id,ISBN.

Les paramètres de ILL-Request sont :

protocol-version-num
transaction-id,initial-requester-id,person-or-institution-symbol,person
transaction-id,initial-requester-id,person-or-institution-symbol,institution
transaction-id,initial-requester-id,name-of-person-or-institution,name-of-person
transaction-id,initial-requester-id,name-of-person-or-institution,name-of-institution
transaction-id,transaction-group-qualifier
transaction-id,transaction-qualifier
transaction-id,sub-transaction-qualifier
service-date-time,this,date
service-date-time,this,time
service-date-time,original,date
service-date-time,original,time
requester-id,person-or-institution-symbol,person
requester-id,person-or-institution-symbol,institution
requester-id,name-of-person-or-institution,name-of-person
requester-id,name-of-person-or-institution,name-of-institution
responder-id,person-or-institution-symbol,person
responder-id,person-or-institution-symbol,institution
responder-id,name-of-person-or-institution,name-of-person
responder-id,name-of-person-or-institution,name-of-institution
transaction-type
delivery-address,postal-address,name-of-person-or-institution,name-of-person
delivery-address,postal-address,name-of-person-or-institution,name-of-institution
delivery-address,postal-address,extended-postal-delivery-address
delivery-address,postal-address,street-and-number
delivery-address,postal-address,post-office-box
delivery-address,postal-address,city
delivery-address,postal-address,region
delivery-address,postal-address,country
delivery-address,postal-address,postal-code
delivery-address,electronic-address,telecom-service-identifier
delivery-address,electronic-address,telecom-service-addreess
billing-address,postal-address,name-of-person-or-institution,name-of-person
billing-address,postal-address,name-of-person-or-institution,name-of-institution
billing-address,postal-address,extended-postal-delivery-address

billing-address,postal-address,street-and-number
billing-address,postal-address,post-office-box
billing-address,postal-address,city
billing-address,postal-address,region
billing-address,postal-address,country
billing-address,postal-address,postal-code
billing-address,electronic-address,telecom-service-identifier
billing-address,electronic-address,telecom-service-addreess
ill-service-type
requester-optional-messages,can-send-RECEIVED
requester-optional-messages,can-send-RETURNED
requester-optional-messages,requester-SHIPPED
requester-optional-messages,requester-CHECKED-IN
search-type,level-of-service
search-type,need-before-date
search-type,expiry-date
search-type,expiry-flag
place-on-hold
client-id,client-name
client-id,client-status
client-id,client-identifier
item-id,item-type
item-id,call-number
item-id,author
item-id,title
item-id,sub-title
item-id,sponsoring-body
item-id,place-of-publication
item-id,publisher
item-id,series-title-number
item-id,volume-issue
item-id,edition
item-id,publication-date
item-id,publication-date-of-component
item-id,author-of-article
item-id,title-of-article
item-id,pagination
item-id,ISBN
item-id,ISSN
item-id,additional-no-letters
item-id,verification-reference-source
copyright-complicance
retry-flag
forward-flag
requester-note
forward-note

Il y a quelques paramètres du package Extended Services Request et ItemOrder :

package-name
user-id
contact-name
contact-phone
contact-email
itemorder-item

yaz_wait (PHP 4 >= 4.0.1)

Attend l’exécution d’une requête

```
int yaz_wait (void)
```

yaz_wait() exécute les requêtes préparée par les fonctions **yaz_search()**, **yaz_present()**, **yaz_scan()** et **yaz_itemorder()**. **yaz_wait()** se termine lorsque tous les hôtes ont terminé leurs requêtes (éventuellement en cas d’erreur).

XCI. NIS

NIS (feu Yellow Pages / Pages jaunes) permet la gestion par le réseau de fichiers d'administration importants (tel un fichier de mot de passe). Pour plus d'informations, reportez vous au manuel NIS, ou à Introduction to YP/NIS (http://www.desy.de/~sieversm/ypdoku/ypdoku/ypdoku.html)Introduction to YP/NIS (en anglais). Il existe un livre en anglais Managing NFS and NIS (http://www.oreilly.com/catalog/nfs/noframes.html)" par Hal Stern.

Pour ajouter ces fonctionnalitées, vous devez compiler PHP avec l'option `--with-yp`(PHP 3) ou `--enable-yp`(PHP 4).

yp_get_default_domain (PHP 3>= 3.0.7, PHP 4 >= 4.0b1)

Retourne le domaine NIS par défaut.

```
int yp_get_default_domain (void)
```

yp_get_default_domain() retourne le nom de domaine NIS par défaut. Ce nom de domaine peut être utilisé pour les futurs appels NIS.

Un domaine NIS peut être décrit comme un regroupement de cartes NIS. Tous les hôtes qui ont besoin d'informations, s'attachent à un domaine. Référez vous aux documents cités en début de document pour plus de détails.

Exemple 1. Exemple avec le domaine par défaut

```
<?php
    $domain = yp_get_default_domain();
    echo "Le domaine par défaut est : " . $domain;
?>
```

yp_order (PHP 3>= 3.0.7, PHP 4 >= 4.0b1)

Retourne le numéro d'ordre d'une carte.

```
int yp_order (string domain, string map)
```

yp_order() retourne le numéro d'ordre d'une carte ou FALSE.

Exemple 1. Exemple d'ordre NIS

```
<?php
    $number = yp_order($domain,$mapname);
    echo "Le numéro d'ordre de cette carte est : " . $order;
?>
```

Voir aussi **yp-get-default-domain()**.

yp_master (PHP 3>= 3.0.7, PHP 4 >= 4.0b1)

Retourne le nom de la machine maître pour une carte.

```
string yp_master (string domain, string map)
```

yp_master() retourne le nom de la machine maître d'une carte.

Exemple 1. Exemple de maître NIS

```
<?php
    $number = yp_master($domain, $mapname);
    echo "Master for this map is: " . $master;
?>
```

Voir aussi **yp-get-default-domain()**.

yp_match (PHP 3>= 3.0.7, PHP 4 >= 4.0b1)

Retourne la ligne associée.

```
string yp_match (string domain, string map, string key)
```

yp_match() retourne la valeur associée à la clé passée en argument, pour la carte spécifiée, ou FALSE. La clé doit exister et être exacte.

Exemple 1. Exemple de recherche NIS

```
<?php
    $entry = yp_match($domain, "passwd.byname", "joe");
    echo "La valeur trouvée est: " . $entry;
?>
```

Dans le cas présent, ce pourrait être: joe:##joe:11111:100:Joe User:/home/j/joe:/usr/local/bin/bash

Voir aussi **yp-get-default-domain()**

yp_first (PHP 3>= 3.0.7, PHP 4 >= 4.0b1)

Retourne le premier couple (clé ; valeur) d'une carte donnée.

```
array yp_first (string domain, string map)
```

yp_first() retourne le premier couple (clé ; valeur) d'une carte donnée, ou FALSE.

Exemple 1. Exemple avec yp_first()

```
<?php
$entree = yp_first($domain, "passwd.byname");
$cle = $entry ["key"];
$valeur = $entry ["value"];
echo "La première entrée de cette carte est " . $key
     . " et sa valeur est " . $entry[$key];
?>
```

Voir aussi **yp-get-default-domain()**

yp_next (PHP 3>= 3.0.7, PHP 4 >= 4.0b1)

Retourne le couple (clé ; valeur) suivant d'une carte donnée.

```
array yp_next (string domain, string map, string key)
```

yp_next() retourne le couple (clé ; valeur) suivant la clé donnée d'une carte donnée ou FALSE.

Exemple 1. Exemple avec yp_next()

```
<?php
    $entry = yp_next($domain, "passwd.byname", "joe");
    if(!$entry) {
echo "Plus d’autres entrées.\n";
        }
    $key = key($entry);
    echo "L’entrée suivante après "joe" a la clé  " . $key
          . " et la valeur " . $entry[$key];
?>
```

Voir aussi **yp-get-default-domain()**.

XCII. Zip (décompression)

Ce module utilise les fonctions de la librairie ZZIPlib (http://zziplib.sourceforge.net/), créée par Guido Draheim pour lire de manière transparente des archives compressées Zip, et les fichiers qu'elles contiennent.

Notez que ZZIPlib ne fournit qu'une partie des fonctions utilisant l'algorithme de compression ZIP : elle ne permet que de lire les fichiers Zip. Un utilitaire Zip est nécessaire pour créer ces archives, vous ne pourrez pas le faire en PHP.

Le support de Zip par PHP n'est pas activé par défaut. Vous devez utiliser l'option --with-zip lorsque vous compilez PHP pour l'activer. Ce module requiert par ailleurs la librairie ZZIPlib version >= 0.10.6.

> **Note :** Le support de Zip pour les versions antérieures à PHP 4.0.7 est expérimental. Cette section décrit l'extension Zip telle qu'elle existe en PHP 4.0.7 et plus récent.

Exemple d'utilisation

Cet exemple ouvre un fichier ZIP, lit chaque fichier de l'archive, et affiche son contenu. Le script `test2.php` utilisé dans cet exemple est un des fichiers de test de la distribution source de ZZIPlib.

Exemple 1. Exemple d'utilisation de l'extension Zip

```
<?php
$zip = zip_open("/tmp/test2.zip");
if ($zip) {
    while ($zip_entry = zip_read($zip)) {
        echo "Name:               " . zip_entry_name($zip_entry) . "\n";
        echo "Actual Filesize:    " . zip_entry_filesize($zip_entry) . "\n";
        echo "Compressed Size:    " . zip_entry_compressedsize($zip_entry) . "\n";
        echo "Compression Method: " . zip_entry_compressionmethod($zip_entry) . "\n";
///
        if (zip_entry_open($zip, $zip_entry, "r")) {
            echo "File Contents:\n";
            $buf = zip_entry_read($zip_entry, zip_entry_filesize($zip_entry));
            echo "$buf\n";
            zip_entry_close($zip_entry);
        }
        echo "\n";
    }
    zip_close($zip);
}
?>
```

zip_close (PHP 4 CVS only)

Ferme une archive Zip

```
void zip_close (resource zip)
```

zip_close() ferme l'archive zip *zip*. Le paramètre *zip* doit être une archive zip, créée par la fonction **zip_open()**.

Cette fonction ne retourne pas de valeur.

Voir aussi **zip_open()** et **zip_read()**.

zip_entry_close (PHP 4 CVS only)

Ferme un élément d'archive

```
void zip_entry_close (resource zip_entry)
```

zip_entry_close() ferme l'élément d'archive *zip_entry*. Le paramètre *zip_entry* doit être un élément d'archive valide, créé par la fonction **zip_entry_open()**.

zip_entry_close() ne retourne pas de valeur.

Voir aussi **zip_entry_open()** et **zip_entry_read()**.

zip_entry_compressedsize (PHP 4 CVS only)

Lit la taille compressée d'un dossier

```
int zip_entry_compressedsize (resource zip_entry)
```

zip_entry_compressedsize() retourne la taille compressée de l'élément d'archive *zip_entry*. Le paramètre *zip_entry* doit être un élément d'archive valide, créé par la fonction **zip_entry_open()**.

Voir aussi **zip_open()** et **zip_read()**.

zip_entry_compressionmethod (PHP 4 CVS only)

Retourne la méthode de compression d'un dossier

```
string zip_entry_compressionmethod (resource zip_entry)
```

zip_entry_compressionmethod() la méthode de compression de l'élément d'archive spécifié *zip_entry*. Le paramètre *zip_entry* doit être un élément d'archive valide, créé par la fonction **zip_entry_open()**.

Voir aussi **zip_open()** et **zip_read()**.

zip_entry_filesize (PHP 4 CVS only)

Retourne la taille réelle d'un fichier dans un dossier

```
int zip_entry_filesize (resource zip_entry)
```

zip_entry_filesize() retourne la taille réelle de l'élément d'archive *zip_entry*. Le paramètre *zip_entry* doit être un élément d'archive valide, créé par la fonction **zip_entry_open()**.

Voir aussi **zip_open()** et **zip_read()**.

zip_entry_name (PHP 4 CVS only)

Retourne le nom de l'élément d'archive

```
string zip_entry_name (resource zip_entry)
```

zip_entry_name() retourne le nom de l'élément d'archive spécifié par *zip_entry*. Le paramètre *zip_entry* doit être un élément d'archive valide, créé par la fonction **zip_entry_open()**.

Voir aussi **zip_open()** et **zip_read()**.

zip_entry_open (PHP 4 CVS only)

Ouvre un nouveau dossier dans une archive

```
bool zip_entry_open (resource zip, resource zip_entry [, string mode])
```

zip_entry_open() ouvre un dossier dans une archive Zip, en lecture seule. Le parmaètre *zip* est une ressource valide, retournée par **zip_open()**. Le paramètre *zip_entry* est une ressource de dossier, retournée par **zip_read()**. Le paramètre optinnel *mode* peut être l'un des mode spécifié dans la documentation de **fopen()**.

> **Note :** Actuellement, *mode* est ignoré et est vaut simplement `"rb"`. Cela est lié au fait que l'extention zip est en lecture seule. Reportez vous à la fonction **fopen()** pour plus de détails sur le mode `"rb"`.

zip_entry_open() retourne `true` en cas de succès, ou `false` en cas d'échec.

> **Note :** Contrairement à **fopen()** et d'autres fonctions du même acabi, la valeur retournée par **zip_entry_open()** indique uniquement le résultat de l'opération, et n'est pas nécessaire pour lire ou fermer le dossier.

Voir aussi **zip_entry_read()** et **zip_entry_close()**.

zip_entry_read (PHP 4 CVS only)

Lit dans un fichier d'archive

```
string zip_entry_read (resource zip_entry [, int length])
```

zip_entry_read() jusqu'à *length* octets dans un fichier d'archive. Si *length* n'est pas spécifié, alors **zip_entry_read()** essaiera de lire 1024 octets. Le paramètre *zip_entry* est un élément d'archive valide, retourné par **zip_read()**.

> **Note :** Le paramètre *length* exprime une taille non compressée.

zip_entry_read() retourne les données lues, ou bien `false` si la fin du fichier est atteinte.

Voir aussi **zip_entry_open()**, **zip_entry_close()** et **zip_entry_filesize()**.

zip_open (PHP 4 CVS only)

Ouvre une archive Zip

```
resource zip_open (string filename)
```

zip_open() ouvre une nouvelle archive en lecture. Le paramètre *filename* est le chemin jusqu'au fichier à ouvrir.

zip_open() retourne une ressource à utiliser plus tard avec les fonctions **zip_read()** et **zip_close()**. **zip_open()** retourne FALSE si *filename* n'existe pas.

Voir aussi **zip_read()** et **zip_close()**.

zip_read (PHP 4 CVS only)

Lit le prochain élément d'archive

```
resource zip_read (resource zip)
```

zip_read() lit le prochain élément d'archive zip, dans l'archive *zip*. Le paramètre *zip* doit être une archive zip, ouverte précédemment par la fonction **zip_open()**.

zip_read() une ressource d'élément d'archive, qui peut être utilisée ultérieurement par les fonctions **zip_open()**, **zip_close()**, **zip_entry_open()** et **zip_entry_read()**.

Voir aussi **zip_open()**, **zip_close()**, **zip_entry_open()** et **zip_entry_read()**.

XCIII. Zlib (Compression)

Ce module utilise les fonctions de la librairie zlib (zlib (http://www.info-zip.org/pub/infozip/zlib/)) de Jean-loup Gailly et Mark Adler pour lire et écrire, de manière transparente, des fichiers compressés avec gzip (.gz). Il faut utiliser la librairie zlib, de version >= 1.0.9.

Ce module contient des versions de la plus part des fonctions du chapitre système de fichier. Mais celles-ci fonctionnent non seulement avec des fichiers compressés, mais aussi des fichiers décompressés (hormis les fonctions utilisant les sockets).

Petit exemple

Ouvre un fichier temporaire, écrit un texte et puis affiche deux fois le contenu.

Exemple 1. Petit exemple avec ZLIB

```
<?php
  $filename = tempnam('/tmp', 'zlibtest').'.gz';
  print "<html>\n<head></head>\n<body>\n<pre>\n";
  $s = "Only a test, test, test, test, test, test, test, test!\n";
  // ouvre un fichier en écriture, avec compression maximale
  $zp = gzopen($filename, "w9");
  // écrit la chaîne dans le fichier
  gzwrite($zp, $s);
  // ferme le fichier
  gzclose($zp);
  // ouvre en lecture
  $zp = gzopen($filename, "r");
  // lis 3 caractères
  print gzread($zp, 3);
  // Affiche le reste du fichier
  gzpassthru($zp);
  print "\n";
  // ouvre le fichier, et affiche le contenu (deuxième passe)
  if (readgzfile($filename) != strlen($s)) {
          echo "Error with zlib functions!";
  }
  unlink($filename);
  print "<pre>\n</h1></body>\n</html>\n";
?>
```

gzclose (PHP 3, PHP 4 >= 4.0b1)

Ferme un pointeur sur un fichier compressé.

```
int gzclose (resource zp)
```

gzclose() ferme le pointeur *zp*.

gzclose() retourne TRUE ou FALSE, suivant le succès ou l'échec.

Le pointeur de fichier compressé doit être valide, et doit référencer un fichier qui a été ouvert par **gzopen()**.

gzeof (PHP 3, PHP 4 >= 4.0b1)

Teste la fin d'un fichier compressé.

```
int gzeof (resource zp)
```

gzeof() retourne TRUE si le pointeur interne du fichier compressé *zp* est à la fin du fichier, sinon retourne FALSE.

Le pointeur de fichier compressé doit être valide, et doit référencer un fichier qui a été ouvert par **gzopen()**.

gzfile (PHP 3, PHP 4 >= 4.0b1)

Lit la totalité d'un fichier compressé dans un tableau.

```
array gzfile (string filename [, int use_include_path])
```

gzfile() est identique à **readgzfile()**, mais **gzfile()** retourne un tableau.

Vous pouvez aussi utiliser le deuxième paramètre optionnel en le mettant à "1", si vous voulez rechercher le fichier dans le dossier include_path.

Voir aussi **readgzfile()**, et **gzopen()**.

gzgetc (PHP 3, PHP 4 >= 4.0b1)

Lit un caractère d'un fichier compressé.

```
string gzgetc (resource zp)
```

gzgetc() retourne une chaîne décompressée, qui contient un caractère unique, lu depuis le fichier référencé par le pointeur *zp*. **gzgetc()** retourne FALSE à la fin du fichier (voir **gzeof()**).

Le pointeur de fichier compressé doit être valide, et doit référencer un fichier qui a été ouvert par **gzopen()**.

Voir aussi **gzopen()** et **gzgets()**.

gzgets (PHP 3, PHP 4 >= 4.0b1)

Lit une ligne d'un fichier compressé

```
string gzgets (resource zp, int length)
```

gzgets() retourne une chaîne décompressée, de longueur inférieure ou égale à length - 1 octets, lue depuis de fichier référencé par le pointeur de fichier *zp*. La lecture s'interrompt lorsque *length* - 1 octets ont été lus, ou bien lorsqu'une nouvelle ligne a été rencontrée, ou bien lorsque la fin du fichier a été atteinte.

Si une erreur survient, retourne FALSE.

Le pointeur de fichier compressé doit être valide, et doit référencer un fichier qui a été ouvert par **gzopen()**.

Voir aussi **gzopen()**, **gzgetc()**, et **fgets()**.

gzgetss (PHP 3, PHP 4 >= 4.0b1)

Lit une ligne d'un fichier compressé et supprime les balises HTML

```
string gzgetss (resource zp, int length [, string allowable_tags])
```

gzgetss() est identique à **gzgets()**, mais elle essaie de supprimer toutes les balises HTML et PHP du texte lu depuis *zp*.

Vous pouvez utiliser le troisième argument optionnel pour indiquer les balises qui ne doivent pas être supprimées.

> **Note :** *allowable_tags* a été ajouté en PHP 3.0.13, PHP 4.0B3.

Voir aussi **gzgets()**, **gzopen()**, et **strip_tags()**.

gzopen (PHP 3, PHP 4 >= 4.0b1)

Ouvre un fichier compressé

```
int gzopen (string filename, string mode [, int use_include_path])
```

gzopen() ouvre un fichier compressé avec gzip (.gz) pour le lire ou l'écrire. Le paramètre de mode est le même que dans **fopen()** ("rb" ou "wb") mais il peut aussi inclure un niveau de compression ("wb9") ou une stratégie: 'f' pour les données filtrées, comme dans "wb6f", 'h' pour Huffman seul , comme dans "wb1h". (Voir la description de deflateInit2 dans zlib.h pour plus de détails a propos des paramètres de stratégie).

gzopen() peut être utilisé pour ouvrir des fichiers qui ne sont pas au format gzip; dans ce cas, **gzread()** lira directement le fichier, sans appliquer de décompression.

gzopen() retourne un pointeur de fichier sur le fichier ouvert. Ce pointeur sera nécessaire pour toutes les opérations ultérieures sur ce fichier. Les opérations de compression/décompression seront transparaentes.

Si l'ouverture échoue, la fonction retourne FALSE.

Vous pouvez utiliser le paramètre optionnel en le mettant à "1", si vous voulez rechercher le fichier dans le dossier include_path.

Exemple 1. Exemple gzopen()

```
<?php
$fp = gzopen("/tmp/file.gz", "r");
?>
```

Voir aussi **gzclose()**.

gzpassthru (PHP 3, PHP 4 >= 4.0b1)

Lit toutes les informations restantes d'un fichier compressé

```
int gzpassthru (resource zp)
```

gzpassthru() lit toutes les informations d'un fichier compressé jusqu'à la fin du fichier *zp*, et écrit le résultat décompressé dans la sortie standard.

Si une erreur survient, retourne FALSE.

Le pointeur de fichier doit être valide, et avoir été ouvert par **gzopen()**.

Le fichier pointé est refermé par **gzpassthru()** ce qui le rend inutilisable pour les opérations ultérieures.

gzputs (PHP 3, PHP 4 >= 4.0b1)

Ecrit dans un fichier compressé

```
int gzputs (resource zp, string str [, int length])
```

gzputs()est un alias de **gzwrite()**, et lui est identique en tous points.

gzread (PHP 3, PHP 4 >= 4.0b1)

Lit un fichier compressé en mode binaire

```
string gzread (resource zp, int length)
```

gzread() lit jusqu'à *length* octets depuis le fichier compressé référencé par *zp*. La lecture stoppe lorsque *length* octets décompressés ont été lus, ou que la fin du fichier a été trouvée.

```
<?php
// lis le contenu d'un fichier compressé et le met dans une chaîne
$filename = "/usr/local/something.txt.gz";
$zd = gzopen( $filename, "r" );
$contents = gzread( $zd, 10000 );
gzclose( $zd );
?>
```

Voir aussi **gzwrite()**, **gzopen()**, **gzgets()**, **gzgetss()**, **gzfile()** et **gzpassthru()**.

gzrewind (PHP 3, PHP 4 >= 4.0b1)

Replace le pointeur courant au début du fichier

```
int gzrewind (resource zp)
```

gzrewind() positionne le pointeur interne du fichier au début du fichier compressé.

Si une erreur survient, retourne 0.

Le pointeur de fichier doit être valide, et avoir été retourné par la fonction **gzopen()**.

Voir aussi **gzseek()** et **gztell()**.

gzseek (PHP 3, PHP 4 >= 4.0b1)

Déplace le pointeur courant dans un fichier compressé

```
int gzseek (resource zp, int offset)
```

gzseek() positionne le pointeur interne du fichier compressé *zp* à la position donnée en *offset*. Equivalent à l'appel (en C) de `gzseek(zp, offset, SEEK_SET)`.

Si le fichier est ouvert en lecture seule, cette fonction émulée peut être extrêmement lente. Si le fichier est ouvert en lecture, seul le déplacement avant (forward seek) sont acceptés. gzseek compresse alors une séquence de zéro jusqu'à la nouvelle position.

gzseek() retourne 0 en cas de succès, sinon retourne -1. Notez que positionner le pointeur au delà de la fin du fichier est une erreur.

Voir aussi **gztell()** et **gzrewind()**.

gztell (PHP 3, PHP 4 >= 4.0b1)

Retourne la position courante du pointeur interne

```
int gztell (resource zp)
```

gztell() retourne la position du pointeur interne du fichier référencé par *zp*, i.e., son offset en octets depuis le début du fichier.

Si une erreur survient, retourne FALSE.

Le pointeur de fichier doit être valide, et doit avoir été retourné par la fonction **gzopen()**.

Voir aussi **gzopen()**, **gzseek()** et **gzrewind()**.

gzwrite (PHP 3, PHP 4 >= 4.0b1)

Ecrit un fichier compressé en mode binaire

```
int gzwrite (resource zp, string string [, int length])
```

gzwrite() écrit le contenu de la chaîne *string* dans le fichier compressé référencé par le pointeur *zp*. Si l'argument *length* est donné, l'écriture cessera après avoir écrit *length* octets (non compressé), ou bien si la fin de la chaîne a été atteinte.

Notez que si l'argument *length* est donnée, alors l'option magic_quotes_runtime sera ignorée et les slashes ne seront pas supprimés de la chaîne *string*.

Voir aussi **gzread()**, **gzopen()** et **gzputs()**.

readgzfile (PHP 3, PHP 4 >= 4.0b1)

Affiche un fichier compressé

```
int readgzfile (string filename [, int use_include_path])
```

readgzfile() lit un fichier, le décompresse et l'écrit dans la sortie standard.

readgzfile() peut être utilisé pour lire un fichier qui n'est pas au format gzip, car dans ce cas, la décompression est omise, et le fichier est directement affiché.

readgzfile() retourne le nombre d'octets (non compressé) lus depuis le fichier. Si une erreur survient, retourne FALSE, et à moins que la fonction n'ai été appelée avec @readgzfile, l'erreur est affichée.

Le fichier *filename* sera ouvert et son contenu sera écrit danas la sortie standard.

Vous pouvez utiliser le paramètre optionnel en le mettant à "1", si vous voulez rechercher le fichier dans le dossier include_path.

Voir aussi **gzpassthru()**, **gzfile()** et **gzopen()**.

gzcompress (PHP 4 >= 4.0.1)

Compresse une cha○ne (ZLIB)

```
string gzcompress (string data [, int level])
```

gzcompress() retourne la version compressé avec le format de données ZLIB de la chaîne *data*, ou FALSE en cas d'erreur. Le paramètre *level* peut prendre des valeurs depuis 0 (pas de compression) jusqu'à 9 (compression maximum).

Pour plus de détails sur la compression ZLIB et son algorithme, reportez vous au document de spécifications ZLIB Compressed Data Format Specification version 3.3 (ftp://ftp.uu.net/pub/archiving/zip/doc/rfc1950.txt) (RFC 1950).

Voir aussi **gzdeflate()**, **gzinflate()**, **gzuncompress()** et **gzencode()**.

gzuncompress (PHP 4 >= 4.0.1)

Décompresse une chaîne gz-compressée

```
string gzuncompress (string data [, int length])
```

gzuncompress() prend la chaîne *data* en entrée (compressée par **gzcompress()**) et retourne la chaîne originale, ou bien FALSE en cas d'erreur. **gzuncompress()** retournera une erreur si la taille de la chaîne décompressée est plus de 256 fois la longueur de la chaîne compressée *data* ou plus que le paramètre optionnel *length*.

> **Note :** Ceci *n'est pas équivalent* à une comrpession gzip, qui inclus en plus des données d'entête. Voir **gzencode()** pour la compression gzip.

Voir aussi **gzdeflate()**, **gzinflate()**, **gzcompress()** et **gzencode()**.

gzdeflate (PHP 4 >= 4.0.4)

Compresse une chaîne (DEFLATE)

```
string gzdeflate (string data [, int level])
```

gzdeflate() retourne la version compressée de *data* en utilisant le format de données DEFLATE, ou FALSE si une erreur est survenue. Le paramètre optionnel *level* peut prendre des valeurs de 0 (pas de compression) jusqu'à 9 (compression maximum, vitesse minimum).

Pour plus de détails sur la compression ZLIB et son algorithme, reportez vous au document de spécifications DEFLATE Compressed Data Format Specification version 1.3 (ftp://ftp.uu.net/pub/archiving/zip/doc/rfc1951.txt) (RFC 1951).

Voir aussi **gzinflate()**, **gzcompress()**, **gzuncompress()** et **gzencode()**.

gzinflate (PHP 4 >= 4.0.4)

Décompresse une chaîne (INFLATE)

```
string gzinflate (string data [, int length])
```

gzinflate() décompresse la chaîne *data*. *data* doit avoir été compressée avec **gzdeflate()**. **gzinflate()** retourne les données originales décompressées, ou bien FALSE si une erreur survient. **gzinflate()** retournera une erreur si les données décompressées sont plus grandes que 256 fois la taille des données compressées *data* ou que le paramètre optionnel *length*.

Voir aussi **gzcompress()**. **gzuncompress()**, **gzdeflate()** et **gzencode()**.

gzencode (PHP 4 >= 4.0.4)

Crée une chaîne compréssée avec gzip

```
string gzencode (string data [, int level])
```

gzencode() retourne la chaîne *data* compressée et compatible avec le programme **gzip**, ou bien FALSE si une erreur survient. Le paramètre optionnel *level* peut prendre des valeurs de 0 (par de compression) jusqu'à 9 (compression maximum, vitess minimum). Par défaut, le niveau de compression est 1.

La chaîne retournée contient les entêtes appropriés et la structure de données demandées par gzip pour faire un fichier .gz file, e.g.:

Exemple 1. Création d'un fichier .gz (gzip)

```
<?php
$data = implode("", "bigfile.txt");
$gzdata = gzencode($data, 9);
$fp = fopen("bigfile.txt.gz", "w");
fwrite($fp, $gzdata);
fclose($fp);
?>
```

Pour plus de détails sur le format GZIP, reportez vous à : GZIP file format specification version 4.3 (ftp://ftp.uu.net/pub/archiving/zip/doc/rfc1952.txt) (RFC 1952).

Voir aussi **gzcompress()**, **gzuncompress()**, **gzdeflate()** et **gzinflate()**.

Partie V. PEAR: the PHP Extension and Application Repository

Chapitre 24. A propos de PEAR

PEAR est dédié à Malin Bakken (http://www.pvv.org/~ssb/malin/bilder/mi/twain001.jpg), né le 21 Novembre 1999 (le premier code de PEAR était écrit quelques heures avant sa naissance).

Qu’est ce que PEAR?

PEAR est une bibliothèque de codes et d’extensions, inspiré du CTAN de TeX et du CPAN de Perl.

Les objectifs de PEAR sont :

- fournir un moyen de partager le codes entre développeurs
- donner à la communauté PHP un système de partage de code
- définir des standards pour aider les développeurs à écrire des codes portables et réutilisables
- fournir des outils de maintenance et de distribution

Chapitre 25. Style de codage PEAR

Indentation

Utilisez une indentation de 4 espaces, mais pas de tabulations. Si vous utilisez Emacs pour éditer du code PEAR, pensez à mettre indent-tabs-mode à nil. Voici un exemple de hook qui va configurer Emacs suivant ces conseils (you devez vous assurer qu'il sera appelé avant l'édition des fichiers PHP) :

```
(defun php-mode-hook ()
  (setq tab-width 4
        c-basic-offset 4
        c-hanging-comment-ender-p nil
        indent-tabs-mode nil))
```

Voici quelques règles pour vim :

```
  set expandtab
  set shiftwidth=4
  set tabstop=4
```

Structures de contrôle

Cela couvre if, for, while, switch, etc... Voici un exemple de condition if, (c'est une des plus compliquées) :

```
if ((condition1) || (condition2)) {
    action1;
} elseif ((condition3) && (condition4)) {
    action2;
} else {
    defaultaction;
}
```

Les structures de contrôle doivent être suivi d'un espace et d'une parenthèse ouvrante, pour les distinguer des appels de fonctions.

Il est vivement recommandé d'utiliser les accolades en toutes occasions, même lorsqu'elles sont techniquement optionnelles. Cela améliore nettement la lisibilité, et réduit la probabilité d'erreurs lorsque de nouvelles lignes sont ajoutées.

Boucle For, switch:

```
switch (condition) {
case 1:
    action1;
    break;

case 2:
    action2;
    break;

default:
    defaultaction;
    break;

}
```

Appels de fonctions

Les fonctions doivent être appelées sans espaces entre le nom de la fonction, la parenthèse ouvrante et le premier argument; il faut place un espace entre chaque argument et la virgule de séparation, mais pas d'espace entre le dernier argument, la parenthèse fermante et le point-virgule. Voici un exemple :

```
$var = foo($bar, $baz, $quux);
```

Comme illustré ci-dessus, il faut un espace autour des signes égal lorsqu'il est utilisé dans une assignation de variable. Dans le cas d'assignation en bloc, plusieurs espaces peuvent être introduit pour améliorer la lisibilité.

```
<?php
$short         = foo($bar);
$long_variable = foo($baz);
>
```

Définitions de fonctions

Les déclarations de fonction suivent la convention "one `TRUE` brace" (une seule accolade véritable) :

```
<?php
function fooFunction($arg1, $arg2 = ”)
{
    if (condition) {
        statement;
    }
    return $val;
}
?>
```

Les arguments ayant une valeur par défaut doivent aller à la fin de la liste des arguments. Essayez de retourner une valeur utile de vos fonctions, dès que c'est le cas. Voici un exemple :

```
<?php
function connect(&$dsn, $persistent = FALSE)
{
    if (is_array($dsn)) {
        $dsninfo = &$dsn;
    } else {
        $dsninfo = DB::parseDSN($dsn);
    }

    if (!$dsninfo || !$dsninfo[’phptype’]) {
        return $this->raiseError();
    }

    return TRUE;
}
?>
```

Commentaires

La documentation du code doit suivre les conventions de PHPDoc, similaire à Javadoc. Plus d'informations sur PHPDoc sont disponibles ici : http://www.phpdoc.de/.

Les commentaires hors documentation sont vivement conseillés. En règle généaral, si vous voyez une portion de code et que vous pensez "Houla!, je n'ai pas envie de tester ça", vous devez la commentez avant d'oublier ce qu'elle fait.

Les commentaires de style C (/* */) et standard C++ (//) sont bien tous les deux. Les autres styles (notamment Perl et shell (#)) sont déconseillés.

Inclusion de code

A chaque fois que vous incluez inconditionnellement une classe, utilisez la fonction **require_once()**. A chaque fois que vous incluez conditionnellement une classe, utilisez la fonction **include_once()**. Ces deux fonctions s'assureront que les classes ne sont déclarées qu'une seule fois. Elles partagent la même liste de fichier, ce qui permet leur utilisation sans gêne. Un fichier inclus par **require_once()** ne sera pas inclus encore une fois avec **include_once()**.

> **Note : include_once()** et **require_once()** sont des commandes, et pas des fonctions. Vous n'êtes pas obligés d'utiliser des parenthèses autour des noms de fichiers.

Balises de code PHP

Utilisez *toujours* la syntaxe `<?php ?>` pour délimiter du code PHP, et jamais `<? ?>`. Ceci est nécessaire pour la compatibilité du code PEAR et c'est la version la plus portable du code PHP sur les différents systèmes d'exploitation.

Entête de fichier

Tous les codes sources des distributions PEAR doivent contenir les commentaires suivants comme entête :

```
/* vim: set expandtab tabstop=4 shiftwidth=4: */
// +----------------------------------------------------+
// | PHP version 4.0                                    |
// +----------------------------------------------------+
// | Copyright (c) 1997, 1998, 1999, 2000, 2001 The PHP Group             |
// +----------------------------------------------------+
// | This source file is subject to version 2.0 of the PHP license,       |
// | that is bundled with this package in the file LICENSE, and is        |
// | available at through the world-wide-web at                           |
// | http://www.php.net/license/2_02.txt.                                 |
// | If you did not receive a copy of the PHP license and are unable to   |
// | obtain it through the world-wide-web, please send a note to          |
// | license@php.net so we can mail you a copy immediately.               |
// +----------------------------------------------------+
// | Authors: Original Author <author@example.com>                    |
// |          Your Name <you@example.com>                             |
// +----------------------------------------------------+
//
// $Id$
```

Il n'y a pas de règle fixe pour déterminer à quel moment un contributeur doit être ajouté dans la liste des auteurs d'un fichier source. En général, les modifications doivent être substancielle (environs 10 à 20% du code initial). Des dérogations peuvent être données pour les réécritures complètes de fonctions, ou les contributions à de nouvelles logiques d'utilisation.

La simple réorganisation de code ou les corrections de bug ne justifie pas l'ajout d'un contributeur dans la liste des auteurs.

Les fichiers qui ne font pas partie de la bibliothèque de base PEAR doivent avoir un bloc comparable à celui cité ci-dessus, indiquant le copyright, la licence et les auteurs. Tous les fichiers devraient inclure une description du modèle suivi, pour améliorer la cohérence de l'ensemble.

Balises CVS

Ajoutez le signe Id (CVS vendor tag) dans chaque fichier. Dans chaque fichier que vous éditez, pensez à l'ajouter s'il n'est pas présent (ou remplacez les anciennes valeurs telles que "Last Modified:", etc.).

> **Note :** Nous avons une marque $Horde dans le CVS Horde pour suivre nos versions séparément. Nous pouvons faire la même chose avec une marque $PEAR, qui resterait même si les fichiers PEAR migrent sur un autre système de suivi de version.

URL d'exemple

Utilisez "example.com" pour tous les exemples, comme spécifié dans la RFC 2606.

Noms des constantes

Les constantes doivent toujours être en majuscule, les mots étant séparés par des soulignés (_). Préfixez les constantes avec le nom de la classe ou du package dont elles font parties. Par exemple, les constantes utilisées dans le package `DB::` commencent toutes par "`DB_`".

XCIV. Manuel de référence PEAR

Ce chapitre contient la documentation de référence des composants PEAR, qui sont distribué avec PHP. Une bonne connaissances des mécanismes objets est nécessaire.

PEAR (unknown)

Classe de base PEAR

Synopsis

```
require_once "PEAR.php";
class classname extends PEAR { ... }
```

La classe de base PEAR fournit les fonctionnalités qui sont utilisées par la plus part des classes PEAR. Normalement, vous n'avez pas à utiliser cette classe directement : il vous suffit d'en hériter.

Les fonctionnalités marquantes sont :

- Fonction de fin de requête
- Gestion des erreurs

Fonction de fin de requête

Si vous faîtes hériter de la classe PEAR, une classe appelée *NomClasse*, vous pouvez définir une méthode appelée *_ClassName* (le nom de la classe précédé d'un souligné), qui sera appelée lorsque le script prend fin. Ce n'est pas un destructeur, car vous pouvez détruire un objet avec cette méthode, mais le destructeur sera quand même appelé. C'est une fonction de callback, qui intervient lorsque le script prend fin. Voyez l'exemple.

Gestion des erreurs

La classe de base PEAR fournit aussi un moyen de manipuler des erreurs plus complexes qu'un simple booléen ou entier. Une erreur PEAR est un objet qui est une instance de la classe PEAR_Error, ou d'une de ses sous-classes.

Une des éxigences de conception des erreurs PEAR est qu'il ne faut pas forcer l'affichage d'erreurs, mais plutôt de fournir les outils pour les afficher, tout en laissant à l'utilisateur le choix de les utiliser. Cela permet de traiter correctement les erreurs, notamment si le format de sortie n'est pas HTML (par exemple WML ou XML).

L'objet d'erreur peut être configuré de nombreuses façons dès sa création : affichage d'erreur, affichage d'erreur suivi de la fin du script, envoie d'une erreur PHP avec **trigger_error()**, appeler une fonction de traitement, ou rien du tout. Vous pouvez typiquement spécifier ces parmètres dès le constructeur de PEAR_Error : tous ces paramètres sont optionnel, et PEAR fournit un moyen de gérer des valeurs par défaut. Voyez les exemples d'erreurs PEAR, pour avoir une illustration, et la documentation de PEAR_Error pour plus de détails.

Les exemples ci-dessus montre comment utiliser les simili destructeurs, pour implémenter une classe qui gère le contenu d'un fichier, ajoute des données, et sauve le tout à la fin du script.

Exemple 1. Les simili destructeurs de PEAR

```
<?php
require_once "PEAR.php";

class FileContainer extends PEAR
{
    var $file = '';
    var $contents = '';
    var $modified = 0;

    function FileContainer($file)
    {
        $this->PEAR(); // cette ligne appelle le constructeur père
        $fp = fopen($file, "r");
        if (!is_resource($fp)) {
            return;
        }
```

```
        while (!empty($data = fread($fp, 2048))) {
            $this->contents .= $data;
     }
        fclose($fp);
    }

    function append($str)
    {
        $this->contents .= $str;
        $this->modified++;
    }

    // le simili-destructeur est nommé d'après le nom de la classe
    // mais avec un souligné devant.
    function _FileContainer()
    {
        if ($this->modified) {
            $fp = fopen($this->file, "w");
            if (!is_resource($fp)) {
                return;
            }
            fwrite($fp, $this->contents);
            fclose($fp);
        }
    }
}

$fileobj = new FileContainer("testfile");
$fileobj->append("Ceci se termine à la fin du fichier\n");

// Lorsque le script est terminé et que PHP s'arrête,
// le simili destructeur de $fileobj est appelé, et
// met à jour les informations sur le disque
?>
```

> **Note :** Les simili destructeurs de PEAR utilisent les fonctions de fin de script de PHP (**register_shutdown_function()**), et vous ne pourrez rien afficher dans ces fonctions, si vous utilisez un serveur web. En ligne de commande, toutefois, l'affichage se fera. C'est comme ca.

L'exemple suivant montre différentes façons d'utiliser le mécanisme d'erreur de PHP.

Exemple 2. Exemple d'erreurs PEAR(1)

```
<?php
function mysockopen($host = "localhost", $port = 8090)
{
    $fp = fsockopen($host, $port, $errno, $errstr);
    if (!is_resource($fp)) {
        return new PEAR_Error($errstr, $errno);
    }
    return $fp;
}

$sock = mysockopen();
if (PEAR::isError($sock)) {
    print "mysockopen error: ".$sock->getMessage()."<BR>\n"
}
?>
```

Cet exemple illustre une fonction de **fsockopen()** qui retourne le code d'erreur et, au cas échéant, le message retourné par le serveur. Notez que **pear->iserror()** sert à savoir si une valeur est une erreur PEAR.

Le mode opératoire de PEAR_Error dans cet exemple est de simplement retourner l'objet d'erreur, et laisser le reste au développeur. C'est le mode par défaut.

Dans le prochain exemple, on utilise d'autres modes :

Exemple 3. Exemple d'erreurs PEAR(2)

```
<?php
class TCP_Socket extends PEAR
{
    var $sock;

    function TCP_Socket()
    {
        $this->PEAR();
    }

    function connect($host, $port)
    {
        $sock = fsockopen($host, $port, $errno, $errstr);
        if (!is_resource($sock)) {
            return $this->raiseError($errstr, $errno);
        }
    }
}

$sock = new TCP_Socket;
$sock->setErrorHandling(PEAR_ERROR_DIE);
$sock->connect("localhost", 8090);
print "Toujours connecté<BR>\n";
?>
```

Ici, le mode par défaut est `PEAR_ERROR_DIE`, et comme on ne spécifie aucun mode d'erreur dans l'appel à (ce devrait être le troisième argument), raiseError utilise le mode par défaut, et termine le script si **fsockopen()** échoue.

PEAR_Error (unknown)

Méchanisme de base des erreurs PEAR

Synopsis

```
$err = new PEAR_Error($msg);
```

Un objet d'erreur a un mode d'opération, qui peut être l'une des constantes suivantes :

PEAR_ERROR_RETURN

: Retourne simplement l'objet, et ne fait rien de spécial dans le constructeur de PEAR_Error.

PEAR_ERROR_PRINT

: Affiche le message d'erreur dans le constructeur. L'exécution n'est pas interrompue.

PEAR_ERROR_TRIGGER

: Utilise la fonction **trigger_error()** pour envoyer une erreur interne à PHP. L'exécution est interrompue si vous avez définit vos propres gestionnaires d'erreurs, ou si vous avez utilisez le niveau d'erreur E_USER_ERROR.

PEAR_ERROR_DIE

: Affiche le message d'erreur et termine le script.

PEAR_ERROR_CALLBACK

Utilise une fonction utilisateur pour gérer les erreurs. L'exécution s'arrête.

pear_error::pear_error ([*message code mode options userinfo*]]]]])

Description

Constructeur PEAR_Error. Paramétres :

message

message d'erreur. Par défaut, `"unknown error"` ("erreur inconnue").

code

code d'erreur (optionnel)

mode

Mode d'opération. Voir les modes d'erreur, pour plus de détails.

options

Si le mode peut avoir des options, utilisez ce paramètre. Actuellement, seules les valeurs de "trigger" et "callback" utilisent ce paramètre. Pour le mode trigger, ce paramètre peut valoir `E_USER_NOTICE`, `E_USER_WARNING` ou `E_USER_ERROR`. Pour le mode callback, ce paramètre doit contenir soit le nom de la fonction de callback (chaîne), soit un tableau à deux éléments (objet, chaîne) représentant un objet et une méthode à appeler.

Partie VI. FAQ: Frequently Asked Questions

Chapitre 26. General Information

This section holds the most general questions about PHP: what it is and what it does.

1. What is PHP?

From the manual (http://www.php.net/manual/):

PHP is an HTML-embedded scripting language. Much of its syntax is borrowed from C, Java and Perl with a couple of unique PHP-specific features thrown in. The goal of the language is to allow web developers to write dynamically generated pages quickly.

A nice introduction to PHP by Stig Sæther Bakken can be found here (http://www.zend.com/zend/art/intro.php) on the Zend website.

2. What is the relation between the versions ?

PHP/FI 2.0 is an early and no longer supported version of PHP. PHP 3 is the successor to PHP/FI 2.0 and is a lot nicer. PHP 4 is the latest generation of PHP, which uses the Zend engine (http://www.zend.com/) under the hood.

3. Can I run several versions of PHP at the same time?

Yes. See the `INSTALL` file that is included in the PHP 4 source distribution.

4. What are the differences between PHP 3 and PHP 4?

There are a couple of articles (http://www.zend.com/zend/art/) written on this by the authors of PHP4. Here's a list of some of the more important new features:

- Extended API module.
- Generalized build process under UNIX
- Generic web server interface that also supports multi-threaded web servers
- Improved syntax highlighter
- Native HTTP session support
- Output buffering support
- More powerful configuration system
- Reference counting

Please see the What's new in PHP4 overview (http://www.zend.com/zend/whats-new.php) for a detailed explanation of these features and more.

Chapitre 27. Mailing lists

This section holds questions how to get in touch with PHP community : the best way is the mailing lists.

1. Is there a PHP mailing list?

Of course! To subscribe, send mail to php-general-subscribe@lists.php.net (mailto:php-general-subscribe@lists.php.net). You don't need to include anything special in the subject or body of the message.

To unsubscribe, send mail to php-general-unsubscribe@lists.php.net (mailto:php-general-unsubscribe@lists.php.net) `php-general-unsubscribe@lists.php.net`.

2. Help! I can't seem to subscribe to the mailing list! Help! I can't seem to unsubscribe from the mailing list!

If you have problems subscribing to or unsubscribing from the PHP mailng list, it may be because the mailing list software can't figure out the correct mailing address to use. If your email address was `joeblow@example.com`, you can send your subscription request to `php-general-subscribe-joeblow=example.com@lists.php.net`, or your unsubscription request to `php-general-unsubscribe-joeblow=example.com@lists.php.net`.

3. Is there an archive of the mailing list anywhere?

Yes, you will find a list of archive sites on the Support (http://www.php.net/support.php) page.

4. What can I ask the mailing list?

Since PHP is growing more and more popular by the day the traffic has increased on the PHP mailing list and as of now the list gets about 150 to 200 posts a day. Because of this it is in everyones interest that you use the list as a last resort when you have looked everywhere else.

Before you post to the list please have a look in this FAQ and the manual to see if you can find the help there. If there is nothing to be found there try out the mailing list archives (see above). If you're having problem with installing or configuring PHP please read through all included documentation and README's. If you still can't find any information that helps you out you're more than welcome to use the mailing list.

5. What information should I include when posting to the mailing list?

Posts like "I can't get PHP up and running! Help me! What is wrong?" are of absolutely no use to anyone. If you're having problems getting PHP up and running you must include what operating system you are running on, what version of PHP you're trying to set up, how you got it (pre-compiled, CVS, RPMs and so on), what you have done so far, where you got stuck and the exact error message.

This goes for any other problem as well. You have to include information on what you have done, where you got stuck, what you're trying to do and, if applicable, exact error messages. If you're having problems with your source code you need to include the part of the code that isn't working. Do not include more code than necessary though! It makes the post hard to read and a lot of people might just skip it all together because of this. If you're unsure about how much information to include in the mail it's better that you include to much than to little.

Another important thing to remember is to summarize your problem on the subject line. A subject like "HELP MEEEE!!!" or "What is the problem here?" will be ignored by the majority of the readers.

Chapitre 28. Obtaining PHP

This section has details about PHP download locations, and OS issues.

1. Where can I obtain PHP?

You can download PHP from any of the members of the PHP network of sites. These can be found at http://www.php.net/. You can also use anonymous CVS to get the absolute latest version of the source. For more information, go to http://cvs.php.net/.

2. Are pre-compiled binary versions available?

Yes, although they are not always up to date. The Windows binary is generally current, but the Unix binaries lag behind and are only available for certain platforms. All download are available in the Downloads (http://www.php.net/downloads.php) section.

3. Where can I get libraries needed to compile some of the optional PHP extensions?

> **Note :** Those marked with * are not thread-safe libraries, and should not be used with PHP as a server module in the multi-threaded Windows web servers (IIS, Netscape). This does not matter in Unix environments, yet.

- LDAP (unix) (ftp://ftp.openldap.org/pub/openldap/openldap-stable.tgz).
- LDAP* (unix) (ftp://terminator.rs.itd.umich.edu/ldap/ldap-3.3.tar.Z).
- LDAP (unix/win) (http://developer.netscape.com/tech/directory/downloads.html) : Netscape Directory (LDAP) SDK 1.1.
- free LDAP server (http://developer.netscape.com/tech/directory/downloads.html).
- Berkeley DB2 (Unix/Win) (http://www.sleepycat.com/) : http://www.sleepycat.com/.
- SNMP* (Unix): (http://www.ece.ucdavis.edu/ucd-snmp/).
- GD* (Unix/Win) (http://www.boutell.com/gd/#buildgd).
- mSQL* (Win) (http://blnet.com/msqlpc/).
- mSQL* (Unix) (http://www.hughes.com.au/).
- MySQL* (Unix) (http://www.mysql.com/).
- IMAP* (Win/Unix) (ftp://ftp.cac.washington.edu/imap/old/imap-4.5.tar.Z).
- Sybase-CT* (Linux, libc5) (http://www.php.net/extra/ctlib-linux-elf.tar.gz) : Available locally.
- FreeType (libttf): (http://www.freetype.org/).
- ZLib (Unix/Win32) (http://www.cdrom.com/pub/infozip/zlib/).
- expat XML parser (Unix/Win32) (http://www.jclark.com/xml/expat.html).
- PDFLib (http://www.pdflib.com/).
- mcrypt (ftp://argeas.cs-net.gr/pub/unix/mcrypt/).
- mhash (http://sasweb.de/mhash/).
- t1lib (http://www.neuroinformatik.ruhr-uni-bochum.de/ini/PEOPLE/rmz/t1lib/t1lib.html).
- dmalloc (http://www.dmalloc.com/).
- aspell (http://download.sourceforge.net/aspell/aspell-.29.1.tar.gz).
- readline (ftp://prep.ai.mit.edu/pub/gnu/readline/).

4. How do I get these libraries to work?

You will need to follow instructions provided with the library. Some of these libraries are detected automatically when you run the 'configure' script of PHP (such as the GD library), and others you will have to enable using '`--with-EXTENSION`' options to '`configure`'. Run '`configure --help`' for a listing of these.

5. I got the latest version of the PHP source code from the CVS repository on my Windows 95/NT machine, what do I need to compile it?

First, you will need Microsoft Visual C++ v6 (v5 may do it also, but we do it with v6), and you will need to download the support files (http://www.php.net/extra/win32build.zip). You will need to unzip this file (which has subdirectories, so make sure your unzip program keeps them) into the win32 subdirectory of the source distribution.

6. Where do I find the Browser Capabilities File?

You can find a `browscap.ini` file at http://www.cyscape.com/asp/browscap/.

Chapitre 29. Connecting to databases

This section holds common questions about relation between PHP and databases : Yes, PHP can acces virtually any database available today.

1. I heard it’s possible to access Microsoft SQL Server from PHP. How?

On Windows 95/NT machines, you can simply use the included ODBC support and the correct ODBC driver.

On Unix machines, you can use the Sybase-CT driver to access Microsoft SQL Servers because they are (at least mostly) protocol-compatible. Sybase has made a free version of the necessary libraries for Linux systems (http://www.php.net/extra/ctlib-linux-elf.tar.gz). For other Unix operating systems, you need to contact Sybase for the correct libraries. Also see the answer to the next question - 4.2.

2. Can I access Microsoft Access databases?

Yes. You already have all the tools you need if you are running entirely under Windows 95/98 or NT, where you can use ODBC and Microsoft’s ODBC drivers for Microsoft Access databases.

If you are running PHP on a Unix box and want to talk to MS-Access on a Windows box you will need Unix ODBC drivers. OpenLink Software (http://www.openlinksw.com/) has Unix-based ODBC drivers that can do this. There is a free pilot program where you can download an evaluation copy that doesn’t expire and prices start at $675 for the commercial supported version.

Another alternative is to use an SQL server that has Windows ODBC drivers and use that to store the data, which you can then access from Microsoft Access (using ODBC) and PHP (using the built-in drivers), or to use an intermediary file format that Access and PHP both understand, such as flat-files or dBase databases. On this point Tim Hayes from OpenLink software writes:

```
Using another database as an intermediary is not a good idea, when you can
use ODBC from PHP straight to your database - i.e. with OpenLink’s drivers. If
you do need to use an intermediary file format, OpenLink have now released
Virtuoso (a virtual database engine) for NT, Linux and other unix platforms.
Please visit our
website (http://www.openlinksw.com/) for a free download.
```

One option that has proven successful is to use MySQL and its MyODBC drivers on Windows and synchronizing the databases. Steve Lawrence writes:

- Install MySQL on your platform according to instructions with MySQL. Latest available from www.mysql.com (http://www.mysql.com/) (get it from your mirror!). No special configuration required except when you set up a database, and configure the user account, you should put % in the host field, or the host name of the Windows computer you wish to access MySQL with. Make a note of your server name, username, and password.
- Download the MyODBC for Windows driver from the MySQL site. Latest release is myodbc-2_50_19-win95.zip (NT available too, as well as source code). Install it on your Windows machine. You can test the operation with the utilities included with this program.
- Create a user or system dsn in your ODBC administrator, located in the control panel. Make up a dsn name, enter your hostname, user name, password, port, etc for you MySQL database configured in step 1.
- Install Access with a full install, this makes sure you get the proper add-ins.. at the least you will need ODBC support and the linked table manager.
- Now the fun part! Create a new access database. In the table window right click and select Link Tables, or under the file menu option, select Get External Data and then Link Tables. When the file browser box comes up, select files of type: ODBC. Select System dsn and the name of your dsn created in step 3. Select the table to link, press ok, and presto! you can now open the table and add/delete/edit data on your MySQL server! You can also build queries, import/export tables to MySQL, build forms and reports, etc.

Tips and Tricks:

- You can construct your tables in access and export them to MySQL, then link them back in. That makes table creation quick.

- When creating tables in access, you must have a primary key defined in order to have write access to the table in access. Make sure you create a primary key in MySQL before linking in access
- If you change a table in MySQL, you have to re-link it in access. Go to tools>add-ins>linked table manager, cruise to your ODBC DSN, and select the table to re-link from there. you can also move your dsn source around there, just hit the always prompt for new location checkbox before pressing ok.

3. I saw PHP offers persistent database connections. What does that mean?

Persistent connections are SQL links that do not close when the execution of your script ends. When a persistent connection is requested, PHP checks if there's already an identical persistent connection (that remained open from earlier) - and if it exists, it uses it. If it does not exist, it creates the link. An 'identical' connection is a connection that was opened to the same host, with the same username and the same password (where applicable).

People who aren't thoroughly familiar with the way web servers work and distribute the load may mistake persistent connects for what they're not. In particular, they do *not* give you an ability to open 'user sessions' on the same SQL link, they do *not* give you an ability to build up a transaction efficently, and they don't do a whole lot of other things. In fact, to be extremely clear about the subject, persistent connections don't give you *any* functionality that wasn't possible with their non-persistent brothers.

Why?

This has to do with the way web servers work. There are three ways in which your web server can utilize PHP to generate web pages.

The first method is to use PHP as a CGI "wrapper". When run this way, an instance of the PHP interpreter is created and destroyed for every page request (for a PHP page) to your web server. Because it is destroyed after every request, any resources that it acquires (such as a link to an SQL database server) are closed when it is destroyed. In this case, you do not gain anything from trying to use persistent connections -- they simply don't persist.

The second, and most popular, method is to run PHP as a module in a multiprocess web server, which currently only includes Apache. A multiprocess server typically has one process (the parent) which coordinates a set of processes (its children) who actually do the work of serving up web pages. When each request comes in from a a client, it is handed off to one of the children that is not already serving another client. This means that when the same client makes a second request to the server, it may be serviced by a different child process than the first time. What a persistent connection does for you in this case it make it so each child process only needs to connect to your SQL server the first time that it serves a page that makes us of such a connection. When another page then requires a connection to the SQL server, it can reuse the connection that child established earlier.

The last method is to use PHP as a plug-in for a multithreaded web server. Currently this is only theoretical -- PHP does not yet work as a plug-in for any multithreaded web servers. Work is progressing on support for ISAPI, WSAPI, and NSAPI (on Windows), which will all allow PHP to be used as a plug-in on multithreaded servers like Netscape FastTrack, Microsoft's Internet Information Server (IIS), and O'Reilly's WebSite Pro. When this happens, the behavior will be essentially the same as for the multiprocess model described before.

If persistent connections don't have any added functionality, what are they good for?

The answer here is extremely simple -- efficiency. Persistent connections are good if the overhead to create a link to your SQL server is high. Whether or not this overhead is really high depends on many factors. Like, what kind of database it is, whether or not it sits on the same computer on which your web server sits, how loaded the machine the SQL server sits on is and so forth. The bottom line is that if that connection overhead is high, persistent connections help you considerably. They cause the child process to simply connect only once for its entire lifespan, instead of every time it processes a page that requires connecting to the SQL server. This means that for every child that opened a persistent connection will have its own open persistent connection to the server. For example, if you had 20 different child processes that ran a script that made a persistent connection to your SQL server, you'd have 20 different connections to the SQL server, one from each child.

An important summary. Persistent connections were designed to have one-to-one mapping to regular connections. That means that you should *always* be able to replace persistent connections with non-persistent connections, and it won't change the way your script behaves. It *may* (and probably will) change the efficiency of the script, but not its behavior!

4. I upgraded to php4, and now mysql keeps telling me "Warning: MySQL: Unable to save result set in ...". What's up?

Most likely what has happened is, PHP4 was compiled with the '--with-mysql' option, without specifying the path to mysql. This means PHP is using its built-in mysql client library. If your system is running applications, such as php3 as a concurrent Apache module, or auth-mysql, that use other versions of mysql clients, then there is a conflict between the two differing versions of those clients.

Recompiling php4, and adding the path to mysql to the flag, '--with-mysql=/your/path/to/mysql' usually solves the problem.

5. After installing shared mysql support, Apache dumps core as soon as libphp4.so is loaded. Can this be fixed?

If your MySQL libs are linked against pthreads this will happen. Check using ldd. If they are, grab the MySQL tarball and compile from source, or recompile from the source rpm and remove the switch in the spec file that turns on the threaded client code. Either of these suggestions will fix this. Then recompile PHP with the new mysql libs.

Chapitre 30. Installation

This section holds common questions about the way to install PHP. PHP is available for almost any OS (except may be for MacOS before OSX), and almost any web server.

To install PHP, follow the instructions in the INSTALL (http://cvs.php.net/co.php/php4/INSTALL) file located in the distribution. Windows 95 and NT users should also read the install.txt (http://cvs.php.net/co.php/php4/win32/install.txt) file. There are also some helpful hints for Windows users here.

If you are trying to install PHP for use with Netscape's web server on Unix see:
http://www.webgenx.com/php/phpnes.php3

1. Where should my php.ini file be located?

By default on UNIX it should be in `/usr/local/lib`. Most people will want to change this at compile-time with the --with-config-file-path flag. You would, for example, set it to something like:

```
--with-config-file-path=/etc
```

And then you would copy `php.ini-dist` from the distribution to `/etc/php.ini` and edit it to make any local changes you want.

2. I installed PHP using RPMS, but Apache isn't processing the PHP pages! What's going on here?

Assuming you installed both Apache and PHP from RPM packages, you need to uncomment or add some or all of the following lines in your http.conf file:

```
# Extra Modules
AddModule mod_php.c
AddModule mod_php3.c
AddModule mod_perl.c

# Extra Modules
LoadModule php_module         modules/mod_php.so
LoadModule php3_module        modules/libphp3.so     /* for PHP 3 */
LoadModule php4_module        modules/libphp4.so     /* for PHP 4 */
LoadModule perl_module        modules/libperl.so
```

And add:

```
AddType application/x-httpd-php3 .php3    /* for PHP 3 */
AddType application/x-httpd-php .php      /* for PHP 4 */
```

... to the global properties, or to the properties of the VirtualDomain you want to have PHP support added to.

3. I installed PHP using RPMS, but it doesn't compile with the database support I need! What's going on here?

Due to the way PHP is currently built, it is not easy to build a complete flexible PHP RPM. This issue will be addressed in PHP 4. For PHP, we currently suggest you use the mechanism described in the INSTALL.REDHAT file in the PHP distribution. If you insist on using an RPM version of PHP, read on...

Currently the RPM packagers are setting up the RPMS to install without database support to simplify installations *and* because RPMS use /usr/ instead of the standard /usr/local/ directory for files. You need to tell the RPM spec file which databases to support and the location of the top-level of your database server.

This example will explain the process of adding support for the popular MySQL database server, using the mod installation for Apache.

Of course all of this information can be adjusted for any database server that PHP supports. We will assume you installed MySQL and Apache completely with RPMS for this example as well.

- First remove mod_php3 :

  ```
  rpm -e mod_php3
  ```

- Then get the source rpm and INSTALL it, NOT --rebuild

  ```
  rpm -Uvh mod_php3-3.0.5-2.src.rpm
  ```

- Then edit the `/usr/src/redhat/SPECS/mod_php3.spec` file

  In the %build section add the database support you want, and the path.

  For MySQL you would add

  ```
  -with-mysql=/usr \
  ```

  The %build section will look something like this:

  ```
  ./configure --prefix=/usr \
  --with-apxs=/usr/sbin/apxs \
  --with-config-file-path=/usr/lib \
  --enable-debug=no \
  --enable-safe-mode \
  --with-exec-dir=/usr/bin \
  --with-mysql=/usr \
  --with-system-regex
  ```

- Once this modification is made then build the binary rpm as follows:

  ```
  rpm -bb /usr/src/redhat/SPECS/mod_php3.spec
  ```

- Then install the rpm

  ```
  rpm -ivh /usr/src/redhat/RPMS/i386/mod_php3-3.0.5-2.i386.rpm
  ```

Make sure you restart Apache, and you now have PHP with MySQL support using RPM's. Note that it is probably much easier to just build from the distribution tarball of PHP and follow the instructions in `INSTALL.REDHAT` found in that distribution.

Chapitre 31. Build Problems

This section gathers most common errors that occur at build time.

1. I got the latest version of PHP using the anonymous CVS service, but there's no configure script!

You have to have the GNU autoconf package installed so you can generate the configure script from configure.in. Just run **./buildconf** in the top-level directory after getting the sources from the CVS server. (Also, unless you run configure with the `--enable-maintainer-mode` option, the configure script will not automatically get rebuilt when the configure.in file is updated, so you should make sure to do that manually when you notice configure.in has changed. One symptom of this is finding things like @VARIABLE@ in your Makefile after configure or config.status is run.)

2. I'm having problems configuring PHP to work with Apache. It says it can't find httpd.h, but it's right where I said it is!

You need to tell the configure/setup script the location of the top-level of your Apache source tree. This means that you want to specify '`--with-apache=/path/to/apache`' and *not* '`--with-apache=/path/to/apache/src`'.

3. When I run configure, it says that it can't find the include files or library for GD, gdbm, or some other package!

You can make the configure script looks for header files and libraries in non-standard locations by specifying additional flags to pass to the C preprocessor and linker, such as:

```
CPPFLAGS=-I/path/to/include LDFLAGS=-L/path/to/library ./configure
```

If you're using a csh-variant for your login shell (why?), it would be:

```
env CPPFLAGS=-I/path/to/include LDFLAGS=-L/path/to/library ./configure
```

4. When it is compiling the file language-parser.tab.c, it gives me errors that say 'yytname undeclared'.

You need to update your version of Bison. You can find the latest version at ftp://ftp.gnu.org/pub/gnu/bison/.

5. When I run 'make', it seems to run fine but then fails when it tries to link the final application complaining that it can't find some files.

Some old versions of make that don't correctly put the compiled versions of the files in the functions directory into that same directory. Try running "**cp *.o functions**" and then re-running '**make**' to see if that helps. If it does, you should really upgrade to a recent version of GNU make.

6. When linking PHP, it complains about a number of undefined references.

Take a look at the link line and make sure that all of the appropriate libraries are being included at the end. Common ones that you might have missed are '-ldl' and any libraries required for any database support you included.

If you're linking with Apache 1.2.x, did you remember to add the appropriate information to the EXTRA_LIBS line of the Configuration file and re-rerun Apache's Configure script? See the INSTALL (http://cvs.php.net/co.php/php4/INSTALL) file that comes with the distribution for more information.

Some people have also reported that they had to add '-ldl' immediately following '`libphp3.a`' when linking with Apache.

7. I can't figure out how to build PHP with Apache 1.3.

This is actually quite easy. Follow these steps carefully:

- Grab the latest Apache 1.3 distribution from http://www.apache.org/dist/.
- Ungzip and untar it somewhere, for example `/usr/local/src/apache-1.3`.
- Compile PHP by first running **./configure --with-apache=/<path>/apache-1.3** (substitute <path> for the actual path to your apache-1.3 directory.
- Type '`make`' followed by '`make install`' to build PHP and copy the necessary files to the Apache distribution tree.
- Change directories into to your `/<path>/apache-1.3/src` directory and edit the `Configuration` file. At the end of the file, add: `AddModule modules/php3/libphp3.a`.
- Type: '**./Configure**' followed by '`make`'.
- You should now have a PHP-enabled httpd binary!

Note: : You can also use the new Apache `./configure` script. See the instructions in the `README.configure` file which is part of your Apache distribution. Also have a look at the `INSTALL` file in the PHP distribution.

8. I have followed all the steps to install the Apache module version on UNIX, and my PHP scripts show up in my browser or I am being asked to save the file.

This means that the PHP module is not getting invoked for some reason. Three things to check before asking for further help:

- Make sure that the httpd binary you are running is the actual new httpd binary you just built. To do this, try running: `/path/to/binary/httpd -l`

  If you don't see `mod_php3.c` listed then you are not running the right binary. Find and install the correct binary.

- Make sure you have added the correct Mime Type to one of your `Apache .conf` files. It should be: `AddType application/x-httpd-php3 .php3` (for PHP 3)

  or `AddType application/x-httpd-php .php` (for PHP 4)

  Also make sure that this AddType line is not hidden away inside a <Virtualhost> or <Directory> block which would prevent it from applying to the location of your test script.

- Finally, the default location of the Apache configuration files changed between Apache 1.2 and Apache 1.3. You should check to make sure that the configuration file you are adding the AddType line to is actually being read. You can put an obvious syntax error into your httpd.conf file or some other obvious change that will tell you if the file is being read correctly.

9. It says to use: `--activate-module=src/modules/php4/libphp4.a`, but that file doesn't exist, so I changed it to `--activate-module=src/modules/php4/libmodphp4.a` and it doesn't work!? What's going on?

Well, you decided to try to outsmart the people who wrote those nice step-by-step instructions for you and you have now discovered that these people cannot be outsmarted. The libphp4.a file is not supposed to exist. The Apache build process will create it.

10. When I try to build Apache with PHP as a static module using `--activate-module=src/modules/php4/libphp4.a` it tells me that my compiler is not ANSI compliant.

This is a misleading error message from Apache that has been fixed in more recent versions.

11. When I try to build PHP using `--with-apxs` I get strange error messages.

There are three things to check here. First, for some reason when Apache builds the apxs Perl script, it sometimes ends up getting built without the proper compiler and flags variables. Edit your apxs (sometimes found in /usr/local/apache/bin/apxs or /usr/sbin/apxs) and check for these lines:

```
my $CFG_CFLAGS_SHLIB  = ' ';      # substituted via Makefile.tmpl
my $CFG_LD_SHLIB      = ' ';          # substituted via Makefile.tmpl
my $CFG_LDFLAGS_SHLIB = ' ';      # substituted via Makefile.tmpl
```

If this is what you see, you have found your problem. Change these lines to say:

```
my $CFG_CFLAGS_SHLIB  = '-fpic -DSHARED_MODULE';      # substituted via Makefile.tmpl
my $CFG_LD_SHLIB      = 'gcc';          # substituted via Makefile.tmpl
my $CFG_LDFLAGS_SHLIB = q(-shared);# substituted via Makefile.tmpl
```

The second possible problem should only be an issue on RedHat-6.1/6.2. The apxs script RedHat ships is broken. Look for this line:

```
my $CFG_LIBEXECDIR    = 'modules';        # substituted via APACI install
```

If you see the above line, change it to this:

```
my $CFG_LIBEXECDIR    = '/usr/lib/apache';        # substituted via APACI install
```

Last, if you reconfigure/reinstall Apache, add a 'make clean' to the process after './configure' and before 'make'.

12. During ’make’, I get errors in microtime, and a lot of ’RUSAGE_’ stuff.

During the ’make’ portion of installation, if you encounter problems that look similar to this:

```
microtime.c: In function ‘php_if_getrusage’:
microtime.c:94: storage size of ‘usg’ isn’t known
microtime.c:97: ‘RUSAGE_SELF’ undeclared (first use in this function)
microtime.c:97: (Each undeclared identifier is reported only once
microtime.c:97: for each function it appears in.)
microtime.c:103: ‘RUSAGE_CHILDREN’ undeclared (first use in this function)
make[3]: *** [microtime.lo] Error 1
make[3]: Leaving directory ‘/home/master/php-4.0.1/ext/standard’
make[2]: *** [all-recursive] Error 1
make[2]: Leaving directory ‘/home/master/php-4.0.1/ext/standard’
make[1]: *** [all-recursive] Error 1
make[1]: Leaving directory ‘/home/master/php-4.0.1/ext’
make: *** [all-recursive] Error 1
```

Your system is broken. You need to fix your /usr/include files either by making sure your /usr/include/linux symlink is pointing to the right place in your kernel sources or by installing a glibc-devel package that matches your glibc. This has absolutely nothing to do with PHP. To prove this to yourself, try this simple test:

```
$ cat >test.c <<X
#include <sys/resource.h>
X
$ gcc -E test.c >/dev/null
```

If that spews out errors, you know your include files are messed up.

Chapitre 32. Using PHP

This section gathers most common errors that occur at build time.

1. I would like to write a generic PHP script that can handle data coming from any form. How do I know which POST method variables are available?

Make sure that the track_vars feature is enabled in your php3.ini file. If you compiled PHP with "--enable-track-vars" it will be on by default. Alternatively you can enable it at run-time on a per-script basis by putting <?php_track_vars?> at the top of your file. When track_vars is on, it creates three associative arrays. $HTTP_GET_VARS, $HTTP_POST_VARS and $HTTP_COOKIE_VARS. So, to write a generic script to handle POST method variables you would need something similar to the following:

```
while (list($var, $value) = each($HTTP_POST_VARS)) {
    echo "$var = $value<br>\n";
}
```

2. I need to convert all single-quotes (’) to a backslash followed by a single-quote. How can I do this with a regular expression?

First off, take a look at the **addslashes()()** function. It will do exactly what you want. You should also have a look at the magic_quotes_gpc directive in your `php.ini` file.

3. When I do the following, the output is printed in the wrong order:

```
function myfunc($argument) {
  echo $argument + 10;
}
$variable = 10;
echo "myfunc($variable) = " . myfunc($variable);
```

what’s going on?

To be able to use the results of your function in an expression (such as concatenating it with other strings in the example above), you need to *return* the value, not **echo()** it.

4. Hey, what happened to my newlines?

```
<PRE>
  1 <?echo $result[1];?>
  2 <?echo $result[2];?>
```

In PHP, the ending for a block of code is either "?>" or "?>\n" (where \n means a newline). This means that you need to insert an extra newline after each block of PHP code in the above example.

Why does PHP do this? Because when formatting normal HTML, this usually makes your life easier because you don’t want that newline, but you’d have to create extremely long lines or otherwise make the raw page source unreadable to achieve that effect.

5. I need to access information in the request header directly. How can I do this?

The **getallheaders()** function will do this if you are running PHP as a module. So, the following bit of code will show you all the request headers:

```
$headers = getallheaders();
for(reset($headers); $key = key($headers); next($headers)) {
    echo "headers[$key] = ".$headers[$key]."<br>\n";
}
```

6. When I try to use authentication with IIS I get ’No Input file specified’.

The security model of IIS is at fault here. This is a problem common to all CGI programs running under IIS. A workaround is to create a plain HTML file (not parsed by php) as the entry page into an authenticated directory. Then use a META tag to redirect to the PHP page, or have a link to the PHP page. PHP will then recognize the authentication correctly. When the ISAPI module is ready, this will no longer be a problem. This should not effect other NT web servers. For more information, see: http://support.microsoft.com/support/kb/articles/q160/4/22.asp.

7. I've followed all the instructions, but still can't get PHP and IIS to work together!

Make sure any user who needs to run a PHP script has the rights to run `php.exe`! IIS uses an anonymous user which is added at the time IIS is installed. This user needs rights to `php.exe`. Also, any authenticated user will also need rights to execute `php.exe`. And for IIS4 you need to tell it that PHP is a script engine.

8. My PHP script works on IE and Lynx, but on Netscape some of my output is missing. When I do a "View Source" I see the content in IE but not in Netscape.

Very good question! ;) This is a tricky little issue and it has come up twice in the past month as of this writing. Both times I ended up spending a good 20 minutes trying to figure out what the heck was going on. The answer is that both IE and Lynx ignore any NULs (`\0`) in the HTML stream. Netscape does not. The best way to check for this is to compile the command-line version of PHP (also known as the CGI version) and run your script from the command line and pipe it through 'od -c' and look for any `\0` characters. (If you are on Windows you need to find an editor or some other program that lets you look at binary files) When Netscape sees a NUL in a file it will typically not output anything else on that line whereas both IE and Lynx will. If this issue has bitten you, congratulations! You are not alone.

9. How am I supposed to mix XML and PHP? It complains about my <?xml> tags!

You need to turn off the short tags by setting short_tags to 0 in your `php.ini` file, or by using the php3_short_tags Apache directive. (You could even use a <File> section to do this selectively.) You can also disable and re-enable the short tags in your script using the short_tags() function.

10. How can I use PHP with FrontPage or Dreamweaver or some other HTML editor that insists on moving my code around?

One of the easiest things to do is to enable using ASP tags in your PHP code. This allows you to use the ASP-style <% and %> code delimiters. Most of the popular HTML editors handle those more intelligently (for now). To enable the ASP-style tags, you need to set the asp_tags `php.ini` variable, or use the asp_tags Apache directive.

11. Where can I find a complete list of pre-set variables available to me, and why are these not documented in the PHP documentation?

The best way is to stick a <?phpinfo()?> tag on a page and load it up. This will show you all sorts of information about your PHP setup, including a list of both environment variables and also special variables set by your web server. This list can't really be documented in the PHP documentation because it will change from one server to another.

12. Why do I get an error that looks something like this: "Warning: 0 is not a MySQL result index in <file> on line <x>" or "Warning: Supplied argument is not a valid MySQL result resource in <file> on line <x>?

You are trying to use a result identifier that is 0. The 0 indicates that your query failed for some reason. You need to check for errors after submitting a query and before you attempt to use the returned result identifier. The proper way to do this is with code similar to the following:

```
$result = mysql_query("select * from tables_priv");
if(!$result) {
    echo mysql_error();
    exit;
}
```

or

```
$result = mysql_query("select * from tables_priv")
            or die("Bad query: ".mysql_error());
```

13. I'm trying to use an <input type="image"> tag, but the $foo.x and $foo.y variables aren't available. Where are they?

When submitting a form, it is possible to use an image instead of the standard submit button with a tag like:

```
<input type="image" SRC="image.gif" NAME="foo">
```

When the user clicks somewhere on the image, the accompanying form will be transmitted to the server with two additional variables: foo.x and foo.y.

Because $foo.x and $foo.y are invalid variable names in PHP, they are automagically converted to $foo_x and $foo_y. That is, the periods are replaced with underscores.

14. How do I get all the results from a SELECT MULTIPLE HTML tag?

The SELECT MULTIPLE tag in an HTML construct allows users to select multiple items from a list. These items are then passed to the action handler for the form. The problem is that they are all passed with the same widget name. ie.

```
<SELECT NAME="var" MULTIPLE>
```

Each selected option will arrive at the action handler as:

```
var=option1
var=option2
var=option3
```

Each option will overwrite the contents of the previous $var variable. The solution is to use PHP's non-indexed array feature. The following should be used:

```
<SELECT NAME="var[]" MULTIPLE>
```

This tells PHP to treat `var` as an array and each assignment of a value to var[] adds an item to the array. The first item becomes `$var[0]`, the next `$var[0]`, etc. The **count()** function can be used to determine how many options were selected, and the **sort()** function can be used to sort the option array if necessary.

Note that if you are using JavaScript the `[]` on the element name might cause you problems when you try to refer to the element by name. Use it's numerical form element id instead, or enclose the variable name in single quotes and use that as the index to the elements array, for example:

```
variable = documents.forms[0].elements['var[]'];
```

Chapitre 33. PHP and HTML

PHP and HTML interact a lot : PHP generate HTML, and HTML has informations that will be sent to PHP.

1. How do I create arrays in a HTML <form>?

To get your <form> result sent as an array to your PHP script you name the <input>, <select> or <textarea> elements like this:

```
<input name="MyArray[]">
<input name="MyArray[]">
<input name="MyArray[]">
<input name="MyArray[]">
```

Notice the square brackets after the variable name, that's what makes it an array. You can group the elements into different arrays by assigning the same name to different elements:

```
<input name="MyArray[]">
<input name="MyArray[]">
<input name="MyOtherArray[]">
<input name="MyOtherArray[]">
```

This produces two arrays, MyArray and MyOtherArray, that gets sent to the PHP script.

Note that you must not use indices with arrays in HTML! The array gets filled in the order the elements appear in the form. For functions you can use to process these arrays once you get them into your scripts, please see the Arrays section in the manual.

Chapitre 34. PHP and other languages

PHP is the best language for Webbing, but what about other languages?

1. PHP vs. ASP?

ASP is not really a language in itself, it's an acronym for Active Server Pages, the actual language used to program ASP with is a script version of Visual Basic. The biggest drawback of ASP is that it's a proprietary system that is natively used only on Microsoft Internet Information Server (IIS). This limits it's availability to Win32 based servers. There are a couple of projects in the works that allows ASP to run in other environments and webservers; InstantASP (http://www.halcyonsoft.com/prods/iasp/iasp.htm) from Halcyon (http://www.halcyonsoft.com/) (commercial), Chili!Soft ASP (http://www.chilisoft.com/) from Chili!Soft (http://www.chilisoft.com/chiliasp/default.asp) (commercial) and OpenASP from ActiveScripting.org (http://www.activescripting.org/) (free). ASP is said to be a slower and more cumbersome language than PHP, less stable as well. Some of the pros of ASP is that since it uses VBScript it's relatively easy to pick up the language if you're already know how to program in Visual Basic. ASP support is also enabled by default in the IIS server making it easy to get up and running.

2. Is there an ASP to PHP converter?

Yes, asp2php (http://asp2php.naken.cc/) is the one most often referred to.

3. PHP vs. Cold Fusion?

PHP is commonly said to be faster and more efficient for complex programming tasks and trying out new ideas. PHP is generally referred to as more stable and less resource intensive as well. Cold Fusion has better error handling, database abstraction and date parsing although database abstraction is being addressed in PHP 4. Another thing that is listed as one of Cold Fusion's strengths is its excellent search engine, but it has been mentioned that a search engine is not something that should be included in a web scripting language. PHP runs on almost every platform there is; Cold Fusion is only available on Win32, Solaris, Linux and HP/UX. Cold Fusion has a better IDE and is generally easier to get started with, whereas PHP initially requires more programming knowledge.

A great summary by Michael J Sheldon on this topic has been posted to the PHP mailing list. A copy can be found here (http://marc.theaimsgroup.com/?l=php3-general&m=95602167412542&w=1).

4. PHP vs. Perl?

The biggest advantage of PHP over Perl is that PHP was designed for scripting for the web where Perl was designed to do a lot more and can because of this get very complicated. The flexibility / complexity of Perl makes it easier to write code that another author / coder has a hard time reading. PHP has a less confusing and stricter format without losing flexibility. PHP is easier to integrate into existing HTML than Perl. PHP has pretty much all the 'good' functionality of Perl; constructs, syntax and so on, without making it as complicated as Perl can be. Perl is a very tried and true language, it's been around since the late eighties, but PHP is maturing very quickly.

Chapitre 35. Common Problems

Here is a bunch of common problems, met every day with PHP, and solved the easy way!

1. I installed PHP, but every time I load a document, I get the message 'Document Contains No Data'! What's going on here?

This probably means that PHP is having some sort of problem and is core-dumping. Look in your server error log to see if this is the case, and then try to reproduce the problem with a small test case. If you know how to use 'gdb', it is very helpful when you can provide a backtrace with your bug report to help the developers pinpoint the problem. If you are using PHP as an Apache module try something like:

- Stop your httpd processes
- gdb httpd
- Stop your httpd processes
- > run -X -f /path/to/httpd.conf
- Then fetch the URL causing the problem with your browser
- > run -X -f /path/to/httpd.conf
- If you are getting a core dump, gdb should inform you of this now
- type: bt
- You should include your backtrace in your bug report. This should be submitted to http://bugs.php.net/

If your script uses the regular expression functions (**ereg()()** and friends), you should make sure that you compiled PHP and Apache with the same regular expression package. (This should happen automatically with PHP and Apache 1.3.x)

2. I'm trying to access one of the standard CGI variables (such as $DOCUMENT_ROOT or $HTTP_REFERER) in a user-defined function, and it can't seem to find it. What's wrong?

Environment variables are now normal global variables, so you must either declare them as global variables in your function (by using "`global $DOCUMENT_ROOT;`", for example) or by using the global variable array (ie, "`$GLOBALS["DOCUMENT_ROOT"]`".

3. I patched Apache with the FrontPage extensions patch, and suddenly PHP stopped working. Is PHP incompatible with the Apache FrontPage extensions?

No, PHP works fine with the FrontPage extensions. The problem is that the FrontPage patch modifies several Apache structures, that PHP relies on. Recompiling PHP (using 'make clean ; make') after the FP patch is applied would solve the problem.

4. I think I found a bug! Who should I tell?

You should go to the PHP Bug Database and make sure the bug isn't a known bug. If you don't see it in the database, use the reporting form to report the bug. It is important to use the bug database instead of just sending an email to one of the mailing lists because the bug will have a tracking number assigned and it will then be possible for you to go back later and check on the status of the bug. The bug database can be found at http://bugs.php.net/.

Chapitre 36. Migrating from PHP 2 to PHP 3

PHP has already a long history behind him : Legendary PHP 1.0, PHP/FI, PHP 3.0 and PHP/Zend 4.0.

1. Migrating from PHP2 to PHP3?

PHP/FI 2.0 is no longer supported. Please see the manual for information about migration from PHP/FI 2.0.

Chapitre 37. Migrating from PHP 3 to PHP 4

PHP has already a long history behind him : Legendary PHP 1.0, PHP/FI, PHP 3.0 and PHP/Zend 4.0.

1. Migrating from PHP3 to PHP4

PHP 4 was designed to be as compatible with earlier versions of PHP as possible and very little functionality was broken in the process. If you’re really unsure about compatibility you should install PHP 4 in a test environment and run your scripts there.

2. Incompatibles functions?

Since PHP 4 is basically a rewrite of the entire PHP engine there was very few functions that were altered and only then some of the more exotic ones.

Chapitre 38. Miscellaneous Questions

There can be some questions we can’t put into other categories. Here you can find them.

1. Where did the pop-ups go on the website? Can I have the code for that?

The yellow pop-up windows on the old site were pretty cool, but were very difficult to maintain (since some companies seem to enjoy changing the way their browsers work with every new release).

All the code for previous versions of the website is still available through CVS. Specifically, the last version of shared.inc (that had all the Javascript and DHTML to do the popups) is available here (http://cvsweb.php.net/co.php/phpweb/include/shared.inc?r=1.123).

Partie VII. Appendices

Annexe A. Migration de PHP/FI 2.0 à PHP 3.0

A propos des incompatibilités en 3.0

PHP 3.0 a été entièrement réécrit. Le nouvel analyseur syntaxique est beaucoup plus robuste et cohérent qu'en version 2.0. Il est aussi nettement plus rapide et utilise encore moins de mémoire. Cependant, ces améliorations n'ont pu être possible qu'au prix de modifications parfois importantes, tant au niveau des syntaxes, qu'au niveau des fonctionnalités.

De plus, l'équipe de développement PHP a essayé de nettoyer la syntaxe et les sémantiques, ce qui a aussi causé quelques incompatibilités. A long terme, nous pensons que ces modifications seront pour le bien de tous.

Ce chapitre va tenter de vous montrer les incompatibilités que vous pourriez rencontrer lors de votre migration de PHP/FI 2.0 à PHP 3.0 et de vous aider à les résoudre. Les nouvelles fonctionnalités ne sont pas signalées, à moins que cela ne soit nécessaire.

Un programme de conversion automatique de vos vieux script PHP/FI 2.0 existe. Il est disponible dans le dossier de convertisseur de la distribution PHP 3.0. Ce programme ne fait que repérer les modifications de syntaxe et ne vous épargnera pas une relecture attentive du script.

Balises PHP

La première chose que vous remarquerez probablement est que les balises de PHP start et end ont changé. L'ancienne forme `<? ?>` a été remplacée par trois nouvelles balises possibles :

Exemple A-1. Migration: Migration: balises start/end

```
<?php
  echo "Ceci est du code PHP/FI 2.0.\n";?
?>
```

Comme en version 2.0, PHP/FI accepte aussi cette variante :

Exemple A-2. Migration: premières nouvelles balises PHP

```
<?php
  echo "Ceci est du code PHP 3.0!\n";
?>
```

Notez bien que la balise de fin contient désormais un point d'interrogation et un signe supérieur ">". Cependant, si vous souhaitez utiliser XML sur votre serveur, vous aurez sÛrement des problèmes avec cette variante, car PHP risque d'essayer d'exécuter des balises XML. A cause de ceci, la notation suivante a été ajoutée :

Exemple A-3. Migration: Nouvelles balises PHP

```
<?php
  echo "Ceci est du code PHP 3.0!\n";
?>
```

Certains d'entre vous rencontrent des problèmes avec les éditeurs qui ne comprennent pas ce type de balises d'instruction : Microsoft FrontPage est l'un de ces éditeurs, et, pour contourner le problème, la variation suivante a été introduite :

Exemple A-4. Nouvelles balises PHP

```
<script language="php">
  echo "Ceci est du code PHP 3.0!\n";
</script>
```

Syntaxe if..endif

La syntaxe alternative pour écrire des instructions if/elseif/else, avec if(); elseif(); else; endif; ne pouvait pas être conservée sans ajouter beaucoup de complexité à l'analyseur syntaxique. De ce fait, cette syntaxe à changée :

Exemple A-5. Migration: ancienne syntaxe if..endif

```
<?php
  if ($foo);
    echo "oui\n";
  elseif ($bar);
    echo "presque\n";
  else;
    echo "non\n";
  endif;
?>
```

Exemple A-6. Migration: nouvelle syntaxe if..endif

```
<?php
  if ($foo):
    echo "oui\n";
  elseif ($bar):
    echo "presque\n";
  else:
    echo "non\n";
  endif;
?>
```

Notez que les points virgules ont été remplacée par des points dans toutes les commandes, sauf pour la dernière expression (endif).

Syntaxe while

Tout comme pour if..endif, la syntaxe des boucles while..endwhile a changée :

Exemple A-7. Migration: ancienne syntaxe while..endwhile

```
<?php
  while ($more_to_come);
    ...
  endwhile;
?>
```

Exemple A-8. Migration: nouvelle syntaxe while..endwhile

```
<?php
  while ($more_to_come):
    ...
  endwhile;
?>
```

Avertissement

Attention : si vous utilisez la vieille syntaxe while..endwhile en PHP 3.0, vous obtiendrez une boucle sans fin !

Types d’expression

PHP/FI 2.0 utilisait le membre à gauche dans les expressions, pour déterminer le type de résultat attendu. PHP 3.0 prend en compte les deux côtés de l’expression et cela peut produire des résultats inattendus avec les scripts 2.0.

Considérez les lignes suivantes:

```
<?php
  $a[0]=5;
  $a[1]=7;
  $key = key($a);
  while ("" != $key) {
    echo "$keyn";
    next($a);
  }
?>
```

En PHP/FI 2.0, cet exemple va afficher les indices des $a. En PHP 3.0, l’exemple ne va rien afficher du tout. La raison est qu’en PHP 2.0, puisque l’argument de gauche est de type chaîne, une comparaison de chaîne était effectuée et, effectivement, `""` n’est pas `""`, ce qui conduit la boucle à continuer. En PHP 3, lorsqu’une chaîne est comparée avec un entier, la comparaison est de type chaîne (la chaîne est convertie en entier). Ce qui revient à faire la comparaison entre (`atoi("")`) qui vaut `0` et la variable qui vaut aussi 0 et comme 0==0, la boucle ne commence même pas.

La correction de ceci est simple : il suffit de remplacer les commandes while par:

```
<?php
  while ((string)$key != "") {
?>
```

Les messages d’erreur ont changé

Les messages d’erreur en PHP 3.0 sont généralement plus précis que ceux de la version 2.0., mais vous ne verrez plus la portion de code qui a causé l’erreur. A la place, un numéro de ligne et un nom de fichier sera retourné.

Evaluation rapide des booléens

En PHP 3., l’évaluation des est court-circuité. Cela signifie dans une expression telle que (`(1 || test_me())`), la fonction test_me() ne sera pas exécutée, car cela ne changera pas le résultat.

C’est une amélioration mineure, mais qui peut avoir des effets secondaires importants.

La valeur `TRUE`/`FALSE` comme retour de fonctions

La plupart des fonctions internes de PHP ont été réécrite pour qu’elle retourne `TRUE` en cas de succès, et `FALSE` en cas d’erreur, au contraire des fonctions qui retournaient 0 et -1 en PHP/FI 2.0. Le nouveau comportement est beaucoup plus logique, comme par exemple $fp = fopen("/your/file") or fail("fichier non trouvé!");. Etant donné que PHP/FI 2.0 n’a pas de règle claire à propos de ce que les fonctions doivent retourner en cas d’échec, la plupart des scripts devront probablement être vérifié manuellement, après avoir utilisé le convertisseur 2.0 à 3.0.

Exemple A-9. Migration depuis 2.0: valeur retournées, ancienne façon

```
<?php
  $fp = fopen($file, "r");
  if ($fp == -1);
    echo("Impossible d’ouvrir le fichier $file en lecture <br>\n");
  endif;
?>
```

Exemple A-10. Migration depuis 2.0: valeur retournées, nouvelle façon

```
<?php
  $fp = @fopen($file, "r") or
     print("Impossible d'ouvrir le fichier $file en lecture<br>\n");
?>
```

Diverses incompatibilités

- Le module PHP 3.0 pour Apache n'accepte plus les versions d'Apache antérieure à la version 1.2. Apache 1.2 ou plus récent est nécessaire.
- **echo()** n'utilise plus de chaîne de formatage. Il faut utiliser **printf()** à la place.
- En PHP/FI 2.0, un effet secondaire de l'implémentation faisait que `$foo[0]` était la même chose que `$foo`. Ce n'est plus vrai en PHP 3.0.
- Lire un tableau avec $array[] n'est plus valable.

  Ainsi, il n'est plus possible de passer en revue un tableau avec des boucles telles que `$data = $array[]`. Utilisez **current()** et **next()** à la place.

  Ainsi, `$array1[] = $array2` n'ajoute pas les valeurs de `$array2` à `$array1`, mais crée un nouvel élément dans `$array1` et y affecte `$array2` comme dernier élément. Voir aussi les tableaux multidimensionnels.

- `"+"` n'est plus utilisable comme opérateur de concaténation de chaîne. A la place, il convertit les arguments en nombres et effectue une addition numérique. Utilisez `"."` à la place.

Exemple A-11. Migration depuis 2.0: concaténation de chaînes

```
<?php
  echo "1" + "1";
?>
```

En PHP 2.0 cela retournerait 11, en PHP 3.0 cela va retourner 2. A la place, faites :

```
<?php
  echo "1"."1";
?>
```

```
<?php
  $a = 1;
  $b = 1;
  echo $a + $b;
?>
```

Cela va afficher 2, tant en PHP 2.0 qu'en 3.0.

```
<?php
  $a = 1;
  $b = 1;
  echo $a.$b;
?>
```

Cela va afficher 11 en PHP 3.0.

Annexe B. Migration de PHP 3.0 à PHP 4.0

Ce qui a changé en PHP 4.0

PHP 4.0 et le moteur Zend ont significativement amélioré les performances et les possibilités de PHP, tout en assurant une compatibilité ascendante maximale. Le maximum de codes existants sous PHP 3.0 fonctionneront sous PHP 4.0. La migration de votre code de PHP 3.0 vers PHP 4.0 sera beaucoup plus facile que celle de PHP/FI 2.0 vers 3.0. Un grand nombre de scripts seront prêts sans modifications, mais il est bon que vous connaissiez les quelques différences, et que vous testiez vos applications avant d'effecteur le changement de cadre de production. Les indications suivantes vous mettront sur la voie.

Comportement de l'analyseur

L'analyse et l'éxecution sont désormais deux étapes complètement dissociées, et l'éxécution intervient lorsque le code, ainsi que tous ses inclusions et pré-requis, ont été complètement analysés et validés.

Une des nouvelles conditions introduites est que les fichiers inclus et requis (**include()** et **require()**) doivent être syntaxiquement complets. Vous ne pouvez plus répartir différents cas de votre code dans plusieurs fichiers. Vous ne pouvez plus commencer une boucle `for` ou `while`, une condition `if` ou un cas `switch` dans un fichier, et finir la boucle ou placer les cas `else`, `endif`, `case` ou `break` dans un autre fichier.

Il est toujours valable d'inclure du code supplémentaire depuis une boucle ou dans une condition, mais les accolades de bloc `{...}`, et les éléments de la boucle doivent être dans le même fichier ou chaîne évaluée avec **eval()**.

Cela ne devrait pas perturber trop de monde, car étaler son code de cette façon est plutôt un style à éviter.

Une autre nouveauté est qu'il est plus possible de faire retourner une valeur avec un fichier requis (**require()**) (mais c'est plutôt rare en PHP 3.0). Retourner une valeur avec un fichier inclus (**include()**) est toujours possible.

Rapport d'erreur

Changement de configuration

Avec PHP 3.0, le niveau de rapport d'erreur était obtenu en ajoutant les constantes numériques de chaque niveau de rapport. Généralement, on utilisait 15 pour afficher toutes les erreurs, et 7 pour afficher toutes les erreurs hormis les alertes simples.

PHP 4.0 dispose d'un nombre significativement plus grand de niveaux de rapport d'erreur, et l'analyseur comprend désormais les constantes, lors des modifications.

Le niveau de rapport d'erreur doit désormais être explicitement configuré en supprimant les niveaux dont vous ne voulez pas du niveau maximal, grâce à la fonction de OU exclusif. Ça a l'air compliqué? Supposons que vous souhaitiez afficher toutes les erreurs, hormis les alertes de style, qui sont repérées par la constante : E_NOTICE. Il suffit d'ajouter la valeur suivante dans le fichier `php.ini`: `error_reporting = E_ALL & ~ ( E_NOTICE )`. Si vous voulez supprimer en plus les alertes, vous pouvez ajouter la constante appropriée, en la combinant avec l'opérateur OU logique '|': `error_reporting= E_ALL & ~ ( E_NOTICE | E_WARNING )`.

Avertissement

L'utilisation des vieilles valeurs de 7 et 15 est une très mauvaise idée, car elles ne prennent pas en compte les nouvelles classes d'erreurs, y compris certaines erreurs d'analyse. Cela peut conduire à de très étranges résultats, où le script n'affiche plus rien, malgré une erreur d'analyse.

Cela a conduit à un grand nombre de rapport d'erreur dans le passé, alors que les programmeurs n'étaient tout simplement pas capables de repérer l'accolade manquante, car l'analyseur avait la consigne de cacher ces erreurs.

Vérifier votre niveau d'erreur doit être le premier réflexe lorsque vos scripts meurent silencieusement. Le moteur Zend est considéré actuellement comme suffisamment mature pour ne plus causer ce genre de problème aujourd'hui.

Nouveaux messages d'erreurs

Un grand nombre de scripts PHP PHP 3.0 utilisent des structures qui doivent être considérées comme un très mauvais style, même s'il effectue bien la tâche qui lui est affectée, car ils ne sont pas robustes. PHP 4.0 affichera de nombreux

messages d'erreurs dans des situations où PHP 3.0 restera coi. La solution de facilité consiste à supprimer les messages de niveau E_NOTICE, mais c'est une meilleure idée de corriger le code à la place.

Le cas le plus courant qui génèrera des messages d'alertes est l'utilisation de constantes sans guillemets comme index de tableaux. PHP 3.0, comme PHP 4.0, finiront par les interpréter littéralement comme des chaînes, si aucune constante n'est définie à la place. Mais si jamais une telle constante est définie dans une autre partie du code, cela risque de produire des résultats étonnants. Cela peut devenir un trou de sécurité si un pirate arrive à redéfinir les constantes de telle manière que le script lui donne accès à un niveau de droits supérieur. PHP 4.0 vous signalera tout oubli de guillemets comme par exemple dans : `$HTTP_SERVER_VARS[REQUEST_METHOD]`. Modifier ce code en `$HTTP_SERVER_VARS['REQUEST_METHOD']` rendra l'analyseur heureux, et améliorera grandement votre style et la sécurité du code.

PHP 4.0 vous signalera les variables ou les éléments de tableaux non initialisés.

Initialiseur

Les variables statiques et les membres de classes n'acceptent plus que des initialiseurs scalaires, tandis que PHP 3.0 acceptait aussi les expressions. Cela est dÛ, encore une fois, à la séparation de l'analyse et de l'exécution : aucun code ne peut être exécuté tant que l'analyse n'est pas terminée.

Pour les classes, il vaut mieux initialiser les membres dans le constructeur. Pour les variables statiques, une valeur fixe et simple est la seule chose qui viennent à l'esprit.

`empty("0")`

L'évolution la plus polémique est celle de **empty()**. Une chaîne contenant seulement le caractère `'0'` (zéro) est maintenant considérée comme vide, alors qu'elle ne l'était pas en PHP 3.0.

Ce nouveau comportement prend tout son sens dans les applications web, puisque tous les résultats de champs de type input sont de type chaîne de caractères, même si un nombre est demandé, et ce, grâce aux capacités de conversion automatique de PHP. D'un autre côté, cela peut casser votre code d'une manière très subtile, menant droit au comportement erratique, difficilement repérable si vous ne savez pas ce qui vous attend.

Fonctions manquantes

Bien que PHP 4.0 dispose de nombreuses nouvelles fonctionnalités fonctions et extensions, vous vous rencontrer des fonctions PHP 3.0 qui manquent. Un petit nombre de fonctions de base n'ont pu être portées en PHP 4.0, maintenant que l'analyse et l'éxécution ont été séparées. D'autres fonctions, et mêmes des extensions entières sont maintenant obsolètes, remplacées par de nouvelles fonctions plus puissantes ou plus efficaces. Certaines fonctions n'ont tout simplement pas été portées pour le moment ou pour des raisons de licences.

Fonctions manquantes pour des raisons de structure

Comme PHP 4.0 sépare l'analyse et l'éxécution, il n'est plus possible de modifier le comportement de l'analyseur (intégré dans le moteur Zend) durant l'éxécution, puisque toute l'analyse a eu lieu, et est terminée. La fonction short_tags() a cessé d'exister. Vous pouvez toujours modifier le comportement de l'analyseur avec le fichier `php.ini`.

Une autre fonctionnalité qui a disparu est le débuggeur de PHP 3.0, comme décrit dans un autre appendice. Un nouveau débuggeur est promis par Zend, mais il n'a pas encore montré le bout de son nez.

Fonctions et extensions obsolètes

Les extensions Adabas et Solid n'existent plus. Elles sont intégrées dans les fonctions ODBC Unifié.

Nouveau statut pour unset()

unset(), bient que toujours disponible, a été implémenté légèrement différemment en PHP 4.0, et elle n'est plus vraiment une 'fonction'.

Cela n’a pas de conséquence directe sur le comportement de **unset()**, mais utiliser cette fonction pour faire un test avec **function_exists()** retournera `FALSE` comme il se doit avec les fonction bas niveau comme **echo()**.

Une autre application pratique disparue est qu’il n’est plus possible d’appeler **unset()** indirectement, c’est-à-dire que `$func="unset"; $func($somevar)` ne fonctionne plus.

Extensions PHP 3.0

Les extensions écrites pour PHP 3.0 ne fonctionnent plus avec PHP 4.0, ni les exécutables, ni les codes sources. Il n’est pas difficile de porter les extensions de PHP 3.0 à 4.0 si vous avez accès aux sources originales. Une description détaillée du processus de portage ne fait pas partie de cet appendice (pour le moment).

Substitution de variables dans les chaînes

PHP 4.0 dispose d’un nouveau mécanisme de substitution des variables dans les chaînes. Vous pouvez désormais accéder aux membres d’objets et aux tableaux multidimensionnels dans une chaîne.

Pour cela, il suffit de placer la variable entre accolades, le signe $ suivant immédiatement la première accolade : `{$variable['a']}`

Pour utiliser la valeur d’un membre d’objet dans une chaîne, il suffit d’écrire : `"text {$obj->member} text"`; alors qu’en PHP 3.0, il fallait faire comme ceci : `"texte".$objet->membre." texte"`.

Cette technique rend le code beaucoup plus lisible, mais risque de poser des problèmes dans certains scripts PHP 3.0. Vous pouvez facilement traquer ce problème en recherche les séquences `{$` dans votre code, et en les remplaçant par `\{$` avec votre outil de remplacement habituel.

Cookies

PHP 3.0 avait la mauvaise habitude d’envoyer les cookies dans l’ordre inverse de celui du code (l’ordre des appels à **setcookie()**). PHP 4.0 rétablit l’ordre naturel en les envoyant dans le même ordre que vous même.

Cela peut aussi prendre à contre-pied certains programmes, mais ce comportement était tellement étrange qu’il méritait un tel traitement un jour ou l’autre, pour éviter d’autres problèmes ultérieurs.

Annexe C. Développement PHP

Créer une fonction PHP 3

Prototypes de fonctions

Toutes les fonctions suivent le schéma suivant :

```
void php3_foo(INTERNAL_FUNCTION_PARAMETERS) {
}
```

Même si votre fonction ne prend aucun argument, c'est comme cela qu'elle doit être appelée.

Arguments de fonctions

Les arguments sont toujours de type val. Ce type contient un membre de type union, qui indique le type reél d'argument. De cette façon, si votre fonction prend deux arguments, elle ressemble à ceci :

Exemple C-1. Argument de fonction de lecture

```
pval *arg1, *arg2;
if (ARG_COUNT(ht) != 2 || getParameters(ht,2,&arg1,&arg2)==FAILURE) {
   WRONG_PARAM_COUNT;
}
```

NOTE: Les arguments peuvent être passé par valeur ou par référence. Dans les deux cas, vous devez passer &(pval *) à getParameters. Si vous voulez vérifier que le n-ième paramètre a été passé par référence ou par valeur, vous devez utiliser la fonction ParameterPassedByReference(ht,n). Elle retournera 1 ou 0.

Lorsque vous modifiez l'un des paramètres, qu'ils soient envoyés par référence ou par valeur, vous pouvez le passer à pval_destructor pour le réinitialiser, ou, s'il s'agit d'un tableau et que vous voulez ajouter des valeurs, vous pouvez utiliser des fonctions similaires à celles qui sont dans internal_functions.h, qui manipule return_value comme tableau.

Par ailleurs, si vous modifiez un paramètre en IS_STRING, assurez-vous que vous avez bien assigné un nouvelle chaîne avec estrdup() et une nouvelle longueur de chaîne. Seulement après, vous pouvez modifier le type en IS_STRING. Si vous modifiez une chaîne en IS_STRING ou IS_ARRAY vous devez d'abord appeler le destructeur pval_destructor.

Fonctions à nombre d'arguments variable

Une fonction peut prendre un nombre variable d'arguments. Si votre fonction peut prendre deux ou trois arguments, utiliser la syntaxe suivante :

Exemple C-2. Fonctions à nombre d'arguments variable

```
pval *arg1, *arg2, *arg3;
int arg_count = ARG_COUNT(ht);
if (arg_count < 2 || arg_count > 3 ||
    getParameters(ht,arg_count,&arg1,&arg2,&arg3)==FAILURE) {
    WRONG_PARAM_COUNT;
}
```

Utiliser les arguments d'une fonction

De type de chaque argument est stocké dans le champs pval. Ce champs peut prendre les valeurs suivantes :

Tableau C-1. Types de données interne PHP

IS_STRING	Chaîne de caractères
IS_DOUBLE	Nombre à virgule flottante, en précision double

IS_LONG	Entier long
IS_ARRAY	Tableau
IS_EMPTY	Aucune
IS_USER_FUNCTION	??
IS_INTERNAL_FUNCTION	?? (Si ce type ne peut pas être passé à une fonction, effacez-le)
IS_CLASS	??
IS_OBJECT	??

Si vous recevez un argument d'un type, et que vous voulez l'utiliser avec un autre type, ou si vous voulez simplement forcer le type, vous pouvez utiliser l'une des fonctions de conversion suivantes :

```
convert_to_long(arg1);
convert_to_double(arg1);
convert_to_string(arg1);
convert_to_boolean_long(arg1);
/* Si la chaîne est "" ou "0" elle devient 0, 1 sinon */
convert_string_to_number(arg1);
/* Convertit une chaîne en LONG ou DOUBLE suivant la chaîne */
```

Ces fonctions convertissent sur place : elles ne retournent aucune valeur.

La valeur de l'argument est enregistrée dans une union. Les membres sont :

- IS_STRING: arg1->value.str.val
- IS_LONG: arg1->value.lval
- IS_DOUBLE: arg1->value.dval

Gestion de la mémoire dans une fonction

Toute la mémoire nécessaire à une fonction doit être allouée avec emalloc() ou estrdup(). Ces fonctions ont le goÛt et l'odeur des fonctions C classiques malloc() et strdup(). La mémoire doit être libérée avec efree().

Il y a deux types de mémoire dans ce programme : la mémoire qui est retournée à l'analyseur, et la mémoire qui nécessaire pour le stockage temporaire dans la fonction. Lorsque vous assignez une chaîne dans une variable qui est retournée à l'analyseur, assurez-vous de bien allouer la mémoire avec emalloc() ou estrdup(). Cette mémoire ne doit JAMAIS être libérée, sauf si vous réécrivez votre original plus loin, dans la même fonction (mais ce n'est pas de la programmation propre).

Pour tous vos besoins en mémoire temporaire/permanante dont vous avez besoin dans vos fonctions/librairies, vous devez utiliser les fonctions emalloc(), estrdup() et efree(). Elles se comportent EXACTEMENT comme leurs homologues. Tout ce qui est créé avec emalloc() ou estrdup() doit être libéré avec efree() à un moment ou un autre, à moins que ce ne soit utile ailleurs dans le programme; sinon, il va y avoir une fuite de mémoire. La signification de "Elles se comportent EXACTEMENT comme leurs homologues" est que si vous libérez une variable qui n'a pas été créée avec emalloc() ou estrdup(), vous courez droit à au crash ("segmentation fault"). Soyez alors extrêmement prudent, et libérez toute votre mémoire inutilisée.

Si vous compilez avec "-DDEBUG", PHP 3 affichera la liste de tous les appels à emalloc() et estrdup() mais jamais à efree() lorsque celui-ci intervient dans un script spécifié.

Affecter une variable dans la table des symboles

Un grand nombre de macros sont disponibles pour rendre plus facile l'insertion de variables dans la table des symboles :

- SET_VAR_STRING(name,value)
- SET_VAR_DOUBLE(name,value)
- SET_VAR_LONG(name,value)

Gestion de la mémoire

> Soyez prudent. La valeur doit être placée dans une portion de mémoire créée avec malloc(), sinon le gestionnaire de mémoire essayera de libérer le pointeur plus tard. Ne passez aucune mémoire allouée statiquement à SET_VAR_STRING.

Les tables des symboles de PHP 3 est une table de hash. A n'importe quel moment, &symbol_table est un pointeur sur la table principale, et active_symbol_table pointe sur la table actuellement utilisée. (ces deux tables peuvent être identiques au démarrage, ou différent, suivant que vous êtes dans une fonction ou non).

Les exemples suivants utilisent 'active_symbol_table'. Vous devriez la remplacer par &symbol_table si vous voulez travailler sur la table principale. De plus, les mêmes fonctions peuvent être appliquées à des tableaux, comme expliqué ci-dessous.

Exemple C-3. Vérification de l'existence de `$foo` dans la table des symboles

```
if (hash_exists(active_symbol_table,"foo",sizeof("foo"))) {
 // existe...
} else {
 // n'existe pas
}
```

Exemple C-4. Rechercher la taille d'une variable dans la table des symboles

```
hash_find(active_symbol_table,"foo",sizeof("foo"),&pvalue);
check(pvalue.type);
```

En PHP 3.0, les tableaux sont implémentés en utilisant les mêmes tables de hash que les variables. Cela signifie que les deux fonctions ci-dessus peuvent être appelées pour vérifier la présence de variables dans un tableau.

Si vous voulez définir un nouveau tableau dans la table des symboles, utilisez le code suivant.

D'abord, vous devez vérifier qu'il n'existe pas, avec hash_exists() ou hash_find().

Puis, initialisez le tableau :

Exemple C-5. Initialisation d'un tableau

```
pval arr;
if (array_init(&arr) == FAILURE) { /*Initialiation échouée*/ };
hash_update(active_symbol_table,"foo",sizeof("foo"),&arr,sizeof(pval),NULL);
```

Ce code déclare un nouveau tableau, appelé $foo, dans la table de symbole. Ce tableau est vide.

Voici comment ajouter deux nouvelles entrées dans ce tableau :

Exemple C-6. Ajout d'entrées dans un tableau.

```
pval entry;
entry.type = IS_LONG;
entry.value.lval = 5;
/* définit $foo["bar"] = 5 */
hash_update(arr.value.ht,"bar",sizeof("bar"),&entry,sizeof(pval),NULL);
/* définit $foo[7] = 5 */
hash_index_update(arr.value.ht,7,&entry,sizeof(pval),NULL);
/* définit la prochaine place libre dans $foo[],
 * $foo[8], qui sera 5 (comme en php2)
```

```
 */
hash_next_index_insert(arr.value.ht,&entry,sizeof(pval),NULL);
```

Si vous voulez modifier une valeur que vous avez inséré dans une table de hash, vous devez d'abord la lire dans la table. Pour éviter cette recherche, vous pouvez fournir une pval ** à la fonction d'ajout dans la table de hash, et elle modifiera la valeur à l'adresse pval *, avec la valeur donnée. Si cette valeur est `NULL`, (comme dans tous les exemples ci dessus), ce paramètre sera ignoré.

hash_next_index_insert() utiliser plus ou moins la même logique que "$foo[] = bar;" in PHP 2.0.

Si vous construisez un tableau, pour le retourner, vous pouvez l'initialiser comme ceci :

```
if (array_init(return_value) == FAILURE) { échec...; }
```

puis ajouter les valeurs grâces aux macros:

```
add_next_index_long(return_value,long_value);
add_next_index_double(return_value,double_value);
add_next_index_string(return_value,estrdup(string_value));
```

Bien sÛr, si l'ajout n'est pas fait juste après l'initialisation, vous devrez d'abord rechercher le tableau :

```
pval *arr;
if (hash_find(active_symbol_table,"foo",sizeof("foo"),(void **)&arr)==FAILURE)
{ introuvable... }
else
{ utilisez arr->value.ht... }
```

Notez que hash_find reçoit un pointeur sur un pointeur sur pval, et pas un pointeur sur pval.

Toutes les fonctions d'accès aux hash retourne `TRUE` (SUCCES) ou `FALSE` (FAILURE), excepté hash_exists(), qui retourne un booléen.

Retourne une valeur simple

Un grand nombre de macros sont disponible pour simplifier le retour des valeurs.

La macro RETURN_* fixe la valeur de retour, et termine la fonction :

- RETURN
- RETURN_FALSE
- RETURN_TRUE
- RETURN_LONG(l)
- RETURN_STRING(s,dup) Si dup est `TRUE`, duplique la chaîne.
- RETURN_STRINGL(s,l,dup) retourne la chaîne (s) en spécifiant la longueur (l).
- RETURN_DOUBLE(d)

La macro RETVAL_* macros fixe la valeur de retour, mais ne termine pas la fonction.

- RETVAL_FALSE
- RETVAL_TRUE
- RETVAL_LONG(l)
- RETVAL_STRING(s,dup) Si dup est `TRUE`, duplique la chaîne
- RETVAL_STRINGL(s,l,dup) retourne la chaîne (s) en spécifiant la longueur (l).
- RETVAL_DOUBLE(d)

Les macros ci-dessus vont utiliser estrdup() sur les arguments passés. Cela vous permet de libérer tranquillement les arguments après avoir appelé cette fonction, ou bien, utiliser de la mémoire allouée statiquement.

Si votre fonction retourne un booléen de succès/erreur, utilisez toujours RETURN_TRUE et RETURN_FALSE respectivement.

Retourner des valeurs complexes

Votre fonction peut aussi retourner des valeurs complexes, tels que des objets ou tableaux.

Retourner un objet:

1. Appeler object_init(return_value).
2. Remplissez les valeurs. Les fonctions utilisables sont listées ci dessous.
3. Eventuellement, enregistrez les fonctions pour cet objet. Afin de lire des valeurs de cet objet, la fonction doit lire dans "this", dans la table de symbole active active_symbol_table. Son type doit être IS_OBJECT, et c'est une table de hash basique. (i.e., vous pouvez utiliser les fonctions habituelles de .value.ht). L'enregistrement reél peut être fait comme suit :

   ```
   add_method( return_value, function_name, function_ptr );
   ```

Les fonctions d'accès aux objets sont :

- add_property_long(return_value, property_name, l) - Ajoute un membre nommé 'property_name', de type long, égal à 'l'
- add_property_double(return_value, property_name, d) - Idem, ajoute un double
- add_property_string(return_value, property_name, str) - Idem, ajoute une chaîne
- add_property_stringl(return_value, property_name, str, l) - Idem, ajoute une chaîne de longueur 'l'.

Retournez un tableau :

1. Appelez array_init(return_value).
2. Remplissez les valeurs. Les fonctions disponibles sont listées ci-dessous.

Les fonctions utilisées pour accéder à un tableau sont :

- add_assoc_long(return_value,key,l) - Ajoute une entrée associative avec la clé 'key' et la valeur 'l', de type long
- add_assoc_double(return_value,key,d) - Ajoute une entrée associative avec la clé 'key' et la valeur 'l', de type double
- add_assoc_string(return_value,key,str,duplicate)
- add_assoc_stringl(return_value,key,str,length,duplicate) spécifie la taille d'une chaîne
- add_index_long(return_value,index,l) - Ajoute une entrée d'index index' avec la valeur 'l', de type long
- add_index_double(return_value,index,d)
- add_index_string(return_value,index,str)
- add_index_stringl(return_value,index,str,length) - spécifie la longueur de la chaîne.
- add_next_index_long(return_value,l) - ajoute une entrée tableau, dans le prochain offset libre, de longueur 'l', de type long
- add_next_index_double(return_value,d)
- add_next_index_string(return_value,str)

- add_next_index_stringl(return_value,str,length) - spécifie la taille d'une chaîne

Utiliser la liste des ressources

PHP 3.0 dispose de standards pour traiter un certains nombre de ressources. Ils remplacent tous les listes de PHP 2.0.

Fonctions accessibles :

- php3_list_insert(ptr, type) - retourne l'identifiant 'id' de la nouvelle ressource insérée.
- php3_list_delete(id) - efface la ressource d'identifiant id
- php3_list_find(id,*type) - retourne le pointeur de la ressource d'identifiant id, et modifie le type 'type'

Typiquement, ces fonctions sont utilisées pour les pilotes SQL, mais elles peuvent servir n'importe quoi d'autre. Par exemple, conserver un pointeur de fichier.

La liste standard de code ressemble à ceci :

Exemple C-7. Ajouter une nouvelle ressource

```
RESOURCE *resource;
/* ...alloue de la mémoire pour la ressource, et l'acquiert ... */
/* Ajoute la nouvelle ressource dans la liste */
return_value->value.lval = php3_list_insert((void *) resource, LE_RESOURCE_TYPE);
return_value->type = IS_LONG;
```

Exemple C-8. Utiliser une ressource existante

```
pval *resource_id;
RESOURCE *resource;
int type;
convert_to_long(resource_id);
resource = php3_list_find(resource_id->value.lval, &type);
if (type != LE_RESOURCE_TYPE) {
    php3_error(E_WARNING,"resource index %d has the wrong type",resource_id->value.lval);
    RETURN_FALSE;
}
/* ...utiliser la ressource... */
```

Exemple C-9. Effacer une ressource existante

```
pval *resource_id;
RESOURCE *resource;
int type;
convert_to_long(resource_id);
php3_list_delete(resource_id->value.lval);
```

Les types de ressources doivent être enregistré dans le fichier php3_list.h, dans l'énumération list_entry_type. En plus, il faut penser à ajouter une fonction de terminaison, pour chaque type de ressource défini, dans le fichier list.c, pour la fonction list_entry_destructor() (même si vous n'avez rien de particulier à faire lors de la terminaison, vous devez au moins ajouter un cas vide.

Utiliser la table des ressources persistantes.

PHP 3.0 dispose d'une lieu de stockage des ressources persistantes (i.e., les ressources qui doivent être conservées d'un hit à l'autre). Le premier module a utiliser cette capacité a été MySQL, et mSQL suivi, ce qui fait que l'on peut se faire une impression du fonctionnement de cette fonction avec mysql.c. Les fonctions ressemblent à ceci :

php3_mysql_do_connect

php3_mysql_connect()
php3_mysql_pconnect()

L'idée conductrice de ces modules est la suivante :

1. Programmez tout votre module pour qu'il travaille avec les ressources standard, comme mentionné dans la section (9).
2. Ajoutez une autre fonction de connexion, qui vérifie d'abord que la ressource existe dans la liste des ressources persistantes. Si c'est le cas, enregistrez cette ressource comme pour les ressources standard (et grâce à la première étape, cela va fonctionner immédiatement). Si la ressource n'existe pas, créez la, ajoutez la à la liste de ressources persistantes, et ajoutez la à la liste de ressources, ce qui fait que le code va fonctionner, et que le prochain appel renverra une ressource existante. Vous devez enregistrer ces fonctions avec un type différent (LE_MYSQL_LINK pour les liens non persistants, et LE_MYSQL_PLINK pour les liens persistants).

Si vous jetez un oeil dans mysql.c, vous verrez que, hormis la fonction de connexion complexe, rien n'a du être changé dans le module.

La même interface existe pour la liste des ressources standard, et pour la liste des ressources persistantes, seule la 'list' est remplacée par 'plist':

- php3_plist_insert(ptr, type) - retourne l'identifiant 'id' de la nouvelle ressource insérée.
- php3_plist_delete(id) - efface la ressource d'identifiant id
- php3_plist_find(id,*type) - retourne le pointeur de la ressource d'identifiant id, et modifie le type 'type'

Cependant, il est probable que ces fonctions seront inutiles pour vous, lorsque vous essayerez d'implémentez un module persistant. Typiquement, on utiliser le fait que la liste de ressources persistantes est une table de hash. Par exemple, dans les modules MySQL/mSQL, lors d'un appel à pconnect(), la fonction construit une chaîne avec l'hôte/utilisateur/mot_de_passe, et l'utilise pour enregistrer dans la table de hash. Au prochain appel, avec les mêmes hôte/utilisateur/mot_de_passe, la même clé sera générée, et la ressource associée sera retrouvée.

Jusqu'à ce que la documentation s'étoffe, jetez un oeil aux fichiers mysql.c ou msql.c pour voir comment implémentez vos accès aux ressources persistantes.

Une chose importante à noter : les ressources qui sont enregistrées dans la liste de ressource persistante ne DOIVENT PAS être allouée avec le gestionnaire de mémoire PHP, c'est-à-dire qu'elles ne doivent pas être créée avec emalloc(), estrdup(), etc. Au contraire, il faut utiliser les fonctions standard malloc(), strdup(), etc. La raison est for simple : à la fin de la requête, la mémoire sera supprimée par le gestionnaire. Etant donné que les liens persistants doivent être conservés, il ne faut pas utiliser le gestionnaire de mémoire.

Lorsque vous enregistrez une ressource qui sera placé dans la liste de ressources persistantes, il faut ajouter les destructeurs dans les deux listes de ressources, persistantes ou pas. Le destructeur de la liste de ressources non persistantes ne doit rien faire du tout, tandis que celui de la liste de ressources persistantes doit libérer proprement toutes les ressources acquises (mémoire, lien SQL...). Commep pour les ressources non persistantes vous DEVEZ ajouter un destructeur, même s'il ne fait rien. N'oubliez pas que emalloc() et compagnie ne doivent pas être utilisé en conjonction avec la liste de ressources persistantes, et donc, vous ne devez pas utiliser efree() non plus.

Ajouter des directives de configuration à l'exécution

De nombreuses caractéristiques de PHP 3 peuvent être configurée à l'éxécution. Ces directives peuvent apparaître dans le fichier `php3.ini`, ou, dans le cas du module Apache, dans le fichier `.conf`. L'avantage de l'avoir dans le fichier `.conf`, est que ces caractéristiques peuvent être configurées dossier par dossier. Cela signifie qu'un dossier peut avoir un safe mode exec dir, tandis qu'un autre en aura un autre. Cette granularité de la configuration peut être extrêmement pratique lorsque le serveur supporte plusieurs serveurs virtuels.

Les étapes de configuration d'une nouvelle directive sont :

1. Ajouter la directive à la structure php3_ini_structure dans le fichier `mod_php3.h`.
2. Dans main.c, éditez la fonction php3_module_startup et ajoutez l'appel aproprié à cfg_get_string() ou cfg_get_long().
3. Ajoutez la directive, ses restrictions et un commentaire dans la structure php3_commands du fichier mod_php3.c. Notez la partie restrictions RSRC_CONF sont des directives qui ne peuvent être disponibles que dans le fichier de

configuration Apache. Toutes les directives OR_OPTIONS peuvent être placées n'importe où, y compris dans un fichier `.htaccess`.

4. Soit dans php3take1handler(), soit dans php3flaghandler(), ajoutez l'entrée appropriée pour votre directive.
5. Dans la section de configuration, de _php3_info(), dans le fichier functions/info.c, vous devez ajouter votre configuration.
6. Finalement, vous devez utiliser votre configuration quelque part. Elle sera accessible par php3_ini.directive.

Appeler des fonctions utilisateurs

Pour appeler des fonctions utilisateurs depuis une fonction interne, vous devez utiliser la fonction call_user_function().

call_user_function() retourne SUCCESS en cas de succès, et FAILURE en cas d'échec, ou si la fonction n'a pas été trouvée. Vous devez vérifier cette valeur. Si la réponse est SUCCESS, vous êtes responsable de la destruction de retval (ou alors, retournez la comme valeur de réponse de votre fonction). Si la réponse est FAILURE, la valeur de retval est indéfinie, et vous ne devez pas y toucher.

Toutes les fonctions internes qui appellent une fonction utilisateur, *DOIVENT* être réentrante. En particulier, elles ne doivent pas utiliser de valeurs globales, ou de variables statiques.

call_user_function() prend 6 arguments :

HashTable *function_table

La table de hash dans laquelle le fonction doit être recherchée.

pval *object

Un pointeur sur un objet sur lequel la fonction est invoquée. Il devrait être à `NULL`, si on invoque une fonction globale. Si il n'est pas à `NULL` (ie, il pointe sur un objet), l'argument function_table est ignorée, et la liste des fonctions sera lue dans l'objet, plutôt que dans l'argument. L'objet PEUT être modifié par la fonction qui est appelée (la fonction y aura accès via $this). Si, vous quelque raison, vous ne le voulez pas, envoyez une copie de l'objet à la place.

pval *function_name

Le nom de la fonction à appeler. Elle doit être de type pval, IS_STRING, avec les valeurs de function_name.str.val et function_name.str.len correctes. function_name est modifié par call_user_function() - il est converti en minuscule. Si vous voulez préserver la casse, envoyez une copie du nom de la fonction.

pval *retval

Un pointeur sur une structure pval, dans laquelle la valeur de retour de la fonction sera placée. La structure doit avoir été allouée au préalable, - call_user_function() ne l'allouera pas.

int param_count

Le nombre de paramètre passé à la fonction.

pval *params[]

Un tableau de pointeur sur les valeurs qui vont être passées comme arguments à la fonction. Le premier argument est à l'offset 0, le second à l'offset 1,... Le tableau est un tableau de pointeurs sur pval; Les pointeurs sont envoyés tels quels à la fonction, ce qui signifie que si la fonction modifie les arguments, les valeurs originales seront modifiées. Si vous voulez l'éviter, passez une copie à la place.

Rapport d'erreurs

Pour signaler les erreurs d'une fonction interne, vous devez appelez la fonction php3_error(). Cette fonction prend deux arguments au moins : le niveau de l'erreur, et le message d'erreur, sous forme de chaîne de caractères. Tous les arguments suivants sont des paramètres de formats de chaîne. Les niveaux d'erreurs sont :

E_NOTICE

Les notes ne sont pas affichées par défaut, et indique que le script a rencontré quelque chose qui peut être une erreur, mais peut aussi être un événement normal dans la vie du script. Par exemple, essayer d'accéder à une valeur qui n'a pas été déclarée, ou appeler **stat()** sur un fichier qui n'existe pas.

E_WARNING

Les alertes sont affichées par défaut, mais n'interrompent pas l'éxécution du script. Elles indiquent un problème qui doit être intercepté par le script avant que l'appel. Par exemple, appeler **ereg()** avec une regex invalide.

E_ERROR

Les erreurs sont aussi affichées par défaut, et l'exécution du script est interrompue. Elles indiquent des erreurs qui ne peuvent pas être ignorées, comme des problèmes d'allocation de mémoire, par exemple.

E_PARSE

Les erreurs d'analyse de doivent être générées que par l'analyseur. Elles ne sont citées ici que dans le but d'être exhaustif.

E_CORE_ERROR

Elles sont similaires aux erreurs E_ERROR, mais elles sont générées par le code de PHP. Les fonctions ne doivent pas générer ce genre d'erreur.

E_CORE_WARNING

Elles sont similaires à E_WARNING, mais elles sont générées par le code de PHP. Les fonctions ne doivent pas générer ce genre d'erreur.

E_COMPILE_ERROR

Elles sont similaires à E_ERROR, mais elles sont générées par Zend Scripting Engine. Les fonctions ne doivent pas générer ce genre d'erreur.

E_COMPILE_WARNING

Elles sont similaires à E_WARNING, mais elles sont générées par Zend Scripting Engine. Les fonctions ne doivent pas générer ce genre d'erreur.

E_USER_ERROR

E_USER_ERROR est comparable à E_ERROR. Elle est générée en PHP par l'utilisation de la fonction **trigger_error()**. Les fonctions ne doivent pas générer ce genre d'erreur.

E_USER_WARNING

E_USER_WARNING est comparable à E_WARNING. Elle est générée en PHP par l'utilisation de la fonction **trigger_error()**. Les fonctions ne doivent pas générer ce genre d'erreur.

E_USER_NOTICE

E_USER_WARNING est comparable à E_NOTICE. Elle est générée en PHP par l'utilisation de la fonction **trigger_error()**. Les fonctions ne doivent pas générer ce genre d'erreur.

Annexe D. Débuggeur PHP

A propos du débuggeur

PHP 3 inclut le support d'un serveur de débuggage.

PHP 4 n'inclut aucun support de débuggage.

Utiliser le débuggeur PHP

Le débuggeur PHP sert à repérer les bugs récalcitrants. Le débuggeur fonctionne en se connectant à un port TCP à chaque démarrage de PHP. Tous les messages d'erreur seront envoyés sur cette connexion. Cette page est faite pour un "serveur de débuggage", qui peut fonctionner avec un IDE ou un éditeur programmable (tel que Emacs).

Comment paramétrer le débuggeur :

1. Réservez un port TCP pour le débuggeur dans le fichier de configuration (debugger.port) et activez le (debugger.enabled).
2. Configurer un client TCP sur ce port (par exemple **socket -l -s 1400** sous UNIX).
3. Dans votre code, placez la ligne "debugger_on(*`host`*)", où *`host`* est l'IP ou le nom de l'hôte qui supporte le client TCP.

Désormais, toutes les alertes, notes, ... seront envoyées sur la socket client, *même si vous avez inactivé le rapport d'erreur avec error_reporting()*.

Protocole du débuggeur

Le protocole de débugage est ligne par ligne. Chaque ligne a un type *type* et plusieurs lignes composent un message Chaque message commence avec une ligne du type `start` et se termine avec une ligne de type `end`. PHP peut envoyer des lignes de plusieurs messages simultanément.

Voici un exemple de ligne à ce format :

`date time`
`host`(*`pid`*)
`type`:
`message-data`

`date`

Les dates sont au format ISO 8601 (*`yyyy-mm-dd`*)

`time`

Les heures, y compris les micro-secondes: *`hh`*:*`mm`*:*`uuuuuu`*

`host`

Le nom DNS ou adresse IP de l'hôte qui a généré l'erreur.

`pid`

PID (process id) sur l'hôte *`host`*, qui a généré l'erreur.

`type`

Type de la ligne. Indique au programme client comment traiter les données suivantes :

Tableau D-1. Types des lignes du Débuggeur

Nom	Signification
`start`	Indique au programme client que le message du débuggeur commence ici. Le contenu de *`data`* sera un type d'erreur, comme listé ci-dessous.
`message`	Le message d'erreur PHP.

Nom	Signification
`location`	Nom du fichier et numéro de ligne, où l'erreur est survenue. La première occurrence de `location` contiendra toujours la localisation générale. *`data`* contiendra : *`file`*`:`*`line`*. Il y a toujours une indication de `location` après un `message` et après chaque `function`.
`frames`	Nombre de frames dans le dump de la pile. S'il y a 4 frames, attendez vous à des informations sur 4 niveaux de fonctions. Si la ligne "frame" n'existe pas, la profondeur doit être 0 (une erreur est survenue au niveau général).
`function`	Nom de la fonction qui a généré l'erreur. Elle sera répétée à chaque niveau de la pile d'appel.
`end`	Indique au client que le message du débuggeur se termine ici.

`data`

Ligne de données.

Tableau D-2. Types d'erreur du débuggeur

Débuggeur	Interne PHP
alerte (warning)	E_WARNING
erreur	E_ERROR
analyse (parse)	E_PARSE
note (notice)	E_NOTICE
core-error	E_CORE_ERROR
core-warning	E_CORE_WARNING
inconnue	(toutes les autres)

Exemple D-1. Exemple de message du débuggeur

```
1998-04-05 23:27:400966 lucifer.guardian.no(20481) start: notice
1998-04-05 23:27:400966 lucifer.guardian.no(20481) message: Uninitialized variable
1998-04-05 23:27:400966 lucifer.guardian.no(20481) location: (NULL):7
1998-04-05 23:27:400966 lucifer.guardian.no(20481) frames: 1
1998-04-05 23:27:400966 lucifer.guardian.no(20481) function: display
1998-04-05 23:27:400966 lucifer.guardian.no(20481) location: /home/ssb/public_html/test.php3:10
1998-04-05 23:27:400966 lucifer.guardian.no(20481) end: notice
```

Annexe E. Mot réservés en PHP

Voici la liste des mots réservés en PHP :

- and.
- $argv.
- $argc.
- break.
- case.
- class.
- continue.
- default.
- do.
- **die()**.
- **echo()**.
- else.
- elseif.
- **empty()**.
- endfor.
- endforeach.
- endif.
- endswitch.
- endwhile.
- E_ALL.
- E_PARSE.
- E_ERROR.
- E_WARNING.
- **exit()**.
- extends.
- FALSE.
- for.
- foreach.
- function.
- $HTTP_COOKIE_VARS.
- $HTTP_GET_VARS.
- $HTTP_POST_VARS.
- $$HTTP_POST_FILES.
- $HTTP_ENV_VARS.
- $HTTP_SERVER_VARS.
- if.
- **include()**.
- **include_once()**.
- global.
- **list()**.
- new.

- not.
- NULL.
- or.
- parent.
- PHP_OS.
- $PHP_SELF.
- PHP_VERSION.
- **print()**.
- **require()**.
- **require_once()**.
- return.
- static.
- switch.
- stdClass.
- $this.
- TRUE.
- var.
- xor.
- **virtual()**.
- while.
- __FILE__.
- __LINE__.
- __sleep.
- __wakeup.

Annexe F. Types des ressources PHP

Voici la liste des fonctions qui créent, utilisent ou détruisent les ressources PHP. Vous pouvez déterminer si une variable contient une ressource avec la fonction **is_resource()**, et le type d'une ressource que vous utilisez avec la fonction **get_resource_type()**.

Tableau F-1. Types de ressource

Ressource	Construite par	Utilisé par	Détruite par	Définition
aspell	**aspell_new()**	**aspell_check()**, **aspell_check_raw()**, **aspell_suggest()**	None	Dictionnaire Aspell
bzip2	**bzopen()**	**bzerrno()**, **bzerror()**, **bzerrstr()**, **bzflush()**, **bzread()**, **bzwrite()**	**bzclose()**	Fichier bzip2
COM	**com_load()**	**com_invoke()**, **com_propget()**, **com_get()**, **com_propput()**, **com_set()**, **com_propput()**	None	Référence sur un object COM
VARIANT				

Ressource	Construite par	Utilisé par	Détruite par	Définition
cpdf	**cpdf_open()**	**cpdf_page_init()**, **cpdf_finalize_page()**, **cpdf_finalize()**, **cpdf_output_buffer()**, **cpdf_save_to_file()**, **cpdf_set_current_page()**, **cpdf_begin_text()**, **cpdf_end_text()**, **cpdf_show()**, **cpdf_show_xy()**, **cpdf_text()**, **cpdf_set_font()**, **cpdf_set_leading()**, **cpdf_set_text_rendering()**, **cpdf_set_horiz_scaling()**, **cpdf_set_text_rise()**, **cpdf_set_text_matrix()**, **cpdf_set_text_pos()**, **cpdf_set_text_pos()**, **cpdf_set_word_spacing()**, **cpdf_continue_text()**, **cpdf_stringwidth()**, **cpdf_save()**, **cpdf_translate()**, **cpdf_restore()**, **cpdf_scale()**, **cpdf_rotate()**, **cpdf_setflat()**, **cpdf_setlinejoin()**, **cpdf_setlinecap()**, **cpdf_setmiterlimit()**, **cpdf_setlinewidth()**, **cpdf_setdash()**, **cpdf_moveto()**, **cpdf_rmoveto()**, **cpdf_curveto()**, **cpdf_lineto()**, **cpdf_rlineto()**, **cpdf_circle()**, **cpdf_arc()**, **cpdf_rect()**, **cpdf_closepath()**, **cpdf_stroke()**, **cpdf_closepath_fill_stroke()**, **cpdf_fill_stroke()**, **cpdf_clip()**, **cpdf_fill()**, **cpdf_setgray_fill()**, **cpdf_setgray_stroke()**, **cpdf_setgray()**, **cpdf_setrgbcolor_fill()**, **cpdf_setrgbcolor_stroke()**, **cpdf_setrgbcolor()**, **cpdf_add_outline()**, **cpdf_set_page_animation()**, **cpdf_import_jpeg()**, **cpdf_place_inline_image()**, **cpdf_add_annotation()**	**cpdf_close()**	Document PDF avec la librairie CPDF

Ressource	Construite par	Utilisé par	Détruite par	Définition
cpdf outline				
curl	**curl_init()**	**curl_init()**, **curl_exec()**	**curl_close()**	Session Curl
dbm	**dbmopen()**	**dbmexists()**, **dbmfetch()**, **dbminsert()**, **dbmreplace()**, **dbmdelete()**, **dbmfirstkey()**, **dbmnextkey()**	**dbmclose()**	Lien vers une base de données DBM
dba	**dba_popen()**	**dba_delete()**, **dba_exists()**, **dba_fetch()**, **dba_firstkey()**, **dba_insert()**, **dba_nextkey()**, **dba_optimize()**, **dba_replace()**, **dba_sync()**	**dba_close()**	Lien vers une base de données DBA
dba persistent	**dba_open()**	**dba_delete()**, **dba_exists()**, **dba_fetch()**, **dba_firstkey()**, **dba_insert()**, **dba_nextkey()**, **dba_optimize()**, **dba_replace()**, **dba_sync()**	None	Lien persistant vers une base de données DBA
dbase	**dbase_open()**	**dbase_pack()**, **dbase_add_record()**, **dbase_replace_record()**, **dbase_delete_record()**, **dbase_get_record()**, **dbase_get_record_with_names()**, **dbase_numfields()**, **dbase_numrecords()**	**dbase_close()**	Lien vers une base de données dbase
dbx_link_object	**dbx_connect()**	**dbx_query()**	**dbx_close()**	Connexion dbx
dbx_result_object	**dbx_query()**	**()**	None	Résultat dbx
domxml attribute				
domxml document				
domxml node				
xpath context				
xpath object				
fbsql database	**fbsql_select_db()**	**()**	None	Base de données fbsql

Ressource	Construite par	Utilisé par	Détruite par	Définition
fbsql link	**fbsql_change_user()**, **fbsql_connect()**	**fbsql_autocommit()**, **fbsql_change_user()**, **fbsql_create_db()**, **fbsql_data_seek()**, **fbsql_db_query()**, **fbsql_drop_db()**, **()**, **fbsql_select_db()**, **fbsql_errno()**, **fbsql_error()**, **fbsql_insert_id()**, **fbsql_list_dbs()**	**fbsql_close()**	Lien vers une base de données fbsql
fbsql plink	**fbsql_change_user()**, **fbsql_pconnect()**	**fbsql_autocommit()**, **fbsql_change_user()**, **fbsql_create_db()**, **fbsql_data_seek()**, **fbsql_db_query()**, **fbsql_drop_db()**, **()**, **fbsql_select_db()**, **fbsql_errno()**, **fbsql_error()**, **fbsql_insert_id()**, **fbsql_list_dbs()**	None	Lien persistant vers une base de données fbsql
fbsql result	**fbsql_db_query()**, **fbsql_list_dbs()**, **fbsql_query()**, **fbsql_list_fields()**, **fbsql_list_tables()**, **fbsql_tablename()**	**fbsql_affected_rows()**, **fbsql_fetch_array()**, **fbsql_fetch_assoc()**, **fbsql_fetch_field()**, **fbsql_fetch_lengths()**, **fbsql_fetch_object()**, **fbsql_fetch_row()**, **fbsql_field_flags()**, **fbsql_field_name()**, **fbsql_field_len()**, **fbsql_field_seek()**, **fbsql_field_table()**, **fbsql_field_type()**, **fbsql_next_result()**, **fbsql_num_fields()**, **fbsql_num_rows()**, **fbsql_result()**, **fbsql_num_rows()**	**fbsql_free_result()**	Résultat fbsql
fdf	**fdf_open()**	**fdf_create()**, **fdf_save()**, **fdf_get_value()**, **fdf_set_value()**, **fdf_next_field_name()**, **fdf_set_ap()**, **fdf_set_status()**, **fdf_get_status()**, **fdf_set_file()**, **fdf_get_file()**, **fdf_set_flags()**, **fdf_set_opt()**, **fdf_set_submit_form_action()**, **fdf_set_javascript_action()**	**fdf_close()**	Fichier FDF

Ressource	Construite par	Utilisé par	Détruite par	Définition
ftp	**ftp_connect()**	**ftp_login()**, **ftp_pwd()**, **ftp_cdup()**, **ftp_chdir()**, **ftp_mkdir()**, **ftp_rmdir()**, **ftp_nlist()**, **ftp_rawlist()**, **ftp_systype()**, **ftp_pasv()**, **ftp_get()**, **ftp_fget()**, **ftp_put()**, **ftp_fput()**, **ftp_size()**, **ftp_mdtm()**, **ftp_rename()**, **ftp_delete()**, **ftp_site()**	**ftp_quit()**	Connexion FTP

Ressource	Construite par	Utilisé par	Détruite par	Définition
gd	**imagecreate()**, **imagecreatefromgif()**, **imagecreatefromjpeg()**, **imagecreatefrompng()**, **imagecreatefromwbmp()**, **imagecreatefromstring()**, **imagecreatetruecolor()**	**imagearc()**, **imagechar()**, **imagecharup()**, **imagecolorallocate()**, **imagecolorat()**, **imagecolorclosest()**, **imagecolorexact()**, **imagecolorresolve()**, **imagegammacorrect()**, **imagegammacorrect()**, **imagecolorset()**, **imagecolorsforindex()**, **imagecolorstotal()**, **imagecolortransparent()**, **imagecopy()**, **imagecopyresized()**, **imagedashedline()**, **imagefill()**, **imagefilledpolygon()**, **imagefilledrectangle()**, **imagefilltoborder()**, **imagegif()**, **imagepng()**, **imagejpeg()**, **imagewbmp()**, **imageinterlace()**, **imageline()**, **imagepolygon()**, **imagepstext()**, **imagerectangle()**, **imagesetpixel()**, **imagestring()**, **imagestringup()**, **imagesx()**, **imagesy()**, **imagettftext()**, **imagefilledarc()**, **imageellipse()**, **imagefilledellipse()**, **imagecolorclosestalpha()**, **imagecolorexactalpha()**, **imagecolorresolvealpha()**, **imagecopymerge()**, **imagecopymergegray()**, **imagecopyresampled()**, **imagetruecolortopalette()**, **imagesetbrush()**, **imagesettile()**, **imagesetthickness()**	**imagedestroy()**	Image GD
gd font	**imageloadfont()**	**imagechar()**, **imagecharup()**, **imagefontheight()**	None	Police pour GD

Ressource	Construite par	Utilisé par	Détruite par	Définition
gd PS encoding				
gd PS font	**imagepsloadfont()**	**imagepstext()**, **imagepsslantfont()**, **imagepsextendfont()**, **imagepsencodefont()**, **imagepsbbox()**	**imagepsfreefont()**	Police PostScript pour GD
GMP integer	**gmp_init()**	**gmp_intval()**, **gmp_strval()**, **gmp_add()**, **gmp_sub()**, **gmp_mul()**, **gmp_div_q()**, **gmp_div_r()**, **gmp_div_qr()**, **gmp_div()**, **gmp_mod()**, **gmp_divexact()**, **gmp_cmp()**, **gmp_neg()**, **gmp_abs()**, **gmp_sign()**, **gmp_fact()**, **gmp_sqrt()**, **gmp_sqrtrm()**, **gmp_perfect_square()**, **gmp_pow()**, **gmp_powm()**, **gmp_prob_prime()**, **gmp_gcd()**, **gmp_gcdext()**, **gmp_invert()**, **gmp_legendre()**, **gmp_jacobi()**, **gmp_random()**, **gmp_and()**, **gmp_or()**, **gmp_xor()**, **gmp_setbit()**, **gmp_clrbit()**, **gmp_scan0()**, **gmp_scan1()**, **gmp_popcount()**, **gmp_hamdist()**	None	Nombre GMP

Ressource	Construite par	Utilisé par	Détruite par	Définition
hyperwave document	**hw_cp()**, **hw_docbyanchor()**, **hw_getremote()**, **hw_getremotechildren()**	**hw_children()**, **hw_childrenobj()**, **hw_getparents()**, **hw_getparentsobj()**, **hw_getchildcoll()**, **hw_getchildcollobj()**, **hw_getremote()**, **hw_getsrcbydestobj()**, **hw_getandlock()**, **hw_gettext()**, **hw_getobjectbyquerycoll()**, **hw_getobjectbyquerycollobj()**, **hw_getchilddoccoll()**, **hw_getchilddoccollobj()**, **hw_getanchors()**, **hw_getanchorsobj()**, **hw_inscoll()**, **hw_pipedocument()**, **hw_unlock()**	**hw_deleteobject()**	Objet Hyperwave

Ressource	Construite par	Utilisé par	Détruite par	Définition
hyperwave link	**hw_connect()**	**hw_children()**, **hw_childrenobj()**, **hw_cp()**, **hw_deleteobject()**, **hw_docbyanchor()**, **hw_docbyanchorobj()**, **hw_errormsg()**, **hw_edittext()**, **hw_error()**, **hw_getparents()**, **hw_getparentsobj()**, **hw_getchildcoll()**, **hw_getchildcollobj()**, **hw_getremote()**, **hw_getremotechildren()**, **hw_getsrcbydestobj()**, **hw_getobject()**, **hw_getandlock()**, **hw_gettext()**, **hw_getobjectbyquery()**, **hw_getobjectbyqueryobj()**, **hw_getobjectbyquerycoll()**, **hw_getobjectbyquerycollobj()**, **hw_getchilddoccoll()**, **hw_getchilddoccollobj()**, **hw_getanchors()**, **hw_getanchorsobj()**, **hw_mv()**, **hw_incollections()**, **hw_info()**, **hw_inscoll()**, **hw_insdoc()**, **hw_insertdocument()**, **hw_insertobject()**, **hw_mapid()**, **hw_modifyobject()**, **hw_pipedocument()**, **hw_unlock()**, **hw_who()**, **hw_getusername()**	**hw_close()**, **hw_free_document()**	Lien vers un serveur Hyperwave

Ressource	Construite par	Utilisé par	Détruite par	Définition
hyperwave link persistent	**hw_pconnect()**	**hw_children()**, **hw_childrenobj()**, **hw_cp()**, **hw_deleteobject()**, **hw_docbyanchor()**, **hw_docbyanchorobj()**, **hw_errormsg()**, **hw_edittext()**, **hw_error()**, **hw_getparents()**, **hw_getparentsobj()**, **hw_getchildcoll()**, **hw_getchildcollobj()**, **hw_getremote()**, **hw_getremotechildren()**, **hw_getsrcbydestobj()**, **hw_getobject()**, **hw_getandlock()**, **hw_gettext()**, **hw_getobjectbyquery()**, **hw_getobjectbyqueryobj()**, **hw_getobjectbyquerycoll()**, **hw_getobjectbyquerycollobj()**, **hw_getchilddoccoll()**, **hw_getchilddoccollobj()**, **hw_getanchors()**, **hw_getanchorsobj()**, **hw_mv()**, **hw_incollections()**, **hw_info()**, **hw_inscoll()**, **hw_insdoc()**, **hw_insertdocument()**, **hw_insertobject()**, **hw_mapid()**, **hw_modifyobject()**, **hw_pipedocument()**, **hw_unlock()**, **hw_who()**, **hw_getusername()**	None	Lien persistant vers un serveur Hyperwave
icap	**icap_open()**	**icap_fetch_event()**, **icap_list_events()**, **icap_store_event()**, **icap_snooze()**, **icap_list_alarms()**, **icap_delete_event()**	**icap_close()**	Lien vers un serveur icap

Ressource	Construite par	Utilisé par	Détruite par	Définition
imap	**imap_open()**	**imap_append()**, **imap_body()**, **imap_check()**, **imap_createmailbox()**, **imap_delete()**, **imap_deletemailbox()**, **imap_expunge()**, **imap_fetchbody()**, **imap_fetchstructure()**, **imap_headerinfo()**, **imap_header()**, **imap_headers()**, **imap_listmailbox()**, **imap_getmailboxes()**, **imap_get_quota()**, **imap_status()**, **imap_listsubscribed()**, **imap_set_quota()**, **imap_set_quota()**, **imap_getsubscribed()**, **imap_mail_copy()**, **imap_mail_move()**, **imap_num_msg()**, **imap_num_recent()**, **imap_ping()**, **imap_renamemailbox()**, **imap_reopen()**, **imap_subscribe()**, **imap_undelete()**, **imap_unsubscribe()**, **imap_scanmailbox()**, **imap_mailboxmsginfo()**, **imap_fetchheader()**, **imap_uid()**, **imap_msgno()**, **imap_search()**, **imap_fetch_overview()**	**imap_close()**	Lien vers un serveur mail (IMAP, POP3)
imap chain persistent				
imap persistent				
ingres	**ingres_connect()**	**ingres_query()**, **ingres_num_rows()**, **ingres_num_fields()**, **ingres_field_name()**, **ingres_field_type()**, **ingres_field_nullable()**, **ingres_field_length()**, **ingres_field_precision()**, **ingres_field_scale()**, **ingres_fetch_array()**, **ingres_fetch_row()**, **ingres_fetch_object()**, **ingres_rollback()**, **ingres_commit()**, **ingres_autocommit()**	**ingres_close()**	Lien vers une base de données ingresII

Ressource	Construite par	Utilisé par	Détruite par	Définition
ingres persistent	**ingres_pconnect()**	**ingres_query()**, **ingres_num_rows()**, **ingres_num_fields()**, **ingres_field_name()**, **ingres_field_type()**, **ingres_field_nullable()**, **ingres_field_length()**, **ingres_field_precision()**, **ingres_field_scale()**, **ingres_fetch_array()**, **ingres_fetch_row()**, **ingres_fetch_object()**, **ingres_rollback()**, **ingres_commit()**, **ingres_autocommit()**	None	Lien persistant vers une base de données ingresII
interbase blob				
interbase link	**ibase_connect()**	**ibase_query()**, **ibase_prepare()**, **ibase_trans()**	**ibase_close()**	Lien vers une base de données Interbase
interbase link persistent	**ibase_pconnect()**	**ibase_query()**, **ibase_prepare()**, **ibase_trans()**	None	Lien persistant vers une base de données Interbase
interbase query	**ibase_prepare()**	**ibase_execute()**	**ibase_free_query()**	Requête Interbase
interbase result	**ibase_query()**	**ibase_fetch_row()**, **ibase_fetch_object()**, **ibase_field_info()**, **ibase_num_fields()**	**ibase_free_result()**	Résultat Interbase
interbase transaction	**ibase_trans()**	**ibase_commit()**	**ibase_rollback()**	Transaction Interbase
java				
ldap link	**ldap_connect()**, **ldap_search()**	**ldap_count_entries()**, **ldap_first_attribute()**, **ldap_first_entry()**, **ldap_get_attributes()**, **ldap_get_dn()**, **ldap_get_entries()**, **ldap_get_values()**, **ldap_get_values_len()**, **ldap_next_attribute()**, **ldap_next_entry()**	**ldap_close()**	Lien vers un serveur LDAP

Ressource	Construite par	Utilisé par	Détruite par	Définition
ldap result	**ldap_read()**	**ldap_add()**, **ldap_compare()**, **ldap_bind()**, **ldap_count_entries()**, **ldap_delete()**, **ldap_errno()**, **ldap_error()**, **ldap_first_attribute()**, **ldap_first_entry()**, **ldap_get_attributes()**, **ldap_get_dn()**, **ldap_get_entries()**, **ldap_get_values()**, **ldap_get_values_len()**, **ldap_get_option()**, **ldap_list()**, **ldap_modify()**, **ldap_mod_add()**, **ldap_mod_replace()**, **ldap_next_attribute()**, **ldap_next_entry()**, **ldap_mod_del()**, **ldap_set_option()**, **ldap_unbind()**	**ldap_free_result()**	Résultat de recherche LDAP
ldap result entry				
mcal	**mcal_open()**, **mcal_popen()**	**mcal_create_calendar()**, **mcal_rename_calendar()**, **mcal_rename_calendar()**, **mcal_delete_calendar()**, **mcal_fetch_event()**, **mcal_list_events()**, **mcal_append_event()**, **mcal_store_event()**, **mcal_delete_event()**, **mcal_list_alarms()**, **mcal_event_init()**, **mcal_event_set_category()**, **mcal_event_set_title()**, **mcal_event_set_description()**, **mcal_event_set_start()**, **mcal_event_set_end()**, **mcal_event_set_alarm()**, **mcal_event_set_class()**, **mcal_next_recurrence()**, **mcal_event_set_recur_none()**, **mcal_event_set_recur_daily()**, **mcal_event_set_recur_weekly()**, **mcal_event_set_recur_monthly_mday()**, **mcal_event_set_recur_monthly_wday()**, **mcal_event_set_recur_yearly()**, **mcal_fetch_current_stream_event()**, **mcal_event_add_attribute()**, **mcal_expunge()**	**mcal_close()**	Lien vers un serveur Mcal
SWFAction				
SWFBitmap				

Ressource	Construite par	Utilisé par	Détruite par	Définition
SWFButton				
SWFDisplayItem				
SWFFill				
SWFFont				
SWFGradient				
SWFMorph				
SWFMovie				
SWFShape				
SWFSprite				
SWFText				
SWFTextField				
mnogosearch agent				
mnogosearch result				
msql link	**msql_connect()**	**msql()**, **msql_create_db()**, **msql_createdb()**, **msql_drop_db()**, **msql_drop_db()**, **msql_select_db()**, **msql_select_db()**	**msql_close()**	Lien vers une base de données mSQL
msql link persistent	**msql_pconnect()**	**msql()**, **msql_create_db()**, **msql_createdb()**, **msql_drop_db()**, **msql_drop_db()**, **msql_select_db()**, **msql_select_db()**	None	Lien persistant vers une base de données mSQL
msql query	**msql_query()**	**msql()**, **msql_affected_rows()**, **msql_data_seek()**, **msql_dbname()**, **msql_fetch_array()**, **msql_fetch_field()**, **msql_fetch_object()**, **msql_fetch_row()**, **msql_fieldname()**, **msql_field_seek()**, **msql_fieldtable()**, **msql_fieldtype()**, **msql_fieldflags()**, **msql_fieldlen()**, **msql_num_fields()**, **msql_num_rows()**, **msql_numfields()**, **msql_numrows()**, **msql_result()**	**msql_free_result()**, **msql_free_result()**	Résultat mSQL
mssql link	**mssql_connect()**	**mssql_query()**, **mssql_select_db()**	**mssql_close()**	Lien vers une base de données Microsft SQL Server
mssql link persistent	**mssql_pconnect()**	**mssql_query()**, **mssql_select_db()**	None	Lien persistant vers une base de données Microsft SQL Server

Ressource	Construite par	Utilisé par	Détruite par	Définition
mssql result	**mssql_query()**	**mssql_data_seek()**, **mssql_fetch_array()**, **mssql_fetch_field()**, **mssql_fetch_object()**, **mssql_fetch_row()**, **mssql_field_length()**, **mssql_field_name()**, **mssql_field_seek()**, **mssql_field_type()**, **mssql_num_fields()**, **mssql_num_rows()**, **mssql_result()**	**mssql_free_result()**	Résultat Microsft SQL Server
mysql link	**mysql_connect()**	**mysql_affected_rows()**, **mysql_change_user()**, **mysql_create_db()**, **mysql_data_seek()**, **mysql_db_name()**, **mysql_db_query()**, **mysql_drop_db()**, **mysql_errno()**, **mysql_error()**, **mysql_insert_id()**, **mysql_list_dbs()**, **mysql_list_fields()**, **mysql_list_tables()**, **mysql_query()**, **mysql_result()**, **mysql_select_db()**, **mysql_tablename()**, **mysql_get_host_info()**, **mysql_get_proto_info()**, **mysql_get_server_info()**	**mysql_close()**	Lien vers une base de données MySQL
mysql link persistent	**mysql_pconnect()**	**mysql_affected_rows()**, **mysql_change_user()**, **mysql_create_db()**, **mysql_data_seek()**, **mysql_db_name()**, **mysql_db_query()**, **mysql_drop_db()**, **mysql_errno()**, **mysql_error()**, **mysql_insert_id()**, **mysql_list_dbs()**, **mysql_list_fields()**, **mysql_list_tables()**, **mysql_query()**, **mysql_result()**, **mysql_select_db()**, **mysql_tablename()**, **mysql_get_host_info()**, **mysql_get_proto_info()**, **mysql_get_server_info()**	None	Lien persistant vers une base de données MySQL

Ressource	Construite par	Utilisé par	Détruite par	Définition
mysql result	**mysql_db_query()**, **mysql_list_dbs()**, **mysql_list_fields()**, **mysql_list_tables()**, **mysql_query()**	**mysql_data_seek()**, **mysql_db_name()**, **mysql_fetch_array()**, **mysql_fetch_assoc()**, **mysql_fetch_field()**, **mysql_fetch_lengths()**, **mysql_fetch_object()**, **mysql_fetch_row()**, **mysql_fetch_row()**, **mysql_field_flags()**, **mysql_field_name()**, **mysql_field_len()**, **mysql_field_seek()**, **mysql_field_table()**, **mysql_field_type()**, **mysql_num_fields()**, **mysql_num_rows()**, **mysql_result()**, **mysql_tablename()**	**mysql_free_result()**	Résultat MySQL
oci8 collection				
oci8 connection	**ocilogon()**, **ociplogon()**, **ocinlogon()**	**ocicommit()**, **ociserverversion()**, **ocinewcursor()**, **ociparse()**, **ocierror()**	**ocilogoff()**	Lien vers une base de données Oracle
oci8 descriptor				
oci8 server				
oci8 session				
oci8 statement	**ocinewdescriptor()**	**ocirollback()**, **ocinewdescriptor()**, **ocirowcount()**, **ocidefinebyname()**, **ocibindbyname()**, **ociexecute()**, **ocinumcols()**, **ociresult()**, **ocifetch()**, **ocifetchinto()**, **ocifetchstatement()**, **ocicolumnisnull()**, **ocicolumnname()**, **ocicolumnsize()**, **ocicolumntype()**, **ocistatementtype()**, **ocierror()**	**ocifreestatement()**	Curseur Oracle

Ressource	Construite par	Utilisé par	Détruite par	Définition
odbc link	**odbc_connect()**	**odbc_autocommit()**, **odbc_commit()**, **odbc_error()**, **odbc_errormsg()**, **odbc_exec()**, **odbc_tables()**, **odbc_tableprivileges()**, **odbc_do()**, **odbc_prepare()**, **odbc_columns()**, **odbc_columnprivileges()**, **odbc_procedurecolumns()**, **odbc_specialcolumns()**, **odbc_rollback()**, **odbc_setoption()**, **odbc_gettypeinfo()**, **odbc_primarykeys()**, **odbc_foreignkeys()**, **odbc_procedures()**, **odbc_statistics()**	**odbc_close()**	Lien vers une base de données ODBC
odbc link persistent	**odbc_connect()**	**odbc_autocommit()**, **odbc_commit()**, **odbc_error()**, **odbc_errormsg()**, **odbc_exec()**, **odbc_tables()**, **odbc_tableprivileges()**, **odbc_do()**, **odbc_prepare()**, **odbc_columns()**, **odbc_columnprivileges()**, **odbc_procedurecolumns()**, **odbc_specialcolumns()**, **odbc_rollback()**, **odbc_setoption()**, **odbc_gettypeinfo()**, **odbc_primarykeys()**, **odbc_foreignkeys()**, **odbc_procedures()**, **odbc_statistics()**	None	Lien persistant vers une base de données ODBC
odbc result	**odbc_prepare()**	**odbc_binmode()**, **odbc_cursor()**, **odbc_execute()**, **odbc_fetch_into()**, **odbc_fetch_row()**, **odbc_field_name()**, **odbc_field_num()**, **odbc_field_type()**, **odbc_field_len()**, **odbc_field_precision()**, **odbc_field_scale()**, **odbc_longreadlen()**, **odbc_num_fields()**, **odbc_num_rows()**, **odbc_result()**, **odbc_result_all()**, **odbc_setoption()**	**odbc_free_result()**	Résultat ODBC

Ressource	Construite par	Utilisé par	Détruite par	Définition
velocis link				
velocis result				
OpenSSL key	**openssl_get_privatekey()**, **openssl_get_publickey()**	**openssl_sign()**, **openssl_seal()**, **openssl_open()**, **openssl_verify()**	**openssl_free_key()**	Clé de cryptage OpenSSL
OpenSSL X.509	**openssl_x509_read()**	**openssl_x509_parse()**, **openssl_x509_checkpurpose()**	**openssl_x509_free()**	Clé publique
oracle cursor	**ora_open()**	**ora_bind()**, **ora_columnname()**, **ora_columnsize()**, **ora_columntype()**, **ora_error()**, **ora_errorcode()**, **ora_exec()**, **ora_fetch()**, **ora_fetch_into()**, **ora_getcolumn()**, **ora_numcols()**, **ora_numrows()**, **ora_parse()**	**ora_close()**	Curseur oracle
oracle link	**ora_logon()**	**ora_do()**, **ora_error()**, **ora_errorcode()**, **ora_rollback()**, **ora_commitoff()**, **ora_commiton()**, **ora_open()**, **ora_commit()**	**ora_logoff()**	Lien vers une base de données oracle
oracle link persistent	**ora_plogon()**	**ora_do()**, **ora_error()**, **ora_errorcode()**, **ora_rollback()**, **ora_commitoff()**, **ora_commiton()**, **ora_open()**, **ora_commit()**	None	Lien persistant vers une base de données oracle

Ressource	Construite par	Utilisé par	Détruite par	Définition
pdf document	**pdf_new()**	**pdf_add_bookmark()**, **pdf_add_launchlink()**, **pdf_add_locallink()**, **pdf_add_note()**, **pdf_add_pdflink()**, **pdf_add_weblink()**, **pdf_arc()**, **pdf_attach_file()**, **pdf_begin_page()**, **pdf_circle()**, **pdf_clip()**, **pdf_closepath()**, **pdf_closepath_fill_stroke()**, **pdf_closepath_stroke()**, **pdf_concat()**, **pdf_continue_text()**, **pdf_curveto()**, **pdf_end_page()**, **pdf_endpath()**, **pdf_fill()**, **pdf_fill_stroke()**, **pdf_findfont()**, **pdf_get_buffer()**, **pdf_get_image_height()**, **pdf_get_image_width()**, **pdf_get_parameter()**, **pdf_get_value()**, **pdf_lineto()**, **pdf_moveto()**, **pdf_open_ccitt()**, **pdf_open_file()**, **pdf_open_image_file()**, **pdf_place_image()**, **pdf_rect()**, **pdf_restore()**, **pdf_rotate()**, **pdf_save()**, **pdf_scale()**, **pdf_setdash()**, **pdf_setflat()**, **pdf_setfont()**, **pdf_setgray()**, **pdf_setgray_fill()**, **pdf_setgray_stroke()**, **pdf_setlinecap()**, **pdf_setlinejoin()**, **pdf_setlinewidth()**, **pdf_setmiterlimit()**, **pdf_setpolydash()**, **pdf_setrgbcolor()**, **pdf_setrgbcolor_fill()**, **pdf_setrgbcolor_stroke()**, **pdf_set_border_color()**, **pdf_set_border_dash()**, **pdf_set_border_style()**, **pdf_set_char_spacing()**, **pdf_set_duration()**, **pdf_set_font()**, **pdf_set_horiz_scaling()**, **pdf_set_parameter()**, **pdf_set_text_pos()**, **pdf_set_text_rendering()**, **pdf_set_value()**, **pdf_set_word_spacing()**,	**pdf_close()**, **pdf_delete()**	Document PDF

Ressource	Construite par	Utilisé par	Détruite par	Définition
pdf image	**pdf_open_image()**, **pdf_open_image_file()**, **pdf_open_memory_image()**	**pdf_get_image_height()**, **pdf_get_image_width()**, **pdf_open_ccitt()**, **pdf_place_image()**	**pdf_close_image()**	Image dans un document PDF
pdf object				
pdf outline				
pgsql large object	**pg_getlastoid()**, **pg_loimport()**, **pg_loimport()**	**pg_loopen()**, **pg_getlastoid()**, **pg_locreate()**, **pg_loexport()**, **pg_loread()**, **pg_loreadall()**, **pg_lounlink()**, **pg_lowrite()**	**pg_loclose()**	Objet de grande taille PostGreSQL
pgsql link	**pg_connect()**	**pg_cmdtuples()**, **pg_dbname()**, **pg_end_copy()**, **pg_errormessage()**, **pg_host()**, **pg_locreate()**, **pg_loexport()**, **pg_loimport()**, **pg_loopen()**, **pg_lounlink()**, **pg_options()**, **pg_port()**, **pg_put_line()**, **pg_set_client_encoding()**, **pg_client_encoding()**, **pg_trace()**, **pg_untrace()**, **pg_tty()**	**pg_close()**	Lien vers une base de données PostGreSQL
pgsql link persistent	**pg_pconnect()**	**pg_cmdtuples()**, **pg_dbname()**, **pg_end_copy()**, **pg_errormessage()**, **pg_host()**, **pg_locreate()**, **pg_loexport()**, **pg_loimport()**, **pg_loopen()**, **pg_lounlink()**, **pg_options()**, **pg_port()**, **pg_put_line()**, **pg_set_client_encoding()**, **pg_client_encoding()**, **pg_trace()**, **pg_untrace()**, **pg_tty()**	None	Lien persistant vers une base de données PostGreSQL

Ressource	Construite par	Utilisé par	Détruite par	Définition
pgsql result	**pg_exec()**	**pg_fetch_array()**, **pg_fetch_object()**, **pg_fieldisnull()**, **pg_fetch_row()**, **pg_fieldname()**, **pg_fieldnum()**, **pg_fieldprtlen()**, **pg_fieldsize()**, **pg_fieldtype()**, **pg_getlastoid()**, **pg_numfields()**, **pg_result()**, **pg_numrows()**	**pg_freeresult()**	Résultat PostGreSQL
pgsql string				
printer				
printer brush				
printer font				
printer pen				
pspell	**pspell_new()**, **pspell_new_config()**, **pspell_new_personal()**	**pspell_add_to_personal()**, **pspell_add_to_session()**, **pspell_check()**, **pspell_clear_session()**, **pspell_config_ignore()**, **pspell_config_mode()**, **pspell_config_personal()**, **pspell_config_repl()**, **pspell_config_runtogether()**, **pspell_config_save_repl()**, **pspell_save_wordlist()**, **pspell_store_replacement()**, **pspell_suggest()**	None	Dictionnaire Pspell
pspell config	**pspell_config_create()**	**pspell_new_config()**	None	Configuration Pspell
Sablotron XSLT	**xslt_create()**	**xslt_closelog()**, **xslt_openlog()**, **xslt_run()**, **xslt_set_sax_handler()**, **xslt_errno()**, **xslt_error()**, **xslt_fetch_result()**, **xslt_free()**	**xslt_free()**	Analyseur XSLT
shmop	**shmop_open()**	**shmop_read()**, **shmop_write()**, **shmop_size()**, **shmop_delete()**	**shmop_close()**	
sockets file descriptor set	**socket()**	**accept_connect()**, **bind()**, **connect()**, **listen()**, **read()**, **write()**	**close()**	Socket
sockets i/o vector				

Ressource	Construite par	Utilisé par	Détruite par	Définition
dir	**dir()**	**readdir()**, **rewinddir()**	**closedir()**	Dossier
file	**fopen()**	**feof()**, **fflush()**, **fgetc()**, **fgetcsv()**, **fgets()**, **fgetss()**, **flock()**, **fpassthru()**, **fputs()**, **fwrite()**, **fread()**, **fseek()**, **ftell()**, **fstat()**, **ftruncate()**, **set_file_buffer()**, **rewind()**	**fclose()**	Fichier
pipe	**popen()**	**feof()**, **fflush()**, **fgetc()**, **fgetcsv()**, **fgets()**, **fgetss()**, **fpassthru()**, **fputs()**, **fwrite()**, **fread()**	**pclose()**	Processus
socket	**fsockopen()**	**fflush()**, **fgetc()**, **fgetcsv()**, **fgets()**, **fgetss()**, **fpassthru()**, **fputs()**, **fwrite()**, **fread()**	**fclose()**	Socket
stream				
sybase-db link	**sybase_connect()**	**sybase_query()**, **sybase_select_db()**	**sybase_close()**	Lien vers une base de données Sybase avec la librairie db
sybase-db link persistent	**sybase_pconnect()**	**sybase_query()**, **sybase_select_db()**	None	Lien persistant vers une base de données Sybase avec la librairie db
sybase-db result	**sybase_query()**	**sybase_data_seek()**, **sybase_fetch_array()**, **sybase_fetch_field()**, **sybase_fetch_object()**, **sybase_fetch_row()**, **sybase_field_seek()**, **sybase_num_fields()**, **sybase_num_rows()**, **sybase_result()**	**sybase_free_result()**	Résultat Sybase avec la librairie DB
sybase-ct link	**sybase_connect()**	**sybase_affected_rows()**, **sybase_query()**, **sybase_select_db()**	**sybase_close()**	Lien vers une base de données Sybase avec la librairie CT
sybase-ct link persistent	**sybase_pconnect()**	**sybase_affected_rows()**, **sybase_query()**, **sybase_select_db()**	None	Lien persistant vers une base de données Sybase avec la librairie CT
sybase-ct result	**sybase_query()**	**sybase_data_seek()**, **sybase_fetch_array()**, **sybase_fetch_field()**, **sybase_fetch_object()**, **sybase_fetch_row()**, **sybase_field_seek()**, **sybase_num_fields()**, **sybase_num_rows()**, **sybase_result()**	**sybase_free_result()**	Résultat Sybase avec la librairie CT
sysvsem	**sem_get()**	**sem_acquire()**	**sem_release()**	Sémaphore Système V

Ressource	Construite par	Utilisé par	Détruite par	Définition
sysvshm	**shm_attach()**	**shm_remove()**, **shm_put_var()**, **shm_get_var()**, **shm_remove_var()**	**shm_detach()**	Mémoire partagée Système V
wddx	**wddx_packet_start()**	**wddx_add_vars()**	**wddx_packet_end()**	Paquet wddx
xml	**xml_parser_create()**	**xml_set_object()**, **xml_set_element_handler()**, **xml_set_character_data_handler()**, **xml_set_processing_instruction_handler()**, **xml_set_default_handler()**, **xml_set_unparsed_entity_decl_handler()**, **xml_set_notation_decl_handler()**, **xml_set_external_entity_ref_handler()**, **xml_parse()**, **xml_get_error_code()**, **xml_error_string()**, **xml_get_current_line_number()**, **xml_get_current_column_number()**, **xml_get_current_byte_index()**, **xml_parse_into_struct()**, **xml_parser_set_option()**, **xml_parser_get_option()**	**xml_parser_free()**	Analyseur syntaxique XML
zlib	**gzopen()**	**gzeof()**, **gzgetc()**, **gzgets()**, **gzgetss()**, **gzpassthru()**, **gzputs()**, **gzread()**, **gzrewind()**, **gzseek()**, **gztell()**, **gzwrite()**	**gzclose()**	Fichier compressé Gzip
ZZIP Directory				
ZZIP Entry				

Annexe G. Liste d’alias

Voici la liste des aliases. Tous les alias sont listés ci-dessous. C'est une très mauvaise habitude d'utiliser ces alias, car ils risquent à tous moment de disparaître, rendus obsolète sans préavis, ou bien par un simple changement de nom, ce qui rend votre script inutilisable avec des versions plus récentes de PHP. Préférez toujours les versions officielles. En fait, cette liste est surtout destinées à ceux qui doivent mettre à jour leur scripts avec les syntaxes plus récentes.

Cette liste est en phase avec PHP 4.0.6.

Tableau G-1. Aliases

Alias	Fonction principale	Extension mère
rtrim	**chop()**	Syntaxe de base
close	**closedir()**	Syntaxe de base
sizeof	**count()**	Syntaxe de base
pos	**current()**	Syntaxe de base
fputs	**fwrite()**	Syntaxe de base
dir	**getdir()**	Syntaxe de base
show_source	**highlight_file ()**	Syntaxe de base
join	**implode()**	Syntaxe de base
ini_alter	**ini_set()**	Syntaxe de base
is_float	**is_double()**	Syntaxe de base
is_real	**is_double()**	Syntaxe de base
is_int	**is_long()**	Syntaxe de base
is_integer	**is_long()**	Syntaxe de base
is_writeable	**is_writable()**	Syntaxe de base
rewind	**rewinddir()**	Syntaxe de base
magic_quotes_runtime	**set_magic_quotes_runtime()**	Syntaxe de base
strchr	**strstr()**	Syntaxe de base
cv_add	**ccvs_add()**	CCVS
cv_auth	**ccvs_auth()**	CCVS
cv_command	**ccvs_command()**	CCVS
cv_count	**ccvs_count()**	CCVS
cv_delete	**ccvs_delete()**	CCVS
cv_done	**ccvs_done()**	CCVS
cv_init	**ccvs_init()**	CCVS
cv_lookup	**ccvs_lookup()**	CCVS
cv_new	**ccvs_new()**	CCVS
cv_report	**ccvs_report()**	CCVS
cv_return	**ccvs_return()**	CCVS
cv_reverse	**ccvs_reverse()**	CCVS
cv_sale	**ccvs_sale()**	CCVS
cv_status	**ccvs_status()**	CCVS
cv_textvalue	**ccvs_textvalue()**	CCVS
cv_void	**ccvs_void()**	CCVS
com_get	**com_propget()**	COM
com_propset	**com_propput()**	COM
com_set	**com_propput()**	COM
add_root	**domxml_add_root()**	DOM XML
attributes	**domxml_attributes()**	DOM XML

Alias	Fonction principale	Extension mère
name	**domxml_attrname()**	DOM XML
children	**domxml_children()**	DOM XML
children	**domxml_children()**	DOM XML
dumpmem	**domxml_dumpmem()**	DOM XML
domxml_getattr	**domxml_get_attribute()**	DOM XML
getattr	**domxml_get_attribute()**	DOM XML
get_attribute	**domxml_get_attribute()**	DOM XML
dtd	**domxml_intdtd()**	DOM XML
lastchild	**domxml_last_child()**	DOM XML
last_child	**domxml_last_child()**	DOM XML
new_child	**domxml_new_child()**	DOM XML
new_xmldoc	**domxml_new_xmldoc()**	DOM XML
node	**domxml_node()**	DOM XML
parent	**domxml_parent()**	DOM XML
root	**domxml_root()**	DOM XML
domxml_setattr	**domxml_set_attribute()**	DOM XML
setattr	**domxml_set_attribute()**	DOM XML
set_attribute	**domxml_set_attribute()**	DOM XML
set_content	**domxml_set_content()**	DOM XML
unlink	**domxml_unlink_node()**	DOM XML
xpath_eval	**xpath_eval()**	DOM XML
xpath_eval_expression	**xpath_eval_expression()**	DOM XML
xpath_init	**xpath_init()**	DOM XML
xpath_new_context	**xpath_new_context()**	DOM XML
xptr_new_context	**xpath_new_context()**	DOM XML
fbsql	**fbsql_db_query()**	FrontBase
imap_fetchtext	**imap_body()**	IMAP
imap_create	**imap_createmailbox()**	IMAP
imap_header	**imap_headerinfo()**	IMAP
imap_listmailbox	**imap_list()**	IMAP
imap_getmailboxes	**imap_list_full()**	IMAP
imap_scanmailbox	**imap_listscan()**	IMAP
imap_scan	**imap_listscan()**	IMAP
imap_listsubscribed	**imap_lsub()**	IMAP
imap_getsubscribed	**imap_lsub_full()**	IMAP
imap_rename	**imap_renamemailbox()**	IMAP
ldap_close	**ldap_unbind()**	LDAP
ming_setcubicthreshold	**ming_setCubicThreshold()**	Ming (Flash)
ming_setscale	**ming_setScale()**	Ming (Flash)
swfaction	**swfaction_init()**	Ming (Flash)
getheight	**swfbitmap_getHeight()**	Ming (Flash)
getwidth	**swfbitmap_getWidth()**	Ming (Flash)
swfbitmap	**swfbitmap_init()**	Ming (Flash)
addaction	**swfbutton_addAction()**	Ming (Flash)
addshape	**swfbutton_addShape()**	Ming (Flash)

Alias	Fonction principale	Extension mère
swfbutton	**swfbutton_init()**	Ming (Flash)
swfbutton_keypress	**swfbutton_keypress()**	Ming (Flash)
setaction	**swfbutton_setAction()**	Ming (Flash)
setdown	**swfbutton_setDown()**	Ming (Flash)
sethit	**swfbutton_setHit()**	Ming (Flash)
setover	**swfbutton_setOver()**	Ming (Flash)
setup	**swfbutton_setUp()**	Ming (Flash)
addcolor	**swfdisplayitem_addColor()**	Ming (Flash)
move	**swfdisplayitem_move()**	Ming (Flash)
moveto	**swfdisplayitem_moveTo()**	Ming (Flash)
multcolor	**swfdisplayitem_multColor()**	Ming (Flash)
rotate	**swfdisplayitem_rotate()**	Ming (Flash)
rotateto	**swfdisplayitem_rotateTo()**	Ming (Flash)
scale	**swfdisplayitem_scale()**	Ming (Flash)
scaleto	**swfdisplayitem_scaleTo()**	Ming (Flash)
setdepth	**swfdisplayitem_setDepth()**	Ming (Flash)
setmatrix	**swfdisplayitem_setMatrix()**	Ming (Flash)
setname	**swfdisplayitem_setName()**	Ming (Flash)
setratio	**swfdisplayitem_setRatio()**	Ming (Flash)
skewx	**swfdisplayitem_skewX()**	Ming (Flash)
skewxto	**swfdisplayitem_skewXTo()**	Ming (Flash)
skewy	**swfdisplayitem_skewY()**	Ming (Flash)
skewyto	**swfdisplayitem_skewYTo()**	Ming (Flash)
swffill	**swffill_init()**	Ming (Flash)
moveto	**swffill_moveTo()**	Ming (Flash)
rotateto	**swffill_rotateTo()**	Ming (Flash)
scaleto	**swffill_scaleTo()**	Ming (Flash)
skewxto	**swffill_skewXTo()**	Ming (Flash)
skewyto	**swffill_skewYTo()**	Ming (Flash)
getascent	**swffont_getAscent()**	Ming (Flash)
getdescent	**swffont_getDescent()**	Ming (Flash)
getleading	**swffont_getLeading()**	Ming (Flash)
getwidth	**swffont_getWidth()**	Ming (Flash)
swffont	**swffont_init()**	Ming (Flash)
addentry	**swfgradient_addEntry()**	Ming (Flash)
swfgradient	**swfgradient_init()**	Ming (Flash)
getshape1	**swfmorph_getShape1()**	Ming (Flash)
getshape2	**swfmorph_getShape2()**	Ming (Flash)
swfmorph	**swfmorph_init()**	Ming (Flash)
add	**swfmovie_add()**	Ming (Flash)
swfmovie	**swfmovie_init()**	Ming (Flash)
labelframe	**swfmovie_labelFrame()**	Ming (Flash)
nextframe	**swfmovie_nextFrame()**	Ming (Flash)
output	**swfmovie_output()**	Ming (Flash)
remove	**swfmovie_remove()**	Ming (Flash)

Alias	Fonction principale	Extension mère
save	**swfmovie_save()**	Ming (Flash)
savetofile	**swfmovie_saveToFile()**	Ming (Flash)
setbackground	**swfmovie_setBackground()**	Ming (Flash)
setdimension	**swfmovie_setDimension()**	Ming (Flash)
setframes	**swfmovie_setFrames()**	Ming (Flash)
setrate	**swfmovie_setRate()**	Ming (Flash)
streammp3	**swfmovie_streamMp3()**	Ming (Flash)
addfill	**swfshape_addfill()**	Ming (Flash)
drawarc	**swfshape_drawarc()**	Ming (Flash)
drawcircle	**swfshape_drawcircle()**	Ming (Flash)
drawcubic	**swfshape_drawcubic()**	Ming (Flash)
drawcubicto	**swfshape_drawcubicto()**	Ming (Flash)
drawcurve	**swfshape_drawcurve()**	Ming (Flash)
drawcurveto	**swfshape_drawcurveto()**	Ming (Flash)
drawglyph	**swfshape_drawglyph()**	Ming (Flash)
drawline	**swfshape_drawline()**	Ming (Flash)
drawlineto	**swfshape_drawlineto()**	Ming (Flash)
swfshape	**swfshape_init()**	Ming (Flash)
movepen	**swfshape_movepen()**	Ming (Flash)
movepento	**swfshape_movepento()**	Ming (Flash)
setleftfill	**swfshape_setleftfill()**	Ming (Flash)
setline	**swfshape_setline()**	Ming (Flash)
setrightfill	**swfshape_setrightfill()**	Ming (Flash)
add	**swfsprite_add()**	Ming (Flash)
swfsprite	**swfsprite_init()**	Ming (Flash)
labelframe	**swfsprite_labelFrame()**	Ming (Flash)
nextframe	**swfsprite_nextFrame()**	Ming (Flash)
remove	**swfsprite_remove()**	Ming (Flash)
setframes	**swfsprite_setFrames()**	Ming (Flash)
addstring	**swftext_addString()**	Ming (Flash)
getascent	**swftext_getAscent()**	Ming (Flash)
getdescent	**swftext_getDescent()**	Ming (Flash)
getleading	**swftext_getLeading()**	Ming (Flash)
getwidth	**swftext_getWidth()**	Ming (Flash)
swftext	**swftext_init()**	Ming (Flash)
moveto	**swftext_moveTo()**	Ming (Flash)
setcolor	**swftext_setColor()**	Ming (Flash)
setfont	**swftext_setFont()**	Ming (Flash)
setheight	**swftext_setHeight()**	Ming (Flash)
setspacing	**swftext_setSpacing()**	Ming (Flash)
addstring	**swftextfield_addString()**	Ming (Flash)
align	**swftextfield_align()**	Ming (Flash)
swftextfield	**swftextfield_init()**	Ming (Flash)
setbounds	**swftextfield_setBounds()**	Ming (Flash)
setcolor	**swftextfield_setColor()**	Ming (Flash)

Alias	Fonction principale	Extension mère
setfont	**swftextfield_setFont()**	Ming (Flash)
setheight	**swftextfield_setHeight()**	Ming (Flash)
setindentation	**swftextfield_setIndentation()**	Ming (Flash)
setleftmargin	**swftextfield_setLeftMargin()**	Ming (Flash)
setlinespacing	**swftextfield_setLineSpacing()**	Ming (Flash)
setmargins	**swftextfield_setMargins()**	Ming (Flash)
setname	**swftextfield_setName()**	Ming (Flash)
setrightmargin	**swftextfield_setRightMargin()**	Ming (Flash)
msql_affected_rows	**msql_affected_rows()**	mSQL
msql_createdb	**msql_create_db()**	mSQL
msql	**msql_db_query()**	mSQL
msql_dropdb	**msql_drop_db()**	mSQL
msql_fieldflags	**msql_field_flags()**	mSQL
msql_fieldlen	**msql_field_len()**	mSQL
msql_fieldname	**msql_field_name()**	mSQL
msql_fieldtable	**msql_field_table()**	mSQL
msql_fieldtype	**msql_field_type()**	mSQL
msql_freeresult	**msql_free_result()**	mSQL
msql_listdbs	**msql_list_dbs()**	mSQL
msql_listfields	**msql_list_fields()**	mSQL
msql_listtables	**msql_list_tables()**	mSQL
msql_numfields	**msql_num_fields()**	mSQL
msql_numrows	**msql_num_rows()**	mSQL
msql_dbname	**msql_result()**	mSQL
msql_tablename	**msql_result()**	mSQL
msql_selectdb	**msql_select_db()**	mSQL
msql_regcase	**sql_regcase()**	mSQL
i18n_convert	**mb_convert_encoding()**	Chaînes de caractères multi-octets
i18n_ja_jp_hantozen	**mb_convert_kana()**	Chaînes de caractères multi-octets
i18n_mime_header_decode	**mb_decode_mimeheader()**	Chaînes de caractères multi-octets
i18n_discover_encoding	**mb_detect_encoding()**	Chaînes de caractères multi-octets
i18n_mime_header_encode	**mb_encode_mimeheader()**	Chaînes de caractères multi-octets
i18n_http_input	**mb_http_input()**	Chaînes de caractères multi-octets
i18n_http_output	**mb_http_output()**	Chaînes de caractères multi-octets
i18n_internal_encoding	**mb_internal_encoding()**	Chaînes de caractères multi-octets
mbstrcut	**mb_strcut()**	Chaînes de caractères multi-octets
mbstrlen	**mb_strlen()**	Chaînes de caractères multi-octets
mbstrpos	**mb_strpos()**	Chaînes de caractères multi-octets
mbstrrpos	**mb_strrpos()**	Chaînes de caractères multi-octets
mbsubstr	**mb_substr()**	Chaînes de caractères multi-octets
mysql_createdb	**mysql_create_db()**	MySQL
mysql	**mysql_db_query()**	MySQL
mysql_dropdb	**mysql_drop_db()**	MySQL
mysql_fieldflags	**mysql_field_flags()**	MySQL
mysql_fieldlen	**mysql_field_len()**	MySQL

Alias	Fonction principale	Extension mère
mysql_fieldname	**mysql_field_name()**	MySQL
mysql_fieldtable	**mysql_field_table()**	MySQL
mysql_fieldtype	**mysql_field_type()**	MySQL
mysql_freeresult	**mysql_free_result()**	MySQL
mysql_listdbs	**mysql_list_dbs()**	MySQL
mysql_listfields	**mysql_list_fields()**	MySQL
mysql_listtables	**mysql_list_tables()**	MySQL
mysql_numfields	**mysql_num_fields()**	MySQL
mysql_numrows	**mysql_num_rows()**	MySQL
mysql_db_name	**mysql_result()**	MySQL
mysql_dbname	**mysql_result()**	MySQL
mysql_tablename	**mysql_result()**	MySQL
mysql_selectdb	**mysql_select_db()**	MySQL
oci8close	**ocicloselob()**	oci8
oci8append	**ocicollappend()**	oci8
oci8assign	**ocicollassign()**	oci8
oci8assignelem	**ocicollassignelem()**	oci8
oci8getelem	**ocicollgetelem()**	oci8
oci8max	**ocicollmax()**	oci8
oci8size	**ocicollsize()**	oci8
oci8trim	**ocicolltrim()**	oci8
oci8free	**ocifreecoll()**	oci8
oci8free	**ocifreedesc()**	oci8
oci8ocifreecursor	**ocifreestatement()**	oci8
oci8load	**ociloadlob()**	oci8
oci8save	**ocisavelob()**	oci8
oci8savefile	**ocisavelobfile()**	oci8
oci8writetofile	**ociwritelobtofile()**	oci8
oci8writetemporary	**ociwritetemporarylob()**	oci8
odbc_do	**odbc_exec()**	oci8
odbc_field_precision	**odbc_field_len()**	oci8
pdf_add_outline	**pdf_add_bookmark()**	PDF
pg_clientencoding	**pg_client_encoding()**	PostgreSQL
pg_setclientencoding	**pg_set_client_encoding()**	PostgreSQL
recode	**recode_string()**	GNU Recode
orbit_caught_exception	**satellite_caught_exception()**	Satellite
orbit_exception_id	**satellite_exception_id()**	Satellite
orbit_exception_value	**satellite_exception_value()**	Satellite
orbit_get_repository_id	**satellite_get_repository_id()**	Satellite
orbit_load_idl	**satellite_load_idl()**	Satellite
snmpwalkoid	**snmprealwalk()**	SNMP
mssql_affected_rows	**sybase_affected_rows()**	Sybase
mssql_close	**sybase_close()**	Sybase
mssql_connect	**sybase_connect()**	Sybase
mssql_data_seek	**sybase_data_seek()**	Sybase

Alias	Fonction principale	Extension mère
mssql_fetch_array	**sybase_fetch_array()**	Sybase
mssql_fetch_field	**sybase_fetch_field()**	Sybase
mssql_fetch_object	**sybase_fetch_object()**	Sybase
mssql_fetch_row	**sybase_fetch_row()**	Sybase
mssql_field_seek	**sybase_field_seek()**	Sybase
mssql_free_result	**sybase_free_result()**	Sybase
mssql_get_last_message	**sybase_get_last_message()**	Sybase
mssql_min_error_severity	**sybase_min_error_severity()**	Sybase
mssql_min_message_severity	**sybase_min_message_severity()**	Sybase
mssql_num_fields	**sybase_num_fields()**	Sybase
mssql_num_rows	**sybase_num_rows()**	Sybase
mssql_pconnect	**sybase_pconnect()**	Sybase
mssql_query	**sybase_query()**	Sybase
mssql_result	**sybase_result()**	Sybase
mssql_select_db	**sybase_select_db()**	Sybase
mssql_affected_rows	**sybase_affected_rows()**	Sybase
mssql_close	**sybase_close()**	Sybase
mssql_connect	**sybase_connect()**	Sybase
mssql_data_seek	**sybase_data_seek()**	Sybase
mssql_fetch_array	**sybase_fetch_array()**	Sybase
mssql_fetch_field	**sybase_fetch_field()**	Sybase
mssql_fetch_object	**sybase_fetch_object()**	Sybase
mssql_fetch_row	**sybase_fetch_row()**	Sybase
mssql_field_seek	**sybase_field_seek()**	Sybase
mssql_free_result	**sybase_free_result()**	Sybase
mssql_get_last_message	**sybase_get_last_message()**	Sybase
mssql_min_client_severity	**sybase_min_client_severity()**	Sybase
mssql_min_server_severity	**sybase_min_server_severity()**	Sybase
mssql_num_fields	**sybase_num_fields()**	Sybase
mssql_num_rows	**sybase_num_rows()**	Sybase
mssql_pconnect	**sybase_pconnect()**	Sybase
mssql_query	**sybase_query()**	Sybase
mssql_result	**sybase_result()**	Sybase
mssql_select_db	**sybase_select_db()**	Sybase
gzputs	**gzwrite()**	Zlib