Sommaire

PHP 4.0 Manuel de Référence

1 Préface

PHP, signifie "PHP: Hypertext Preprocessor" (Preprocesseur HyperTexte), est un langage de script HTML La plupart de sa syntaxe est empruntée aux langages C, Java et Perl, mais y ajoute plusieurs fonctionnalités uniques. Le but de ce langage est de permettre aux développeurs web de concevoir rapidement des sites, aux pages dynamiques.

1.1 A propos de ce manuel

Ce manuel est écrit en **SGML** en utilisant DocBook DTD, avec **DSSSL** (Document Style and Semantics Specification Language) pour le formattage. Les utilitaires utilisés pour générer le format **HTML**, **TeX** et **RTF** sont Jade, écrit James Clark et The Modular DocBook Stylesheets écrit par Norman Walsh. La documentation PHP a été assemblée par texi.
Ce manuel a été traduit en Français par php@nexen.net Il a été généré à partir de la documentation en Anglais originale du PHP documentation Group, au format XML, grâce à une version adaptée de texi.

2 Preface

2.1 Les auteurs

Stig Sæther Bakken Alexander Aulbach Egon Schmid Jim Winstead Lars Torben Wilson Rasmus Lerdorf Zeev Suraski Stig Sæther Bakken Email: stig@php.net
1997 1998 1999 2000 **the PHP Documentation Group**

2.2 Copyright

3 Copyright, distribution, historique

4 Installation

4.1 Télécharger la dernière version

Le code source ainsi que des binaires pour certaines plates-formes (notamment Windows), sont disponibles à l'adresse suivante: http://www.php.net/ .

4.2 Installation sous UNIX

Ce chapitre va vous aider lors de la configuration et de l'installation du PHP. Les connaissances requises sont les suivantes :

- Connaissances basiques d'UNIX (savoir faire un "make" et utiliser un compilateur C)
- Avoir un compilateur C ANSI installé
- Avoir installé un serveur web

4.2.1 Installation rapide (Version Module Apache)

1. gunzip apache_1.3.x.tar.gz
2. tar xvf apache_1.3.x.tar
3. gunzip php-3.0.x.tar.gz
4. tar xvf php-3.0.x.tar
5. cd apache_1.3.x
6. ./configure --prefix=/www
7. cd ../php-3.0.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 --prefix=/www --activate-module=src/modules/php3/libphp3.a
13. make
14. make install

A la place de cette étape, vous pouvez simplement écraser le binaire httpd. Assurez-vous d'avoir bien arrêté le démon d'abord.

15. cd ../php-3.0.x
16. cp php3.ini-dist /usr/local/lib/php3.ini

Vous pouvez éditer le fichier de configuration /usr/local/lib/php3.ini.
Si vous préférez installer le fichier dans un autre répertoire, il faut utiliser l'option de configuration --with-config-file-path=/path à l'étape 8.

17. Editez le fichier de configuration apache httpd.conf ou srm.conf et ajoutez :

AddType application/x-httpd-php3 .php3

Ici, il faut choisir l'extension que vous souhaitez donner au fichier php.
.php est simplement celle que nous suggérons.

18. Utilisez la procédure normale afin de démarrer le serveur Apache. (Vous devez impérativement arrêter et redémarrer le serveur Apache, et pas seulement le relancer à l'aide d'un signal HUP ou USR1).

4.2.2 Configuration

Il y a deux moyens de configurer PHP.

- Utilisation du script "setup" qui est fourni avec la distribution PHP. Ce script vous pose une série de questions (comme le script d'installation de PHP/FI 2.0) et lance le script "configure" à la fin. Afin de lancer le script, tapez ./setup.
  Ce script va aussi créer un fichier appelé "do-conf", qui contient les options de configuration. Vous pouvez éditer ce fichier afin de modifier certaines options sans avoir à réexécuter la totalité du script "setup". Dans ce cas là, tapez ./do-conf afin de lancer le script "configure" avec les nouvelles options.

- Lancez le script "configure" manuellement. Pour voir toutes les options de configuration disponibles, tapez ./configure --help.

Tous les détails à propos des différentes options de configuration sont regroupés ici:

4.2.3 Module Apache

Pour compiler PHP comme un module Apache, répondre "yes" à la question "Build as an Apache module ?" (cela correspond à l'option de configuration 4.2.3 Module Apache=*DIR*) et spécifie la racine de la distribution Apache. Si vous avez décompressé Apache dans le répertoire `/usr/local/www/apache_1.2.4', c'est la racine de la distribution Apache. Le répertoire par défaut est : `/usr/local/etc/httpd'.

4.2.4 Module 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 4.2.4 Module 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.

4.2.5 `/usr/local/src/fhttpd'

Par défaut, PHP est compilé comme une CGI. Si vous voulez que votre serveur web supporte le PHP, compiler le PHP comme une 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 6 Sécurité si vous souhaitez utiliser cette solution.

4.2.6 Configuration pour le support des bases de données

PHP supporte de nombreuses bases de données (et aussi ODBC):

4.2.6.1 Adabas D

--with-adabas=*DIR* Configure PHP pour le support des bases de données Adabas D. Le paramètre est le répertoire d'installation de la base de données et par défaut `/usr/local/adabasd'.

Adabas home page

4.2.6.2 dBase

--with-dbase Configure PHP pour le support des bases de données dbase. Aucune librairie n'est nécessaire.

4.2.6.3 filePro

--with-filepro Configure PHP pour le support des bases de données filePro. Aucune librairie supplémentaire n'est nécessaire.

4.2.6.4 mSQL

--with-msql=*DIR* Compile PHP pour le support des bases de données mSQL. Le paramètre est le répertoire d'installation de la base de données et par défaut c'est `/usr/local/Hughes', c'est à dire, le répertoire par défaut oú est installé mSQL 2.0. configure détecte automatiquement quelle version de mSQL est installée. PHP supporte aussi bien la version 1.0 que la version 2.0, mais si vous compilez PHP avec mSQL 1.0, vous pourrez ne accéder qu'à mSQL 1.0, et vice-versa.

Voir aussi 7.1.7 Directives de configuration mSQL au chapitre 7.1 Le fichier de configuration.

mSQL home page

4.2.6.5 MySQL

--with-mysql=*DIR* Compile PHP pour le support des bases de données MySQL. Le paramètre est le répertoire d'installation de la base de données et par défaut `/usr/local'. C'est le répertoire par défaut oú est installé MySQL.

Voir aussi 7.1.6 MySQL Configuration Directives au chapitre 7.1 Le fichier de configuration.

Home page MySQL

4.2.6.6 iODBC

--with-iodbc=*DIR* Compile PHP pour le support des bases de données iODBC. Cette fonction a été développée au départ pour supporter "iODBC Driver Manager", un pilote ODBC qui fonctionne sous la plupart des versions d'UNIX. Le paramètre est le répertoire d'installation de la base de données et par défaut : `/usr/local'.

Home page FreeODBC ou home page iODBC

4.2.6.7 OpenLink ODBC

--with-openlink=*DIR* Compile PHP pour le support des bases de données OpenLink ODBC. Le paramètre est le répertoire d'installation de la base de données et par défaut `/usr/local/openlink'.

Home page OpenLink Software

4.2.6.8 Oracle

--with-oracle=*DIR* Compile PHP pour le support des bases de données Oracle. Cette option fonctionne avec les versions 7.0 à 7.3 d'Oracle. Le paramètre est le répertoire d'installation de la base de données et par défaut : **ORACLE_HOME**. Vous n'avez pas à spécifier ce paramètre si votre base de données Oracle est déjà installée.
Home page Oracle

4.2.6.9 PostgreSQL

--with-pgsql=*DIR* Compile PHP pour le support des bases de données PostgreSQL. Le paramètre est le répertoire d'installation de la base de données PostgreSQL et par défaut`/usr/local/pgsql'.
Voir aussi 7.1.8 Directives de configuration Postgres dans le fichier 7.1 Le fichier de configuration.
Home page PostgreSQL

4.2.6.10 Solid

--with-solid=*DIR* Compile PHP pour le support des bases de données Solid. Le paramètre est le répertoire d'installation de la base de données et par défaut `/usr/local/solid'.
Home page Solid

4.2.6.11 Sybase

--with-sybase=*DIR* Compile PHP pour le support des bases de données Sybase. Le paramètre est le répertoire d'installation de la base de données et par défaut `/home/sybase'.
Voir aussi 7.1.9 Directives de configuration Sybase dans le fichier 7.1 Le fichier de configuration.
Home page Sybase

4.2.6.12 Sybase-CT

--with-sybase-ct=*DIR* Compile PHP pour le support des bases de donnés Sybase-CT. Le paramètre est le répertoire d'installation de la base de données Sybase-CT et par défaut `/home/sybase'.
Voir aussi 7.1.10 Sybase-CT Configuration Directives dans le 7.1 Le fichier de configuration.

4.2.6.13 Velocis

--with-velocis=*DIR* Compile PHP pour le support des bases de données Velocis. Le paramètre est le répertoire d'installation de la base de données Velocis et par défaut `/usr/local/velocis'.
Home page Velocis

4.2.6.14 A custom ODBC library

--with-custom-odbc=*DIR* Compile PHP pour le support d'une autre libraire ODBC. Le paramètre est le répertoire d'installation de la base de données et par défaut `/usr/local'.
Cette option implique que vous ayez défini la variable CUSTOM_ODBC_LIBS avant de lancer le script de configuration. Vous devez aussi avoir un fichier odbc.h quelque part dans votre répertoire d'installation. Si vous n'en avez pas, créez en un et incluez le dans vos fichiers d'entete. Vos entêtes demanderons sûrement des définitions supplémentaires, notamment en cas de base de données multiplateforme. Définissez les dans la variable CFLAGS.
Par exemple, vous pouvez utiliser Sybase SQL n'import oú sous QNX en définissant la variable CFLAGS comme suit: CFLAGS=-DODBC_QNX LDFLAGS=-lunix CUSTOM_ODBC_LIBS="-ldblib -lodbc" ./configure --with-custom-odbc=/usr/lib/sqlany50

4.2.6.15 ODBC unifié

--disable-unified-odbc Désactive le module ODBC unifié, qui est une interface commune à toutes les bases de données dotées d'une interface ODBC tel que Solid et Adabas D. Cette interface fonctionne aussi avec les librairies classiques ODBC. Elle a été testé avec iODBC, Solid, Adabas D et Sybase SQL, sur tous les types d'OS. Cela implique qu'un et un seul des modules oú le module Velocis est utilisé ou bien une librairie personnelle ODBC. Cette option n'est valide que si une des options suivantes est utilisée: 4.2.6.6 iODBC, 4.2.6.10 Solid, 4.2.6.1 Adabas D, 4.2.6.13 Velocis, or 4.2.6.1 Adabas D. Voir aussi 7.1.14 Directives de configuration du driver ODBC unifié dans le 7.1 Le fichier de configuration.

4.2.6.16 LDAP

--with-ldap=*DIR* Ajoute le support **LDAP**(Lightweight Directory Access Protocol). Le paramètre est le répertoire d'installation de LDAP et par défaut `/usr/local/ldap'.

Plus d'informations à propos de LDAP sont disponibles dans les RFC1777 et RFC1778.

4.2.7 Autres options de configuration

4.2.7.1 --with-mcrypt=*DIR*

--with-mcrypt Ajoute le support de la librairie mcrypt. Consultez la documentation concernant 10.37 de cryptage pour plus d'informations. Si vous utilisez le paramètre optionnel, *DIR*, PHP cherchera le fichier mcrypt.h dans le répertoire *DIR*/include.

4.2.7.2 --enable-sysvsem

--enable-sysvsem Ajoute le support des sémaphores Sys V (supportés par la plupart des versions d'UNIX). Consultez la documentation concernant 10.56 Sémaphores et gestion de la mémoire partagée pour plus d'informations.

4.2.7.3 --enable-sysvshm

--enable-sysvshm Ajoute le support des sémaphores Sys V (supportés par la plupart des versions d'UNIX) Consulter la documentation concernant 10.56 Sémaphores et gestion de la mémoire partagée pour plus d'information.

4.2.7.4 --with-xml

--with-xml Ajoute le support du XML en utilisant la librairie expatde James Clark. Voir aussi les références aux 10.67 Analyseur syntaxique XML pour plus de renseignements.

4.2.7.5 --enable-maintainer-mode

--enable-maintainer-mode Ajoute des dépendences supplémentaires ainsi que des options de compilation utilisées par certains développeurs du language PHP.

4.2.7.6 --with-system-regex

--with-system-regex Cette option permet d'utiliser les expressions régulières du système en lieu et place de celles fournies avec le language. Si vous compilez PHP comme module serveur, vous devez utilisez la même librairie lorsque vous compilez PHP et lorsque vous le linkez au serveur. Vous pouvez ajouter cette option si la librairie système ajoute des fonctionnalités supplémentaires dont vous avez besoin. Dans le cas contraire, il est recommandé d'utiliser la librairie fournie avec le language.

4.2.7.7 --with-config-file-path

--with-config-file-path=DIR Le chemin utilisé pour rechercher le 7.1 Le fichier de configuration lorsque PHP est lancé.

4.2.7.8 --with-exec-dir

--with-exec-dir=*DIR* Permet l'exécution des programmes qui se trouvent dans le répertoire DIR lorsque l'option "safe mode" est activée. Par défaut, DIR =`/usr/local/bin'. Cette option ne permet pas de modifier la valeur par défaut. Cette valeur peut être modifiée avec la directive 7.1.3.2 ini.safe-mode-exec-dir dans le 7.1 Le fichier de configuration.

4.2.7.9 --enable-debug

--enable-debug Ajoute la possibilité d'obtenir des informations complémentaires. Il est alors possible d'obtenir plus d'informations, notamment lorsqu'il y a des problèmes avec PHP. (Il est à noter que cela n'a rien à voir avec des facilités de débuggage ou des informations à propos des scripts PHP.)

4.2.7.10 --enable-safe-mode

--enable-safe-mode Active le "safe mode" par défaut. Cela impose de nombreuses restrictions sur les fonctionnalités du PHP, concernant notamment l'ouverture des fichiers. Consultez le chapitre sur la 6 Sécurité pour avoir plus de renseignements. Si vous compilez PHP comme CGI, vous devriez toujours activer le "safe mode". Cela active l'option par défaut. Ce mode peut être activé ou désactivé en utilisant la directive 7.1.3.1 ini.safe-mode dans le 7.1 Le fichier de configuration.

4.2.7.11 --enable-track-vars

--enable-track-vars Permet au PHP de stocker dans les tableaux HTTP_GET_VARS, HTTP_POST_VARS et HTTP_COOKIE_VARS les informations recues par les méthodes GET/POST ou bien en provenance d'un cookie. Cela active l'option par défaut. Cette option peut être activée ou désactivée grâce à la directive 7.1.1.25 ini.track-vars dans le 7.1 Le fichier de configuration.

4.2.7.12 --enable-magic-quotes

--enable-magic-quotes Active l'option "magic quotes" par défaut. Cette option peut être activée ou désactivée grâce à la directive 7.1.1.17 ini.magic-quotes-runtime dans le 7.1 Le fichier de configuration. Voir aussi les directives 7.1.1.16 ini.magic-quotes-gpc et 7.1.1.18 ini.magic-quotes-sybase.

4.2.7.13 --enable-debugger

--enable-debugger Permet d'utiliser le débuggeur interne du PHP. Cette fonctionnalité est encore au stade expérimental. Voir aussi la directive 7.1.4 Directives de configuration de débbugage. dans le 7.1 Le fichier de configuration.

4.2.7.14 --enable-discard-path

--enable-discard-path Si cette option est activée, l'exécutable PHP (dans le cas d'une compilation comme CGI) peut être placée en toute sûreté à l'extérieur de l'arborescence du serveur web. Les utilisateurs ne pourront pas aller au delà des règles de sécurité imposée par le fichier ".htaccess". Voir aussi le chapitre concernant la 6.1.5 Cas 4: L'exécutable PHP à l'extérieur de l'arborescence du serveur à propos de cette option.

4.2.7.15 --enable-bcmath

--enable-bcmath Ajoute les fonctions mathématiques de précision arbitraire bc. Voir aussi l'option 7.1.12 Directives de configuration pour les calculs mathématiques. dans le 7.1 Le fichier de configuration.

4.2.7.16 --enable-force-cgi-redirect

--enable-force-cgi-redirect Active une option de sécurité concernant la redirection interne du serveur web. Si vous utilisez Apache comme CGI, vous devriez utiliser cette option.
Lorsque vous utilisez PHP comme CGI, PHP vérifie toujours par défaut qu'il est utilisé par redirection. (par exemple, sous Apache, en utilisant les directives "Action Directives"). Cela assure que l'éxécutable PHP ne peut pas être utilisé pour passer par dessus les procédures d'authentification du serveur web en appelant une procédure directement. Par exemple, `http://my.host/cgi-bin/php/secret/doc.html'. Dans cette exemple, on accède à la page `http://my.host/secret/doc.html' mais cela n'utilise aucune règle de sécurité pour le répertoire `/secret'.
Ne pas activer cette option revient à annuler la vérification et permet éventuellement de passer par dessus les procédures d'authentification et de sécurité du démon httpd. Désactivez cette option uniquement si votre serveur n'est pas capable d'indiquer si une redirection sécurisée a été effectuée et si tous les fichiers sur votre serveur et dans les répertoires utilisateurs peuvent être accessibles de l'extérieur par tout le monde. Consultez le chapitre concernant la 6.1.3 Cas 2: Utilisation de la directive de compilation --enable-force-cgi-redirect à propos de cette option.

4.2.7.17 --disable-short-tags

--disable-short-tags Désactive la version courte <? ?>des balises PHP. Vous devez désactiver la version courte des balises si vous souhaitez utiliser le PHP avec XML. Si vous désactivez la version courte des balises, les seules balises PHP acceptées sont <?php ?>. Cette option établit la valeur par défaut. Elle peut être activée ou désactivée avec la directive 7.1.1.22 ini.short-open-tag dans le 7.1 Le fichier de configuration.

4.2.7.18 --enable-url-includes

--enable-url-includes Ajoute la possibilité d'exécuter du code PHP à partir d'un autre serveur HTTP ou FTP directement grâce à la fonction include(). Voir aussi l'option 7.1.1.13 ini.include-path dans le 7.1 Le fichier de configuration.

4.2.7.19 --disable-syntax-hl

--disable-syntax-hl Annule la surbrillance de la syntaxe.

4.2.7.20 CPPFLAGS et LDFLAGS

Pour que, lors de la compilation et de l'installation, PHP cherche les différents fichiers ou librairies dans des répertoires précis, vous devez modifier les variables d'environnement **CPPFLAGS** et **LDFLAGS** . Si vous utilisez un shell "sensible" (non-sécurisé?), vous pouvez aussi exécuter la commande suivante : LDFLAGS=-L/my/lib/dir CPPFLAGS=-I/my/include/dir ./configure

4.2.8 Compilation

Lorsque vous avez exécuté le script de configuration, vous êtes prêt pour compiler le PHP comme module ou bien comme CGI. La commande make devrait s'occuper de la compilation. Si cela ne fonctionne pas correctement, vous trouverez dans le paragraphe problèmes de nombreuses réponses aux 4.4 Problèmes? de compilation.

4.2.9 Testing

Si vous avez compilé PHP comme CGI, vous pouvez vérifier que votre compilation c'est bien déroulée en tapant la commande make test. C'est toujours une bonne idée pour vérifier que la compilation s'est bien déroulée. Dans ce sens, vous serez capable de détecter une erreur à la source au lieu de chercher des erreurs le problèmes plus tard.

4.2.10 Etalonnage

Si vous avez compilé PHP comme CGI, vous pouvez utiliser le script de benchmark en tapant la commande make bench. Il est à noter que si le "safe mode" est activé, le bench ne pourra pas se finir si l'exécution prend plus que les 30 secondes permises. Cela tient au fait que la fonction set_time_limit() ne peut pas être utilisé lorsque le "safe mode" est activé. Utilisez la directive 7.1.1.19 ini.max-execution-time pour contrôler le temps d'exécution dans vos scripts. make bench ne tient pas compte du 7.1 Le fichier de configuration.

4.3 Installation sous Windows 95/98/NT

Ce guide d'installation vous guidera pour installer et configurer PHP sur vos serveurs Windows 9x/NT. Ce guide a été préparé par Bob Silva, et traduit par Damien Seguy. La version la plus récente est disponible (en anglais) à http://www.umesd.k12.or.us/php/win32install.html.
Ce guide fournit des informations pour les logiciels :

- Apache 1.3.x
- Omni HTTPd 2.0b1

4.3.1 Configuration de PHP

Les instructions suivantes doivent être suivies pour tous les types d'installation, avant de prendre en compte les spécificités de chaque serveur.

- Décompressez la distribution dans le dossier de votre choix. "C:\PHP3\" est une bonne idée.
- Tous les modules sont préfixés par 'php3_'. Il faudra que vous modifiez le fichier `php3.ini' et/ou tous les scripts de chargement d'extension avec la fonction dl(). Vous pouvez toujours supprimer le préfixe 'php3_', mais il évite les confusions entre les modules PHP et leur bibliothèque de support.

- c:\windows for Windows 95/98
- c:\winnt or c:\winnt40 for NT servers

- Editez votre fichier 'php3.ini' :
  - Vous devez changer 'extension_dir' pour qu'il désigne votre dossier d'installation de PHP, c'est à dire l'endroit oú vous avez installé les fichiers 'php3_*.dll'. Par exemple : c:\php3
  - Si vous utilisez Omni Httpd, passez à l'étape suivante. Modifiez 'doc_root' pour qu'il désigne la racine de votre serveur web. Par exemple : c:\apache\htdocs ou c:\webroot
  - Choisissez les modules que vous voulez lancer au démarrage de PHP. Vous pouvez décommenter la ligne : 'extension=php3_*.dll' pour charger automatiquement ces modules. Certains modules nécessitent d'autres librairies dans votre système pour fonctionner correctement. La FAQ; a plus d'informations sur ces librairies supplémentaires. Vous pouvez toujours charger dynamiquement un module avec: dl("php_*.dll")("php_*.dll");
  - Avec 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' sur les serveur NT. D'autres informations sur les fonctionnalités de browscap sous PHP sont disponibles (en anglais) ici, sélectionnez le bouton "source" pour le voir en action.

Les DLLs des extensions PHP sont préfixées avec 'php3_'. Cela évite la confusion avec les extensions PHP, et leurs librairies.

4.3.2 Windows 95/98/NT et PWS/IIS 3

La méthode recommandée pour configurer ces serveurs est d'utiliser le fichier INF inclus avec la distribution (php_iis_reg.inf). Vous aurez à éditer ce fichier pour vous assurer que les extensions et les dossiers d'installation de PHP sont bien ceux de votre configuration. Ou alors, il vous faudra le faire manuellement. ATTENTION : Ces instructions impliquent la modification directe des tables de registry de Windows. Une erreur peut laisser votre système dans un état instable. Nous recommandons que vous fassiez une sauvegarde de secours de votre registry d'abord. L'équipe de developpement et le traducteur de cette page ne pourront pas être tenu pour responsable si vous endommagez votre registry.

- Lancez Regedit.
- Allez 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 vos script PHP. Par exemple: .php3
- Double-cliquez sur la valeur de la chaîne et entrez le chemin jusqu'à php.exe dans la valeur du champs. Par exemple c:\php3\php.exe %s %s. La chaîne '%s %s' est TRES TRES importante, PHP ne fonctionnera pas correctement sans!
- Recommencez ces deux dernières instructions avec chacune des extensions que vous souhaitez utiliser pour vos scripts PHP.
- Allez maintenant jusqu'à: HKEY_CLASSES_ROOT
- Dans le menu edit, sélectionnez : New->Key.
- Donnez à la clé le nom de l'extension que vous avez donné à la précédente section. Par exemple: .php3
- Selectionnez la nouvelle clé, et dans la boîte de droite, double cliquez sur "default value" et entrez phpfile.
- Recommencez ces deux dernières instructions avec chacune des extensions de la dernière section.

- Dans le menu edit, sélectionnez New->Key dans HKEY_CLASSES_ROOT et donnez lui le nom de phpfile.
- Sélectionnez la clé phpfile et dans la boîte de droite, double cliquez sur et dans le panneau de droite, double cliquez dans "default value" et entrez Script PHP.
- Clic-droit dans la clé phpfile et dans la boîte de droite, double cliquez sur New->Key,et entrez Shell.
- Clic-droit dans la clé Shell et selectionnez New->Key, appelez la open.
- Clic-droit dans la clé open et selectionnez New->Key, appelez la command.
- Sélectionnez la clé command dans la boîte de droite, double cliquez sur "default value" et entrez le chemin jusqu'à php.exe. Par exemple : c:\php3\php.exe -q %1. (n'omettez pas le %1).
- Exit Regedit.

Les utilisateurs PWS et IIS 3 sont maintenant prêts à travailler. Les utilisateurs IIS 3 peuvent utiliser l' outil de Steven Genusa pour configurer leur script maps.

4.3.3 Windows NT et IIS 4

Pour installer PHP sur un serveur NT avec IIS 4, suivez les instructions suivantes :

- Dans Internet Service Manager (MMC), sélectionnez le site web, ou le dossier racine d'une application.
- Ouvrez la boîte de propriété du dossier (clic droit et sélectionnez l'item de propriétés), puis cliquez dans Home Directory, Virtual Directory, ou Directory tab.
- Cliquez dans le bouton Configuration, puis cliquez dans l'onglet App Mappings.
- Cliquez dans Add, et dans la boîte Executable, tapez: c:\path-to-php-dir\php.exe %s %s. Vous DEVEZ mettre les %s %s à la fin, sinon PHP ne fonctionnera pas.
- Dans la boîte Extension, tapez le nom de l'extension que vous voulez associer à vos script PHP (Vous devrez répetez les étapes 5 et 6 pour chaque extension que vous voulez associer à vos script PHP). (.php3 et .phtml sont les valeurs les plus répandues).
- Mettez en place la sécurité appropriée. (Cela se fait dans le gestionnaier Internet Service Manager), et si votre serveur NT utilise le sytème de fichier NTFS, ajoutez les droits d'exécution à I_USR_, pour le dossier qui contient php.exe.

4.3.4 Windows 9x/NT et Apache 1.3.x

Vous devez éditer votre fichier srm.conf ou httpd.conf pour configurer Apache et PHP.
Bien qu'il y ait quelques variations de configuration PHP sous Apache, elle est suffisamment simple pour être maîtrisée par le néophyte. Reportez vous aux documentations Apache (Apache Docs) pour plus de détails sur les directives de configuration.

- ScriptAlias /php3/ "c:/path-to-php-dir/"
- AddType application/x-httpd-php3 .php3
- AddType application/x-httpd-php3 .phtml
- Action application/x-httpd-php3 "/php3/php.exe"

Pour utiliser le fonction de surimpression du code, créez simplement un script PHP et mettez-y le code: <?php show_source ("original_php_script.php3"); ?>. Remplacez original_php_script.php3 par le nom du fichier dont vous souhaitez afficher le code source. (C'est le seul moyen de le faire). *Note:* sous Win-Apache tous les back slashes d'un chemin tel que: "c:\directory\file.ext", doivent être convertis en slash.

4.3.5 Omni HTTPd 2.0b1 pour Windows

Ceci est la configuration la plus simple :

- 1: Installez Omni server
- 2: Clic droit sur l'icône bleue OmniHTTPd et sélectionnez Propriétés
- 3: Cliquez sur Web Server Global Settings
- 4: Dans l'onglet 'Externe', entrez :virtual = .php3 | actual = c:\path-to-php-dir\php.exe
- 5: Dans l'onglet Mime, entrez: virtual = wwwserver/stdcgi | actual = .php3 6: Cliquez OK

Répetez les étapes 2 - 6 pour chaque nouvelle extension à associer avec PHP.

4.3.6 PHP Modules

php3_calendar.dll	Fonction de conversions calendaires
php3_crypt.dll	Fonction de cryptage
php3_dbase.dll	Fonctions DBase
php3_dbm.dll	Emulation GDBM avec la librairie Berkely DB2
php3_filepro.dll	Accès aux bases de données File pro (accès en lecture seule).
php3_gd.dll	Manipulation des images GIF/JPG/PNG avec GD Library
php3_hyperwave.dll	Fonctions HyperWave
php3_imap4r2.dll	Fonctions IMAP 4
php3_ldap.dll	Fonctions LDAP
php3_msql1.dll	Fonctions client mSQL 1
php3_msql2.dll	Fonctions client mSQL 2 client
php3_mssql.dll	Fonctions client MSSQL client (requiert MSSQL DB-Libraries)
php3_mysql.dll	Fonctions MySQL
php3_nsmail.dll	Fonctions Netscape mail
php3_oci73.dll	Fonctions Oracle
php3_snmp.dll	Fonctions SNMP get et walk (NT seulement!)
php3_zlib.dll	Fonctions ZLib

4.4 Problèmes?

4.4.1 Consultez la FAQ

Certains problèmes sont plus courants que d'autres. La solution aux problèmes les plus courants sont rassemblés dans la FAQ PHP, disponibles à l'adresse http://www.php.net/FAQ.php.

4.4.2 Rapporter un bug

Si vous pensez que vous avez trouvé un bug dans PHP, veuillez le faire savoir. Les développeurs PHP ne le connaissent probablement pas, et si vous ne le faites pas connaître, il n'y a aucune chance que celui-ci soit corrigé. Vous pouvez le faire savoir en utilisant le bug-tracking système à l'adresse http://www.php.net/bugs.php.

4.4.3 Autres problèmes

Si vous êtes toujours dans l'impasse, il y a probablement quelqu'un sur la liste de diffusion PHP qui pourra vous aider. Vous devriez déjà vérifier dans les archives de la liste de diffusion au cas oú quelqu'un aurait déjà répondu à votre question. Les archives sont accessibles à partir de la page "support" à l'adresse http://www.php.net/. Pour s'inscrire sur la liste de diffusion PHP, envoyer un message vide à l'adresse suivante: L'adresse de la liste de diffusion est: php-general@lists.php.net.
Si vous voulez obtenir de l'aide sur la liste de diffusion, veuillez essayer de préciser votre environnement (quel OS, quelle version de PHP, quel serveur web, si vous utilisez PHP comme CGI ou commer module serveur, etc...), et donnez assez de code afin que les membres de la liste puissent reproduire votre problème et le tester.

5 Introduction

5.1 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 :

```
<html>
   <head>
      <title>Exemple</title>
   </head>
   <body>
      <?php echo "Salut, 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 9.1.1 Le passage du HTML au PHP 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.

5.2 Que peut faire PHP pour vous?

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
- dBase
- Empress
- FilePro
- Informix
- InterBase
- mSQL
- MySQL
- Oracle
- PostgreSQL
- Solid
- Sybase
- Velocis

- Unix dbm

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.

5.3 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 marco spéciales et d'un petit nombre d'utilitaires couramment utilisés dans les pages web. Un guestbook, 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'analyseyr parseur fut de nouveau réécrit par Zeev Suraskyi 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.
Aujourd'hui (été 1999) PHP/FI ou PHP3 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, et leurs statistiques Netcraft Web Server Survey) que le PHP est utilisé sur 150 000 sites web dans le monde entier. Pour comparaison, ce chiffre est supérieur au nombre de serveur tournant sous Netscape Enterprise server.

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 pour améliorer les performances et améliorera le support des seveurs web autre que Apache.

6 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 fragilise 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 permettront 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 fragilise la sécurité du serveur. Ce chapitre explique comme les différentes options de configurations peuvent être combinées, tout en conservant une sécurité maximum.

6.1 CGI binary

6.1.1 Faiblesses connues

Utiliser le PHP comme un **CGI** exécutable vient la plupart 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 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 pour obtenir un document `http://my.host/secret/script.php3' par l'interpréteur PHP. Dans une telle configuration, le serveur web vérifie d'abord si il a accès au répertoire `/secret', et après cette vérification redirige la requête vers `http://my.host/cgi-bin/php/secret/script.php3'. 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.php3', 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 au document protégés sur le serveur web.
  Avec le PHP, l'option de compilation 4.2.7.16 --enable-force-cgi-redirect et les options d'exécution 7.1.1.6 ini.doc-root et 7.1.1.27 ini.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.

6.1.2 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 4.2.7.16 --enable-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.php3' ou bien de manière indirecte, par redirection,`http://my.host/dir/script.php3'.
Les redirections peuvent être configurées dans les fichiers de configuration d'Apache en utilisant les directives "AddHandler" et "Action" (voir ci-dessous).

6.1.3 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.php3'. Dans ce cas là, PHP parsera le fichier uniquement si il y a eu redirection.
Habituellement, le serveur web Apache réalise une redirection grâce aux directives suivantes :
Action php3-script /cgi-bin/php
AddHandler php3-script .php3
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 oú 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 manières pour utiliser la version binaire CGI du PHP, comme exposé ci-dessous.

6.1.4 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 plupart des administrateurs préfèrent mettre en place un répertoire spécial pour les scripts qui est uniquement accessible par le biais du binaire CGI du PHP, et donc, tous les fichiers de ce répertoire seront interprétés et non affichés tels quel.
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 7.1.1.6 ini.doc-root dans le 7.1 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 prédéfinit. (à l'exception du répertoire désigné par la directive **user_dir** Voir ci-dessous).

Une autre option possible ici est la directive 7.1.1.27 ini.user-dir. Lorsque la directive n'est pas activée, seulement les fichiers contenues dans le répertoire **doc_root** peuvent être ouverts. Ouvrir un fichier possédant l'url `http://my.host/~user/doc.php3' ne correspond pas à l'ouverture d'un fichier sous le répertoire racine de l'utilisateur mais à l'ouverture du fichier `~user/doc.php3' sous le repertoire "doc_root" (oui, un répertoire comment 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.php3' ouvrira un fichier appelé `doc.php3' 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.php3'.
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".

6.1.5 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. itself.
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 4.2.7.14 --enable-discard-path.

6.2 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 "noboby").

7 Configuration

7.1 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().

7.1.1 Directives de configuration générale

7.1.1.1 ini.asp-tags

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 9.1.1 Le passage du HTML au PHP.
  Note : *Le support des balises ASP a été ajouté dans la version 3.0.4.*

7.1.1.2 ini.auto-append-file

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 7.1.1.13 ini.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.*

7.1.1.3 ini.auto-prepend-file

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 7.1.1.13 ini.include-path est utilisé.
  Le mot réservé NONE désactive l'auto-appending.

7.1.1.4 ini.cgi-ext

cgi_ext chaîne de caractères

- (NDT : aucune documentation n'est fournie).

7.1.1.5 ini.display-errors

display_errors booléen

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

7.1.1.6 ini.doc-root

doc_root string

- Le dossier racine de PHP. Utilisée uniquement si elle est définie. Si PHP est configuré en 7.1.3.1 ini.safe-mode, aucun fichier en dehors de ce dossier ne sera accessible.

7.1.1.7 ini.engine

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.

7.1.1.8 ini.error-log

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.

7.1.1.9 ini.error-reporting

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 :

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).

7.1.1.10 ini.open-basedir

- **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.

7.1.1.11 ini.gpc-order

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.

7.1.1.12 ini.ignore-user-abort

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().

7.1.1.13 ini.include-path

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.
-
- include_path=.:/home/httpd/php-lib
-
- include_path=".;c:\www\phplib"
-

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

7.1.1.14 ini.isapi-ext

isapi_ext string

- (Aucune documentation n'est fournie)

7.1.1.15 ini.log-errors

log_errors boolean

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

7.1.1.16 ini.magic-quotes-gpc

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), \ (backslash) et NUL sont échappés avec un backslash. Si magic_quotes_sybase fonctionne aussi, les guillemets simples seront échappés avec un autre guillemet simple, plutôt qu'un backslash.

7.1.1.17 ini.magic-quotes-runtime

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 echappés avec un backslash. Si **magic_quotes_sybase** est aussi activé, les guillemets simples seront échappés avec un autre guillemet simple, plutôt qu'un backslash.

7.1.1.18 ini.magic-quotes-sybase

magic_quotes_sybase boolean

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

7.1.1.19 ini.max-execution-time

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.

7.1.1.20 ini.memory-limit

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.

7.1.1.21 ini.nsapi-ext

nsapi_ext chaîne de caractères

- Aucune documentation n'est fournie.

7.1.1.22 ini.short-open-tag

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 ?>).

7.1.1.23 ini.sql.safe-mode

sql.safe_mode booléen

- Aucune documentation n'est fournie.

7.1.1.24 ini.track-errors

track_errors booléen

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

7.1.1.25 ini.track-vars

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.

7.1.1.26 ini.upload-tmp-dir

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.

7.1.1.27 ini.user-dir

user_dir chaîne de caractères

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

7.1.1.28 ini.warn-plus-overloading

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.

7.1.2 Configuration des directives concernant le mail

7.1.2.1 ini.smtp

SMTP chaîne de caractères

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

7.1.2.2 ini.sendmail-from

sendmail_from chaîne de caractères

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

7.1.2.3 ini.sendmail-path

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. Dans ce cas la, vous devez mettre: `/var/qmail/bin/sendmail'.

7.1.3 Directives de configuration du "Safe Mode"

7.1.3.1 ini.safe-mode

safe_mode booléen

- Cette directive active ou désactive l'option "safe mode". Lisez le chapitre 6 Sécurité pour plus d'informations.

7.1.3.2 ini.safe-mode-exec-dir

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.

7.1.4 Directives de configuration de débbugage.

7.1.4.1 ini.debugger.host

debugger.host chaîne de caractères

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

7.1.4.2 ini.debugger.port

debugger.port chaîne de caractères

- Numéro du port utilisé pour le déboggage.

7.1.4.3 ini.debugger.enabled

debugger.enabled booléen

- Activation ou désactivation du debugger.

7.1.5 Directives de chargement des extensions

7.1.5.1 ini.enable-dl

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().

7.1.5.2 ini.extension-dir

extension_dir chaîne de caractères

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

7.1.5.3 ini.extension

extension chaîne de caractères

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

7.1.6 MySQL Configuration Directives

7.1.6.1 ini.mysql.allow-persistent

mysql.allow_persistent booléen

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

7.1.6.2 ini.mysql.default-host

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.

7.1.6.3 ini.mysql.default-user

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.

7.1.6.4 ini.mysql.default-password

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.

7.1.6.5 ini.mysql.max-persistent

mysql.max_persistent entier

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

7.1.6.6 ini.mysql.max-links

mysql.max_links entier

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

7.1.7 Directives de configuration mSQL

7.1.7.1 ini.msql.allow-persistent

msql.allow_persistent booléen

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

7.1.7.2 ini.msql.max-persistent

msql.max_persistent entier

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

7.1.7.3 ini.msql.max-links

msql.max_links entier

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

7.1.8 Directives de configuration Postgres

7.1.8.1 ini.pgsql.allow-persistent

pgsql.allow_persistent booléen

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

7.1.8.2 ini.pgsql.max-persistent

pgsql.max_persistent entier

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

7.1.8.3 ini.pgsql.max-links

pgsql.max_links entier

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

7.1.9 Directives de configuration Sybase

7.1.9.1 ini.sybase.allow-persistent

sybase.allow_persistent booléen

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

7.1.9.2 ini.sybase.max-persistent

sybase.max_persistent entier

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

7.1.9.3 ini.sybase.max-links

sybase.max_links entier

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

7.1.10 Sybase-CT Configuration Directives

7.1.10.1 ini.sybct.allow-persistent

sybct.allow_persistent booléen

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

7.1.10.2 ini.sybct.max-persistent

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.

7.1.10.3 ini.sybct.max-links

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.

7.1.10.4 ini.sybct.min-server-severity

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.

7.1.10.5 ini.sybct.min-client-severity

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.

7.1.10.6 ini.sybct.login-timeout

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'expire, le script sera terminé avant le message d'erreur. Par défaut, cette valeur vaut 1 minute.

7.1.10.7 ini.sybct.timeout

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'expire, votre script sera terminé avant le message d'erreur. Par défaut, il n'y a pas de limite.

7.1.10.8 ini.sybct.hostname

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.

7.1.11 Directives de configuration Informix

7.1.11.1 ini.ifx.allow-persistent

ifx.allow_persistent booléen

- Active les connexions persistantes à une base de données Informix.

7.1.11.2 ini.ifx.max-persistent

ifx.max_persistent entier

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

7.1.11.3 ini.ifx.max-links

ifx.max_links entier

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

7.1.11.4 ini.ifx.default-host

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().

7.1.11.5 ini.ifx.default-user

ifx.default_user chaîne de caractères

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

7.1.11.6 ini.ifx.default-password

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().

7.1.11.7 ini.ifx.blobinfile

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().

7.1.11.8 ini.ifx.textasvarchar

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().

7.1.11.9 ini.ifx.byteasvarchar

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().

7.1.11.10 ini.ifx.charasvarchar

ifx.charasvarchar booléen

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

7.1.11.11 ini.ifx.nullformat

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().

7.1.12 Directives de configuration pour les calculs mathématiques.

7.1.12.1 ini.bcmath.scale

bcmath.scale entier

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

7.1.13 Directives de configuration du navigateur.

7.1.13.1 ini.browscap

browscap chaîne de caractères

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

7.1.14 Directives de configuration du driver ODBC unifié

7.1.14.1 ini.uodbc.default-db

uodbc.default_db chaîne de caractères

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

7.1.14.2 ini.uodbc.default-user

uodbc.default_user chaîne de caractères

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

7.1.14.3 ini.uodbc.default-pw

uodbc.default_pw chaîne de caractères

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

7.1.14.4 ini.uodbc.allow-persistent

uodbc.allow_persistent booléen

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

7.1.14.5 ini.uodbc.max-persistent

uodbc.max_persistent entier

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

7.1.14.6 ini.uodbc.max-links

uodbc.max_links entier

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

8 Caractéristiques

8.1 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 .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 vrai si la connexion a été annulée.
Votre script peut aussi expirer 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 expire, 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'expiration qui a causé la fin du script, en appelant la fonction connection_timeout(). Cette fonction retournera vrai si le délai d'expiration 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 expirer. A ce moment là, les deux fonctions connection_timeout() et connection_aborted() vont retourner vrai. 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.

8.2 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().

8.3 Gestion des erreurs

Il y a 4 types d'erreurs et d'alerte PHP :

- 1 - Erreur de fonction
- 2 - Alerte normale
- 4 - Erreur d'analyse

- 8 - Notes (alertes qui peuvent être ignorées mais qui peuvent avoir des conséquences sur le code)

Les 4 numéros attachés à un type d'erreur ont été ajoutés pour définir un niveau d'erreur. Par défaut, le niveau d'erreur accepté est de 7, c'est à dire 1 + 2 + 4, ou n'importe quoi, sauf les notes. Ce niveau peut être changé, dans le fichier php.ini avec la directive error_reporting. Il peut aussi être modifié dans le fichier de configuration Apache httpd.conf avec la directive php3_error_reporting ou bien encore, lors de l'exécution, en définissant la fonction error_reporting().
Toutes les 9.4 Les expressions peuvent être appelées avec le préfixe "@", ce qui permet d'ignorer le rapport d'erreur pour cette fonction particulière. Si une erreur survient dans une telle expression, et que l'option 7.1.1.24 ini.track-errors est activée, vous pouvez retrouver le message d'erreur dans la variable globale php_errormsg.

8.4 Gestion des chargements de fichier

8.4.1 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 texte binaire. 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 8.4.4 Chargement par méthode PUT.
Un écran de chargement de fichiers peut être constitué en créant un formulaire de la manière suivante :

```
<FORM ENCTYPE="multipart/form-data" ACTION="_URL_" METHOD=POST>
<INPUT TYPE="hidden" name="MAX_FILE_SIZE" value="1000">
Send this file: <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 :

- $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. system.
- $userfile_size - La taille envoyée 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".
Les fichiers seront enregistrés par défaut dans le dossier temporaire du système. Ceci peut être changé en modifiant la variable d'environnement **TMPDIR** de l'environnement dans lequel PHP fonctionne. Vous pouvez modifier cette valeur depuis PHP même, avec la fonction putenv().
Le script PHP qui recoit 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 petit. Vous pouvez utiliser la variable $file_type pour recaler les fichiers qui n'ont pas le bon type. Quelque soit 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, si il n'a pas été déplacé ou renommé.

8.4.2 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 Megaoctets.
Attention : il semble que CERN httpd supprime tout ce qui est après le premier caractère dans l'entête MIME. Tant que c'est le cas, CERN httpd ne pourra pas effectuer de chargement.

8.4.3 Chargement multiples de fichiers

Il est possible de charger plusieurs fichiers en même temps, et 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 boîte à cocher multiples.
Note : *Le support du chargement multiple de fichier a été ajouté dans la version 3.0.10.*

```
<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 est 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, si on veut envoyer les fichiers `/home/test/review.html' et `/home/test/xwp.out', $userfile_name[0] contiendra review.html, et $userfile_name[1] contiendra xwp.out. De même, $userfile_size[0] contiendra la taille de filesize, etc

8.4.4 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 oú dans le fichier de configuration d'Apache. En général, les webmestres la place dans le block <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 :

```
<? 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 recoit un chargement par méthode PUT, il va enregistrer le fichier dans le dossier temporaire, tout comme avec la 8.4.1 Chargements de fichiers par méthode POST. 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.

8.5 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 :

```
<?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'expiration, ou alors, fournissent un bouton de déconnexion.

```
<?php
function authenticate() {
Header( "WWW-authenticate: basic realm='Test Authentication System'");
Header( "HTTP/1.0 401 Unauthorized");
echo "You must enter a valid login ID and password to access this resource\n";
exit;
  }
if(!isset($PHP_AUTH_USER) || ($SeenBefore == 1 && !strcmp($OldAuth, $PHP_AUTH_USER)) ) {
authenticate();
  }
else {
echo "Welcome: $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.
Notez aussi que tout ceci ne fonctionne pas sous Microsoft's IIS et que les limitations de PHP en version CGI sont dues aux limitations de IIS.

8.6 Création d'images

PHP n'est pas limité à la création de fichier HTML. Il peut aussi servir à créer des images JPG/PNG à la volée, aussi bien pour les émettre que pour les sauver. Il faut alors compiler PHP avec la librairie GD.

```
<?php
Header("Content-type: image/gif");
```

```
   $string=implode($argv," ");
   $im = imagecreatefromgif("images/button1.gif");
   $orange = ImageColorAllocate($im, 220, 210, 60);
   $px = (imagesx($im)-7.5*strlen($string))/2;
ImageString($im,3,$px,9,$string,$orange);
ImageGif($im);
ImageDestroy($im);
?>
```

Cet exemple sera appelé depuis une page HTML avec un tag tel 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.

8.7 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 de 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 habitués 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, cela ne permet par l'ouverture de plusieurs sessions avec le même lien, cela ne permet pas la réalisation de transactions efficaces, et cela ne fait pas le café. En fait, pour être extrêmement clair sur le sujet, les connexions persistantes ne vous donnent aucune fonctionnalité de plus qu'avec 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. Pour un serveur multi-thread, actuellement, c'est purement théorique, car PHP ne fonctionne par encore sous cette forme. Le développement est en cours pour supporter ISAPI, WSAPI, et NSAPI (sous Windows), qui permettront d'utiliser PHP comme un module pour des serveurs tels que Netscape FastTrack, Microsoft's Internet Information Server (IIS), et O'Reilly's WebSite Pro. Lorsque cela sera fait, le comportement sera le même que pour les serveur multi-process.
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.
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.

8.8 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.

```
<?;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.)

```
<?;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().*

9 Langage

9.1 La syntaxe de base

9.1.1 Le passage du HTML au PHP

Il y a quatre moyens pour passer du mode HTML au mode PHP :

1. <? echo ("Ceci est un exemple d'affichage à l'écran en PHP.\n"); ?>
2. <?php echo("Si vous voulez afficher du texte, faites comme ceci.\n"); ?>
3. <script language="php">
echo ("Certain éditeur HTML n'accepte pas les délimiteurs ci-dessus.");
   </script>
4. <% echo ("Vous pouvez aussi utiliser le style ASP comme délimiteur."); %>
   <%= $variable; # ceci est un raccourci pour "<%%echo .." %>

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 7.1.1.22 ini.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 7.1.1.1 ini.asp-tags, soit en utilisant l'option de compilation --enable-asp-tags compile-time. Note : *Le support de la quatrième possibilité, ASP-style, a été ajouté dans la version 3.0.4.*

La parenthèse fermante pour un bloc ajoutera automatiquement un retour à la ligne si il y en a un de présent.

9.1.2 Le séparateur d'instruction

Les instructions sont séparées comme en C ou en Perl par un point virgule à chaque fin d'instruction.
Le tag 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" ?>
```

9.1.3 Comments

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 <?# echo "simple";?> exemple.</h1>
<p>La ligne du dessus affichera 'Ceci est un 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 */
 */
?>
```

9.2 Les constantes

Le PHP définit un certain nombre de constantes et propose des mécanismes pour en définir d'autres durant l'exécution. Les constantes se comporent des variables, à l'exception du fait que leur valeur est définie grâce à la fonction define(), et qu'elle ne peut pas être modifiée par la suite.
Les constantes prédéfinies (toujours disponibles) sont :

__FILE__

- Le nom du fichier qui est actuellement exécuté. Si cette constante est utilisée dans le cadre d'un fichier "inclus" (ou require), alors le nom du fichier inclus est renvoyé, et non le nom du fichier parent.

__LINE__

- Le numéro de la ligne qui est actuellement exécutée. Si cette constante est utilisée dans le cadre d'un fichier "inclus" (ou 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. Par exemple, 'Linux'.
  TRUE

- La valeur TRUE.
  FALSE

- La valeur FALSE.
  E_ERROR

- Dénote une erreur autre qu'une "parsing error" (erreur d'analyse) 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.

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.

```
<?php
define("CONSTANT", "Bonjour le monde.");
echo CONSTANT; // affiche "Bonjourle monde."
?>
```

```
<?php
function report_error($file, $line, $message) {
echo "Une erreur a eu lieu dans le fichier $file à la ligne $line: $message.";
}
report_error(__FILE__,__LINE__, "Y a un problème!");
?>
```

9.3 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.

9.3.1 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 :
if (expr)
statement

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*:

```
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 conditonnelle 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*:

```
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.

9.3.2 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 avec 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:

```
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.

9.3.3 elseif

elseif, comme son nom l'indique, est une combinaison de if et 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:

```
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.

9.3.4 Syntaxe alternative

Le PHP propose une autre manière de rassembler des instructions à l'intérieur d'un bloc, pour les fonctions de contrôle if, while, for, 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 is equal to 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:

```
if ($a == 5):
print "a equals 5";
print "...";
elseif ($a == 6):
print "a equals 6";
print "!!!";
else:
print "a is neither 5 nor 6";
endif;
```

Allez voir 9.3.5 while, 9.3.7 for, et 9.3.1 if pour d'autres exemples.

9.3.5 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 :
while (expr) statement

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:
while (expr): statement ... endwhile;

Les exemples suivants sont identiques, et affichent tous les nombres de 1 à 10:

```
/* 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;
```

9.3.6 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'un syntaxe possible pour les boucles do..while:

```
$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 9.3.9 break. Le code suivant montre une utilisation possible:

```
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 comprennez pas tout correctement. Vous pouvez écrire des scripts très très puissants sans utiliser cette fonctionnalité.

9.3.7 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:
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 9.3.9 break.
Considérons les exemples suivants. Tous affichent les chiffres de 1 à 10:

```
/* 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 :
for (expr1; expr2; expr3): statement; ...; endfor;

Les autres langages ont l'instruction foreach pour accéder aux éléments d'un tableau. PHP3 ne dispose pas d'une telle fonction; PHP4 en dispose (voir 9.3.8 foreach). En PHP3, vous pouvez combiner 9.3.5 while avec list() et each() pour obtenir le même résultat. Reportez vous aux exemples de la documentation.

9.3.8 foreach

PHP4 (mais pas PHP3) inclus 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:
foreach(array_expression as $value) statement
foreach(array_expression as $key => $value) statement

La première forme passe en revue le tableau array_expression. A chaque itération, la valeur de l'élément courant est assigné à $value et le pointeur interne de tableau est avancé d'un élément (ce qui fait qu'à la prochaîne 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 démare, 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.
Vous pouvez remarquer que les exemples suivants fonctionnent de manière identique :

```
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 :

```
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 :

```
/* exemple 1: valeur seule */
$a = array (1, 2, 3, 17);
foreach ($a as $v) {
print "Valeur courante de \$a: $v.\n";
}
/* exemple 1: valeur (avec clé associée) */
$a = array (1, 2, 3, 17);
$i = 0; /* pour affichage seulement*/
foreach($a as $v) {
print "\$a[$i] => $k.\n";
}
/* exemple 1: valeur et clé */
$a = array (
    "one" => 1,
    "two" => 2,
    "three" => 3,
    "seventeen" => 17
);
foreach($a as $k => $v) {
print "\$a[$k] => $v.\n";
}
```

9.3.9 break

L'instruction break permet de sortir d'une structure if, for, while, ou switch.
break accepte un argument numérique optionnel qui vous indiquera combien de structures emboîtées ont été interrompues.

```
$i = 0;
while ($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;
   }
}
```

9.3.10 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 emboîtées ont été ignorées.

```
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";
}
```

9.3.11 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:

```
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 :

```
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, 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 si il est possible que vous l'omettiez dans certaines circonstances).
Dans une commande switch, une condition n'est évaluée qu'une fois, est le résultat est comparé à chaque case. Dans une structure elseif, les conditions sont évaluées à chaque comparaisons. Si votre condition est plus compliquée qu'une simple comparaison, ou bien fait partie d'une boucle, switch sera plus rapide.
La liste de commande d'un case peut être vide, auquel cas PHP utilisera la liste de commande du cas suivant.

```
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. Par exemple :

```
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 caractère. Les tableaux sont sans interêt dans ce contexte-là.
La syntaxe alternative pour cette structure de contrôle est la suivante : (Pour plus d'informations, voir 9.3.4 Syntaxe alternative).

```
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;
```

9.3.12 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 9.1.1 Le passage du HTML au PHP.
require() n'est pas vraiment une fonction PHP : c'est plus un instruction du langage. Elle ne fonctionne pas comme les fonctions standards. Par exemple, require() ne peut pas contenir d'autres structures de contrôle. De plus, il 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().

```
require ('header.inc');
```

Attention : include() et require() ajoute le contenu du fichier cible dans le script lui-même. Elle n'utilise 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.

```
require ("file.inc?varone=1&vartwo=2"); /* Ne fonctionne pas. */
$varone = 1;
$vartwo = 2;
require ("file.inc");  /* $varone et $vartwo seront accessible 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 8.8 Utilisation des fichiers à distance ce qui est au dessus reste vrai.
En PHP3, 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 PHP4, cette possibilité a été supprimée. Si vous en avez besoin, utilisez plutôt include().

9.3.13 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 9.1.1 Le passage du HTML au PHP.
Cela a lieu à chaque fois que la fonction include() est rencontrée. Donc vous vous pouvez utiliser la

fonction include() dans une boucle pour inclure un nombre infini de fois un fichier, ou même des fichiers différents.

```
$files = array ('first.inc', 'second.inc', 'third.inc');
for ($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 si il est placé dans une structure conditionnelle, comme dans un 9.3.1 if).

Parce que la fonction include() nécessite une construction particulière, vous devez l'inclure dans un bloc si elle est inclue dans une structure conditionnelle.

```
/*  Ceci est faux, et ne fonctionnera pas ce qu'on attend. */
if ($condition)
include($file);
else
include($other);
/* Ceci est CORRECT. */
if ($condition) {
include($file);
} else {
include($other);
}
```

En PHP3, 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 PHP4, cette possibilité a été supprimée. Cependant, PHP4 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 PHP3.

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 PHP3, 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 PHP3. En PHP4, 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 PHP4, l'affichage sera :

```
Avant le retour
Retour dans le main.html
```

Au contraire, PHP3 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'. Lors que le return est sorti du bloc, l'affichage
devient :

```
Avant le retour
27Retour dans le main.html
```

Le '27' est du au fait que PHP3 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 9.1.1 Le passage du HTML au 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 8.8 Utilisation des fichiers à distance ce qui est au dessus reste vrai.
Voir aussi readfile(), require() et virtual().

9.4 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 :

```
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:

```
$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:

```
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 variable $f ($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 une 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 faites 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 tout expression PHP valide.

9.5 Fonctions

9.5.1 Les fonctions utilisateurs

une fonction peut être définie en utilisant la syntaxe suivante :

```
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 9.6.1 class.
En PHP3, les fonctions doivent être définies avant qu'elles ne soient utilisées. Ce n'est plus le cas en PHP4. PHP ne supporte pas le surchargement de fonction, ni la destruction ou la redéfinition de fonctions déjà déclarées.
PHP3 ne supporte pas un nombre variable d'arguments (voir 9.5.2.2 Valeur par défaut des arguments pour plus d'informations). PHP4 supporte les deux : voir 9.5.2.3 Variable-length argument lists et les fonctions de références que sont func_num_args(), func_get_arg(), et func_get_args() pour plus d'informations.

9.5.2 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 9.5.2.2 Valeur par défaut des arguments (méthode par défaut), par 9.5.2.1 Passage d'arguments par référence. Les listes variable d'arguments sont supportées par PHP4 et plus récent. Voir 9.5.2.3 Variable-length argument lists 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];
}
```

9.5.2.1 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 puisse 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.'
```

9.5.2.2 Valeur par défaut des arguments

Vous pouvez définir comme en C++ des valeurs par défaut pour les arguments de type scalaire :

```
function makecoffee ($type = "cappucino") {
return "Faire une casse de $type.\n";
}
echo makecoffee ();
echo makecoffee ("thé");
```

La fonction ci-dessus affichera :
Faire une tasse de cappucino.
Faire une tasse de thé.

La valeur par défaut d'un argument doit obligatoirement être une constante, et ne peut être ni une variable ou ni un membre de classe.
Il est à noter que vous utilisez les arguments par défaut, la valeur par défaut doit se trouver du côté droit du signe '='; sinon, cela ne fonctionnera pas. Considérons le code suivant :

```
function makeyogurt ($type = "acidophilus", $flavour) {
return "Préparer un bol de $type $flavour.\n";
}
echo makeyogurt ("framboise");   // ne fonctionne pas comme voulu
```

L'affiche du code ci-dessus est le suivant :
Warning: Missing argument 2 in call to makeyogurt() in
/usr/local/etc/httpd/htdocs/php3test/functest.html on line 41
Préparer un bol de framboise.

Maintenant comparons l'exemple précédent avec l'exemple suivant :

```
function makeyogurt ($flavour, $type = "acidophilus") {
return "Préparer un bol de $type $flavour.\n";
}
echo makeyogurt ("framboise");   // fonctionne comme voulu
```

L'affichage de cette exemple est le suivant :
Préparer un bol de acidophilus framboise.

9.5.2.3 Variable-length argument lists

PHP4 supporte le fonctions à nombre d'argument variable. C'es 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 défintion de la fonction, et se comportera comme normalement.

9.5.3 Les valeurs de retour

Les valeurs sont renvoyées en utilisant une instruction de retour optionnelle. Tous types de variables peuvent être renvoyées, tableaux et objets compris.

```
function square ($num) {
return $num * $num;
}
echo square (4);   // affiche '16'.
```

Vous ne pouvez pas renvoyer plusieurs valeurs en même temps, mais vous pouvez obtenir le même résultat en renvoyant un tableau.

```
function small_numbers() {
return array (0, 1, 2);
}
list ($zero, $one, $two) = small_numbers();
```

9.5.4 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 PHP3
Les fonctions déclarées comme old_function ne peuvent pas être appelée 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 appelera la fonction old_function.

9.5.5 Variable functions

PHP supporte le concept de fonctions variables. Cela signifie que si le nom d'une variable est entourée de parenthèses, PHP recherchera une fonction de même nom, et essaiera de l'exécuter. Cela peut servir, entre autre, lors de call back, de tables de fonctions...

```
<?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' );
?>
```

9.6 Classes et objets

9.6.1 class

Une classe est une collection de variables et de fonctions qui concernent ces variables. Une classe est définie en utilisant la syntaxe suivante :

```
<?php
class Cart {
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 Cart 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.
Les classes sont un type de variable. Pour créer une variable du type désiré, vous devez utiliser l'opérateur new.

```
 $cart = new Cart;
 $cart->add_item("10", 1);
```

L'instruction ci-dessus crée l'objet $cart de la class Cart. La fonction add_item() est appelée afin d'ajouter l'article numéro 10 dans la panier.
Une classe peut être une extension d'une autre classe. Les classes "extended" ou "derived" héritent de toutes les variables et de toutes les fonctions de la classe père plus toutes les définitions que vous rajoutez à cette classe. Cela se fait avec le mot clef "extends". L'héritage multiple n'est pas supporté.

```
class Named_Cart extends Cart {
var $owner;
function set_owner ($name) {
       $this->owner = $name;
   }
}
```

L'exemple ci-desssus définit la classe Named_Cart qui possède les même variables que la classe Cart 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.

```
$ncart = new Named_Cart;    // 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)
```

Dans les fonctions d'une classe, la variable $this est égale à l'objet de la classe. Vous pouvez utilisez la forme "$this->quelquechose" pour accéder aux fonctions ou aux variables de l'objet courant.
Le constructeur est la fonction qui est appelée automatiquement par la classe lorsque vous créez une nouvelle instance d'une classe. La fonction constructeur a le même nom que la classe.

```
class Auto_Cart extends Cart {
function Auto_Cart () {
       $this->add_item ("10", 1);
   }
}
```

L'exemple ci-dessus définit la classe Auto_Cart qui hérite de la classe Cart 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.

```
class Constructor_Cart extends Cart {
function Constructor_Cart ($item = "10", $num = 1) {
       $this->add_item ($item, $num);
   }
}
// Place dans le caddie toujours la même chose...
$default_cart   = new Constructor_Cart;
// Place dans le caddie des objets différents, comme dans la réalité
$different_cart = new Constructor_Cart ("20", 17);
```

Pour les classes qui utilisent l'héritage, le constructeur de la classe père n'est pas automatiquement appelé lorsque le constructeur de la classe dérivée est appelé.

9.7 Les opérateurs

9.7.1 Les opérateurs arithmétiques

Vous rappelez vous des opérations élémentaires apprises à l'école ?

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 etd $b.	
$a % $b	Modulo	Reste de $a divisé par $b.	

9.7.2 Les opérateurs d'assignement

L'opérateurs d'assignement 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'assignement est la valeur assignée. Par exemple, la valeur de l'expression '$a = 3' est la valeur 3. Cela permet de faire d'utiliser des astuces telles que : @example $a = ($b = 4) + 5; // $a est maintenant égal à 9, et $b vaut 4.

En plus du simple opérateur d'assignement, 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:
$a = 3;
$a += 5; // affecte la valeur 8 à la variable $a. (correspond à l'instruction '$a = $a + 5');
$b = "Hello ";
$b .= "There!"; // affecte la valeur "Bonjour ici!" à la variable $b (correspond à $b = $b."ici!";

On peut noter que l'assignement copie le contenu de la variable originale dans la nouvelle (assignement 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. PHP4 supporte aussi l'assignement par référence, en utilisant la syntaxe $var = &$othervar;, mais ce n'était pas possible en PHP3. 'L'assignement par référence' signifie que les deux variables contiennent les mêmes données, et que la modification de l'une affecte l'autre. D'un autre coté, la recopie est très rapide.

9.7.3 Bitwise Operators

Bitwise operators allow you to turn specific bits within an integer on or off.

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)	Bits that are set in either $a or $b are set.	
$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).	

9.7.4 Opérateurs de comparaison

Les opérateurs de comparaison, comme le nom l'indique, vous permettent de comparer deux valeurs.

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 (PHP4 seulement).	
$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 conditionnelle est l'opérateur ternaire (":?"), qui fonctionne comme en langage C.
(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.

9.7.5 Opérateur de contrôle d'erreur

PHP supports one error control operator: the at sign (@). When prepended to an expression in PHP, any error messages that might be generated by that expression will be ignored.
If the 7.1.1.24 ini.track-errors feature is enabled, any error message generated by the expression will be saved in the global variable $php_errormsg. This variable will be overwritten on each error, so check early if you want to use it.

```
<?php
/* Intentional SQL error (extra quote): */
$res = @mysql_query( "select name, code from 'namelist" ) or
die( "Query failed: error was '$php_errormsg'" );
?>
```

See also error_reporting().

9.7.6 Opérateur d'exécutions

PHP supports one execution operator: backticks (``). Note that these are not single-quotes! PHP will attempt to execute the contents of the backticks as a shell command; the output will be returned (i.e., it won't simply be dumped to output; it can be assigned to a variable).

```
$output = `ls -al`;
echo "<pre>$output</pre>";
```

See also system(), passthru(), exec(), popen(), and escapeshellcmd().

9.7.7 Incrementing/Decrementing Operators

PHP supports C-style pre- and post-increment and decrement operators.

example	name	effect
++$a	Pre-increment	Increments $a by one, then returns $a.
$a++	Post-increment	Returns $a, then increments $a by one.
--$a	Pre-decrement	Decrements $a by one, then returns $a.
$a--	Post-decrement	Returns $a, then decrements $a by one.

Here's a simple example script:

```
<?php
echo "<h3>Postincrement</h3>";
$a = 5;
echo "Should be 5: " . $a++ . "<br>\n";
echo "Should be 6: " . $a . "<br>\n";
echo "<h3>Preincrement</h3>";
$a = 5;
echo "Should be 6: " . ++$a . "<br>\n";
echo "Should be 6: " . $a . "<br>\n";
echo "<h3>Postdecrement</h3>";
$a = 5;
echo "Should be 5: " . $a-- . "<br>\n";
echo "Should be 4: " . $a . "<br>\n";
echo "<h3>Predecrement</h3>";
$a = 5;
echo "Should be 4: " . --$a . "<br>\n";
echo "Should be 4: " . $a . "<br>\n";
?>
```

9.7.8 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(Or)	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 9.7.9 La précédence des opérateurs.

9.7.9 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.

Associativité	Opérateurs
gauche	,
gauche	or
gauche	xor
gauche	and
droite	print
gauche	= += -= *= /= .= %= &= |= ^= ~= ><= >>=
gauche	? :
gauche	||
gauche	&&
gauche	|
gauche	^
gauche	&
non-associative	== != ===
non-associative	< <= > >=
gauche	<< >>
gauche	+ - .
gauche	* / %
droite	! ~ ++ -- (int) (double) (string) (array) (object)
droite	[
non-associative	new

9.7.10 String Operators

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'assignement concaténant ('.='). Reportez vous à 9.7.2 Les opérateurs d'assignement pour plus de détails.

```
$a = "Bonjour ";
$b = $a . "Monde!"; // $b contient "Bonjour Monde!"
$a = "Bonjour ";
$a = $a . "Monde!"; // $a contient "Bonjour Monde!"
```

9.8 Types

PHP supporte les types suivants :

- 9.8.4 Les tableaux
- 9.8.2 Les nombres à virgule flottante
- 9.8.1 Entiers

- 9.8.5 Les objets
- 9.8.3 Les chaînes de caractères

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 (9.8.6.1 Transtypage) 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 9.8.6 Définition du type.

9.8.1 Entiers

Il est possible de spécifier les nombres entiers (integers) de la manière suivante :
$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)

9.8.2 Les nombres à virgule flottante

Les nombres à virgule flottante ("double") peuvent êtres spécifiés en utilisant la syntaxe suivante:
$a = 1.234;
$a = 1.2e3;

9.8.3 Les chaînes de caractères

Les chaînes de caractères peuvent être définies en utilisant deux types de délimiteurs.
Si la chaîne de caractères est délimitée par des guillemets doubles ("), les variables à l'intérieur de la chaîne seront évaluées, et remplacées par leur valeur. Comme en C ou en Perl, le caractère backslash (\) est utilisé pour protéger (échapper) un caractère spécial.

séquence	valeur
\n	Nouvelle ligne
\r	Retour à la ligne
\t	Tabulation horizontale
\\	Backslash
\$	Caractère $
\"	Guillemet double
\[0-7]{1,3}	Une séquence de caractère qui permet de rechercher un nombre en notation octale. @tab
\x[0-9A-Fa-f]{1,2}	Une séquence de caractère qui permet de rechercher un nombre en notation hexadécimale. @tab

Vous pouvez utiliser le caractère d'échappement backslash sur n'importe quel autre caractère, mais cela produira une alerte (si le niveau d'alerte maximal a été fixé).
Le deuxième moyen de délimiter une chaîne de caractère est d'utiliser les guillemets simples ('). Dans une telle chaîne de caractères, les variables *ne seront pas* évaluées, et le caractère backslash n'aura aucun effet (à deux exceptions près, pour "\\" et "\'", afin de pouvoir utiliser les caractères guillemets simples, et backslash dans la chaîne de caractères).
Another way to delimit strings is by using here doc syntax ("<<<"). One should provide an identifier after <<<, then the string, and then the same identifier to close the quotation. The closing identifier *must* begin in the first column of the line.

```
$str = <<<EOD
Example of string
spanning multiple lines
using heredoc syntax.
EOD;
```

Note : *Here doc support was added in PHP 4.*

Strings may be concatenated using the '.' (dot) operator. Note that the '+' (addition) operator will not work for this. Please see 9.7.10 String Operators for more information.
Characters within strings may be accessed by treating the string as a numerically-indexed array of characters, using C-like syntax. See below for examples.

```
<?php
/* Assigning a string. */
$str = "This is a string";
/* Appending to it. */
$str = $str . " with some more text";
/* Another way to append, includes an escaped newline. */
$str .= " and a newline at the end.\n";
/* This string will end up being '<p>Number: 9</p>' */
$num = 9;
$str = "<p>Number: $num</p>";
/* This one will be '<p>Number: $num</p>' */
$num = 9;
$str = '<p>Number: $num</p>';
/* Get the first character of a string  */
$str = 'This is a test.';
$first = $str[0];
/* Get the last character of a string. */
$str = 'This is still a test.';
$last = $str[strlen($str)-1];
?>
```

9.8.3.1 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.

```
$foo = 1 + "10.5";              // $foo est du type  double (11.5)
$foo = 1 + "-1.3e3";            // $foo est du type  double (-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  double (11)
```

Pour plus d'informations sur les conversions de type, voir les pages de man à propos de la fonction strtod(3).
If you would like to test any of the examples in this section, you can cut and paste the examples and insert the following line to see for yourself what's going on:

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

9.8.4 Les tableaux

Les tableaux ressemblent aux tables de hashage (tableaux associatifs) et aux tableaux indexés (vecteurs).

9.8.4.1 Tableaux à une dimension

PHP supporte les tableaux scalaires et les tableaux associatifs. En fait, il n'y a aucune différence entre les deux. Vous pouvez créer un tableau en utilisant les fonctions list() ou array(), ou bien en affectant explicitement chacune des valeurs.

```
$a[0] = "abc";
$a[1] = "def";
```

```
$b["foo"] = 13;
```

Vous pouvez aussi créer un tableau en ajoutant simplement les valeurs à ce tableau.

```
$a[] = "hello"; // $a[2] == "hello"
$a[] = "world"; // $a[3] == "world"
```

Un tableau peut être trié en utilisant les fonctions asort(), arsort(), ksort(), rsort(), sort(), uasort(), usort(), ou uksort() en fonction du type de classement que vous voulez.
Vous pouvez compter le nombre d'éléments qu'il y a dans un tableau en utilisant la fonction count().
Vous pouvez vous déplacer à l'intérieur d'un tableau en utilisant les fonctions next() et prev(). Un autre moyen de se déplacer dans un tableau est d'utiliser la fonction each().

9.8.4.2 Tableaux à plusieurs dimensions

Les tableaux à plusieurs dimensions sont extrêmement simples. Pour chaque dimension du tableau, vous ajouter une nouvelle [clef] à la fin:

```
$a[1]      = $f;             # tableau à une dimension
$a["foo"]  = $f;
$a[1][0]     = $f;           # tableau à deux dimensions
$a["foo"][2] = $f;           # (vous pouvez mélanger les indices associatifs et numériques)
$a[3]["bar"] = $f;           # (vous pouvez mélanger les indices associatifs et numériques)
$a["foo"][4]["bar"][0] = $f;   # tableau à quatre dimensions
```

En PHP3 il n'est pas possible de référencer un tableau à l'intérieur d'une chaîne. Par exemple, ceci ne fonctionne pas :

```
$a[3]['bar'] = 'Bob';
echo "Cela ne marche pas : $a[3][bar]";
```

En PHP3, l'exemple ci dessu va afficher : Cela ne marche pas : Array[bar]. L'opérateur de concaténation, peut être utilisé pour corriger cela :

```
$a[3]['bar'] = 'Bob';
echo "Cela ne marche pas : " . $a[3][bar];
```

En PHP4, cependant, le problème peut être contourné en entourant le tableau par des accolades :

```
$a[3]['bar'] = 'Bob';
echo "Cela marche  : {$a[3][bar]}";
```

Vous pouvez remplir un tableau à plusieurs dimensions par de nombreux moyens mais la méthode la plus simple à comprendre est l'utilisation de la fonction array(). Les deux exemples suivants montre comment remplir un tableau à une dimension:

```
# Exemple 1:
$a["color"]= "red";
$a["taste"]= "sweet";
$a["shape"]            = "round";
$a["name"]             = "apple";
$a[3]                  = 4;
# Exemple 2:
$a = array(
    "color" => "red",
    "taste" => "sweet",
    "shape" => "round",
    "name"  => "apple",
3       => 4
);
```

La fonction array() peut être emboîtée pour remplir un tableau à plusieurs dimensions :

```
<?
$a = array(
    "apple"  => array(
        "color"  => "red",
        "taste"  => "sweet",
        "shape"  => "round"
    ),
    "orange"  => array(
```

```
        "color"  => "orange",
        "taste"  => "tart",
        "shape"  => "round"
    ),
    "banana"  => array(
        "color"  => "yellow",
        "taste"  => "paste-y",
        "shape"  => "banana-shaped"
    )
);
echo $a["apple"]["taste"];    # will output "sweet"
?>
```

9.8.5 Les objets

9.8.5.1 Initialisation d'un objet

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

```
class foo {
function do_foo () {
echo "Doing foo.";
   }
}
$bar = new foo;
$bar->do_foo();
```

9.8.6 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.

```
$foo = "0";  // $foo est une chaîne de caractères (ASCII 48)
$foo++;      // $foo est la chaîne de caractères "1" (ASCII 49)
$foo += 1;   // $foo est maintenant du type entier (2)
$foo = $foo + 1.3;  // $foo est maintenant du type double (3.3)
$foo = 5 + "10 Little Piggies"; // $foo est du type entier (15)
$foo = 5 + "10 Small Pigs";     // $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 la fonction 9.8.3.1 Conversion de type.
Si vous voulez forcer le type d'une variable, vous pouvez vous reporter à la section 9.8.6.1 Transtypage. SI vous voulez changer le type d'une variable, utilisez settype().
If you would like to test any of the examples in this section, you can cut and paste the examples and insert the following line to see for yourself what's going on:

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

Note : *The behaviour of an automatic conversion to array is currently undefined.*

```
$a = 1;      // $a is an integer
$a[0] = "f";  // $a becomes an array, with $a[0] holding "f"
```

While the above example may seem like it should clearly result in $a becoming an array, the first element of which is 'f', consider this:

$a = "1"; // $a is a string
$a[0] = "f"; // What about string offsets? What happens?

Since PHP supports indexing into strings via offsets using the same syntax as array indexing, the example above leads to a problem: should $a become an array with its first element being "f", or should "f" become the first character of the string $a?
For this reason, as of PHP 3.0.12 and PHP 4.0b3-RC4, the result of this automatic conversion is considered to be undefined. Fixes are, however, being discussed.

9.8.6.1 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").
$foo = 10; // $foo is an integer
$bar = (double) $foo; // $bar est un double

Les conversions autorisées sont:

- (int), (integer) - type entier
- (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:
$foo = (int) $bar;
$foo = (int) $bar;

Le transtypage n'a pas toujours le résultat attendu. Par exemple :
Lorsque vous transtypez un scalaire ou une chaîne en tableau, la variable verra son contenu affecté au premier élément du tableau.
$var = 'ciao';
$arr = (array) $var;
echo $arr[0]; // outputs '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':
$var = 'ciao';
$obj = (object) $var;
echo $obj->scalar; // outputs 'ciao'

9.9 Les variables

9.9.1 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).
$var = "Bob";
$Var = "Joe";
echo "$var, $Var"; // outputs "Bob, Joe"

En PHP3, 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 une variable n'aura pas d'effet sur l'autre. Pour plus de détails sur ce genre d'assignement, reportez vous à 9.4 Les expressions.
PHP4 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'assignement est donc beaucoup plus rapide. Cela se fera notamment sentir dans des boucles, ou lors d'assignement de grands objets (tableaux).

Pour assigner par référence, ajoutze simplement un & (ET commercial) au début de 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 seule les variables nommées peuvent être assignées par référence.

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

9.9.2 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 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.*

9.9.2.1 Variables Apache

Ces variables sont créées par le serveur Apache 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, et on peut s'attendre à les retrouver.
Notez que peu d'entre elles seront accessible 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 données dans les entêtes 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'entête Accept: de la requête courant, si il y en a une.
  HTTP_ACCEPT_CHARSET

- Contenu de l'entête Accept-Charset: de la requête courante, si il existe. Par exemple : 'iso-8859-1,*,utf-8'.
  HTTP_ENCODING

- Contenu de l'entête Accept-Encoding: de la requête courante, si elle existe. Par exemple : 'gzip'.
  HTTP_ACCEPT_LANGUAGE

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

- Contenu de l'entête Connection: de la requête courante, si elle existe. Par exemple : 'Keep-Alive'.
  HTTP_HOST

- Contenu de l'entête 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'entête 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 capacité 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 donné à la directive SERVER_ADMIN (pour Apache), dans le fichier de configuration. Si le script est exécuté par un hôte virtuel, cela 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 (par le document root-) jusqu'au script courant, une fois que le serveur à fait une chemin traduction virtuel->réél.
  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'.

9.9.2.2 Variables d'environnement

Ces variables sont importées dans l'espace de nom global de PHP's, depuis l'environnement sous lequel PHP fonctionne. Beaucoup d'entre elles sont fournies par le shell qui exécute PHP et différents systèmes étant suceptibles 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'environments inclues les variables CGI, placées ici, quelque fois la méthode d'éxécution de PHP (CGI ou module).

9.9.2.3 Variables PHP

Ces variables sont créées par PHP lui_même.

argv

- Tableau des rguments passées au script. Lorsque le script est appelé en ligne de commande, cela dconne 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 il 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 variable a été activé avec la directive générale 7.1.1.25 ini.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 variable a été activé avec la directive générale 7.1.1.25 ini.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 variable a été activé avec la directive générale 7.1.1.25 ini.track-vars ou avec la directive locale <?php_track_vars?>.

9.9.3 Portée des variables

La portée d'une variable dépends du contexte dans lequel la variable est définie. Pour la plupart 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:

```
$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 :

```
$a = 1; /* portée global */
Function Test () {
echo $a; /* portée locale */
}
Test ();
```

Le script n'affichera rien à l'écran car la fonction echo() utilise la variable local $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'intérieure de chaque fonction afin de pouvoir être utilisée dans cette fonction. Par exemple:

```
$a = 1;
$b = 2;
Function Sum () {
global $a, $b;
    $b = $a + $b;
}
Sum ();
```

```
echo $b;
```

Le script ci-dessus va afficher la valeur "3". En déclarant global les variables $a et $b localement dans la fonction, 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 pécédent exemple peut être réécrit de la manière suivante:

```
$a = 1;
$b = 2;
Function Sum () {
   $GLOBALS["b"] = $GLOBALS["a"] + $GLOBALS["b"];
}
Sum ();
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:

```
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 static:

```
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 static 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:

```
Function Test () {
static $count = 0;
   $count++;
echo $count;
if ($count < 10) {
Test ();
   }
   $count--;
}
```

9.9.4 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 affecté et utilisé dynamiquement. Une variable classique est affecté avec l'instruction suivante:

```
$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, *hello*, peut être utilisé comme le nom d'une variable en utilisant le "$$" précédent la variable. c'est-à-dire

```
$$a = "world";
```

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

```
echo "$a ${$a}";
```

produira le même affichage que :

```
echo "$a $hello";
```

c'est-à-dire : *hello world*.
Afin de pouvoir utiliser les variables dynamiques avec les tableaus, vous avez a 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 ambiguité est la suivante: ${$a[1]} pour le premier cas, et ${$a}[1] pour le deuxième.

9.9.5 Variables externes à PHP

9.9.5.1 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:

```
<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 entré 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 :

```
<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="beer[]">
      <option value="warthog">Warthog
      <option value="guinness">Guinness
      <option value="stuttgarter">Stuttgarter Schwabenbräu
      </select>
   <input type="submit">
</form>
```

Si l'option "track_vars" est activée, soit par l'option de compilation 7.1.1.25 ini.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.

9.9.5.1. 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 oú l'utilisateur à cliqué. Les utilisateurs expérimentés remarquerons que les noms de variables sont transmis avec une virgule à la place du caractère "_", mais le PHP fait la conversion automatiquement.

9.9.5.2 HTTP Cookies

Le PHP supporte les cookies HTTP de manière totalement transparente, comme défini dans les Netscape's Spec. 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 appelé 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 :
SetCookie ("MyCookie[]", "Testing", 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 un "shopping cart", vous devez implémenter un compteur et l'incrémenter au fur et à mesure. C'est-à-dire:

```
$Count++;
SetCookie ("Count", $Count, time()+3600);
SetCookie ("Cart[$Count]", $item, time()+3600);
```

9.9.5.3 Variables d'environnement

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

```
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 sûre de la valeur. La fonction getenv() peut être utilisé pour récupéré la valeur des variables d'environnement. Vous pouvez aussi affecter une variable d'environnement grâce à la fonction putenv().

9.9.5.4 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 :

```
$varname.ext;  /* invalid variable name */
```

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).

9.9.5.5 Détermination du type des variables

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

9.10 Les références

9.10.1 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.

9.10.2 Que font les références

Les références vous permettent de faire pointer deux variables sur le même contenu. Par exemple, lorsque vous faites :

```
$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.*

Le deuxième interet 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 :

```
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**.
Le troisième interêts des références est de 9.10.4 Retourner des références.

9.10.3 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 de à quoi on peut s'attendre :

```
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 **$GLOBALS["baz"]**. Il n'y a pas moyen de le lier **$bar** à quelque chose d'autre en utilisant le mécanisme de référence, car **$bar** n'est pas accessible dans la fonction foo. (il est représenté par **$var**, mais **$var** possède la même valeur, mais n'est pas relié par la table des symboles).

9.10.4 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

```
function &find_var ($param) {
   ...code...
return $found_var;
}
$foo =& find_var ($bar);
$foo->x = 2;
```

Dans cet exemple, la propriété de l'objet est retourné dans **find_var** et lui sera affecté, 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 indiquee que vous retournez par référence (pas une copie habituelle), et pour indiquer que vous assigner aussi par référence (pas la copie habituelle).*

9.10.5 Détruire une références

Lorsque vous détruiser 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,

```
$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.

9.10.6 Repérer une référence

De nombreuses syntaxes de PHP sont implementées via le mécanisme de référence, et tout ce qui a été vu concernant les liaisons entre variables s'applique à ces syntaxe. Par exemple, le passage et le retour d'arguments par référence. Quelques autres exemples de syntaxes :

9.10.6.1 global références

Lorsque vous déclarez une variable comme global $var vous faites en réalité une référence sur une variable globale.

```
$var =& $GLOBALS["var"];
```

Cela signifie notamment que si vous détruisez **$var**, vous ne détruirez par la variable globale.

9.10.6.2 $this

Dans une méthode d'objet, **$this** est toujours une référence sur l'objet courant.

10 Fonctions

10.1 spécifiques à Apache

10.1.1 apache_lookup_uri

class apache_lookup_uri (string **filename**)
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 : *Note: apache_lookup_uri ne fonctionne que lorsque le PHP est installé sous la forme d'un module Apache.*

10.1.2 apache_note

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 la fonction est appelée avec deux arguments, la variable note_name la valeur note_value et la fonction retournera la valeur précédente de la variable note_name.

10.1.3 getallheaders

array getallheaders
Cette fonction renvoie un tableau associatif de tous les entêtes HTTP correspondants à la requête courante.
Note : *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 en CGI. Utilisez la fonction phpinfo() pour avoir une liste de toutes les variables d'environnement disponibles.*

```
$headers = getallheaders();
while (list($header, $value) = each($headers)) {
echo "$header: $value<br>\n";
}
```

Cette exemple est un exemple d'affichage de toutes les entêtes de la requête courante. Note : *Note: La fonction getallheaders() ne fonctionne que si PHP est installé comme module **Apache**.*

10.1.4 virtual

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 entête valide, c'est-à-dire, au minimum une entête "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.

10.2 Tableaux

10.2.1 array

array array
Retourne un tableau, créé à partir des paramètres fournis. Les paramètres peuvent avoir un index, fourni sous la forme clé => valeur. Note : *array() n'est pas une fonction standard, elle existe simplement pour représenter littéralement des tableaux.*

Les exemples suivants montrent la construction de tableaux bi-dimensionnels, l'assignation de clés pour les tableaux associatifs, et comment écarter certains intervalle d'indices numériques.

```
$fruits = array (
   "fruits"  => array("a"=>"orange", "b"=>"banane", "c"=>"pomme"),
   "numbres" => array(1, 2, 3, 4, 5, 6),
   "trous"   => array("premier", 5 => "deuxième", "troisième")
);
```

Voir aussi : list().

10.2.2 array_count_values

array array_count_values (array **input**)
array_count_values() retourne un tableau contenant les valeurs du tableau **input** comme clés et leurs fréquence **input** comme valeur.

```
$array = array(1, "bonjour", 1, "monde", "bonjour");
array_count_values($array); // retourne array(1=>2, "bonjour"=>2, "monde"=>1)
```

Note : *Cette fonction a été ajoutée dans PHP 4.0.*

10.2.3 array_flip

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

```
$trans = array_flip ($trans);
```

```
$original = strtr ($str, $trans);
```

Note : *Cette fonction a été ajoutée dans PHP 4.0.*

10.2.4 array_keys

array array_keys (array **input**, mixed **search_value**)

array_keys() retourne les clés numériques et litérales du tableau **input**.
Si l'option **search_value** est spécifiée, seule les clés ayant cette valeur seront retournées. Sinon, toutes les clés de **input** sont retournées.

```
$array = array(0 => 100, "couleur" => "rouge");
array_keys ($array);      // retourne array (0, "couleur")
$array = array(1, 100, 2, 100);
array_keys ($array, 100);  //  retourne array (0, 2)
```

Voir aussi array_values(). Note : *Cette fonction a été ajoutée dans PHP 4.0.*

10.2.5 array_merge

array array_merge (array **array1**, array **array2**, **...**)

array_merge() rassemble les éléments de plusieurs tableaux ensemble, 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.

```
$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" => "trapézoïde").

Note : *Cette fonction a été ajoutée dans PHP 4.0.*

10.2.6 array_pad

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, si il est négatif, il est complété à gauche. Si la valeurs absolue de **pad_size** est plus petite que la taille du tableau **input** alors le tableau n'est pas complété.

```
$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
```

10.2.7 array_pop

mixed array_pop (array **array**)

array_pop() array_pop() dépile et retourne le dernier élément du tableau **array**, le raccourcissant d'un élément.

```
$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 : *Cette fonction a été ajoutée dans PHP 4.0.*

10.2.8 array_push

int array_push (array **array**, mixed **var**, **...**)
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 :
$array[] = $var;
repeté pour chaque **var**.
Retourne le nouveau nombre d'éléments du tableau.

```
$stack = array (1, 2);
array_push($stack, "+", 3);
```

Cet exemple fait que $stack a 4 éléments: 1, 2, "+", et 3.
Voir aussi: array_pop(), array_shift(), et array_unshift(). Note : *Cette fonction a été ajoutée dans PHP 4.0.*

10.2.9 array_reverse

array array_reverse (array **array**)
array_reverse() prend un tableau **array** et retourne un nouveau tableau qui contient les mêmes éléments mais dans l'ordre inverse.

```
$input = array ("php", 4.0, array ("rouge", "vert"));
$result = array_reverse ($input);
```

Au final, $result contient (array ("rouge", "vert"), 4.0, "php"). Note : *Cette fonction a été ajoutée dans PHP 4.0 Beta 3.*

10.2.10 array_shift

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.

```
$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 : *Cette fonction a été ajoutée dans PHP 4.0.*

10.2.11 array_slice

array array_slice (array **array**, int **offset**, int **length**)
array_slice() retourne une série d'élément du tableau **array** commencant 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 commencant à la fin du tableau **array**.
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** précisé jusqu'à la fin du tableau.

```
$input = array ("a", "b", "c", "d", "e");
$output = array_slice ($input, 2);      // retourne "c", "d", et "e"
$output = array_slice ($input, 2, -1);  // 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 : *Cette fonction a été ajoutée dans PHP 4.0.*

10.2.12 array_splice

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 commencant 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 précisé 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 **loffset** 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 à array avec array(), à moins que cette variable ne soit elle même un tableau.
The following equivalences hold:

```
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)
```

Retourne le tableau des éléments supprimés.

```
$input = array("red", "green", "blue", "yellow");
array_splice($input, 2);      // $input est array("red", "green")
array_splice($input, 1, -1);  // $input est array("red", "yellow")
array_splice($input, 1, count($input), "orange");
                              // $input est array("red", "orange")
array_splice($input, -1, 1, array("black", "maroon"));
                              // $input est array("red", "green",
                              //         "blue", "black", "maroon")
```

Voir aussi array_slice(). Note : *Cette fonction a été ajoutée dans PHP 4.0.*

10.2.13 array_unshift

int array_unshift (array **array**, mixed **var**, **...**)

array_unshift() ajoute les éléments passé en arguments au début du tableau **array**. Notez que les éléments sont ajoutés comme un tout, et qu'ils restent dans le même ordre.
Retourne le nouveau nombre d'éléments du tableau **array**.

```
$queue = array("p1", "p3");
array_unshift($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 : *Cette fonction a été ajoutée dans PHP 4.0.*

10.2.14 array_values

array array_values (array **input**)
array_values()retourne les valeurs d'un tableau **input**.

```
$array = array("taille" => "XL", "couleur" => "or");
array_values($array);    // // retourne array("XL", "or")
```

Note : *Cette fonction a été ajoutée dans PHP 4.0.*

10.2.15 array_walk

int array_walk (array **arr**, string **func**, mixed **userdata**)
Exécute la fonction **func** avec chaque élément du tableau **arr**. Les éléments sont passés en tant que premier argument de la fonction **func**;
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() call, or by using error_reporting().
Note : *Si **func** doit travailler avec les véritables valeur 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 dans PHP 4.0.*
Dans PHP 4, reset() doit être appelés si nécessaire, car array_walk() ne réinitialise pas automatiquement le tableau.

```
$fruits = array("d"=>"citron","a"=>"orange","b"=>"banane","c"=>"pomme");
function test_alter( $item1 ) {
   $item1 = 'bidon';
}
function test_print( $item2 ) {
echo "$item2<br>\n";
}
array_walk( $fruits, 'test_print' );
array_walk( $fruits, 'test_alter' );
array_walk( $fruits, 'test_print' );
```

Voir aussi each() et list().

10.2.16 arsort

void arsort (array **array**)
Cette fonction trie un tableau de telle manière que la corrélation entre les index et les valeurs soient conservées. L'usage principal est lors de tri de tableaux associatifs oú l'ordre des éléments est important.

```
$fruits = array("d"=>"papaye","a"=>"orange","b"=>"banane","c"=>"ananas");
arsort ($fruits);
for (reset ($fruits); $key = key ($fruits); next ($fruits)) {
```

```
echo "fruits[$key] = ".$fruits[$key]."\n";
}
```

Cet exemple va afficher: fruits[a] = orange fruits[d] = papaye fruits[b] = banane fruits[c] = ananas Les fruits ont été triés en ordre alphabétique inverse, et leur index respectifs ont été conservé.
Voir aussi: asort(), rsort(), ksort(), et sort().

10.2.17 asort

void asort (array **array**)
Cette fonction trie un tableau de telle manière que la corrélation entre les index et les valeurs soient conservées. L'usage principal est lors de tri de tableaux associatifs oú l'ordre des éléments est important.

```
$fruits = array("d"=>"papaye","a"=>"orange","b"=>"banane","c"=>"ananas");
asort($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[d] = papaye fruits[a] = orange Les fruits ont été triés par ordre alphabétique, et leur index respectifs ont été conservé.
Voir aussi arsort(), rsort(), ksort(), et sort().

10.2.18 compact

array compact (**string varname) array varnames**, **...** |
compact() accepte différents paramètres. Les paramètres peuvent être des variables contenant des chaînes, ou un tableau de chaîne, qui peut contenir d'autres tableau 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(). Retourne le tableau ainsi créé.

```
$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 : *Cette fonction a été ajoutée dans PHP 4.0.*

10.2.19 count

int count (mixed **var**)
Retourne le nombre d'élément dans **var**, qui est généralement un tableau (et tout le reste n'aura qu'un élément).
Retourne 1 si la variable n'est pas un tableau.
Retourne 0 si la variable n'est pas créée. count() va retourner 0 pour une variable non créée, mais il peu aussi retourner 0 pour un tableau vide. Utilisez plutôt la commande isset() pour savoir si une variable existe ou pas.

Voir aussi: sizeof(), isset(), et is_array().

10.2.20 current

mixed current (array **array**)
Chaque tableau entretien un pointeur interne, qui est initialisé lors lorsque le premier élément est inséré dans le tableau.
La fonction current() ne fait que retourner l'élément courant pointé par le pointeur interne. current() ne déplace pas le pointeur. Si le pointeur est au dela du dernier élément de la liste, current() retourne faux. Si le tableau des éléments vides ou des zéros (0 ou "", la chaîne vide) alors cette fonction retournera false pour ces éléments. Il est donc impossible de determiner 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().

10.2.21 each

array each (array **array**)
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* and *value* contienent la valeur.
Si le pointeur interne de fichier est au dela de la fin du tableau, each() retourne faux.

```
$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'
-

```
$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:

```
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 éléments, ou sur le dernier élément, lorsqu'on arrive à la fin.
Voir aussi key(), list(), current(), reset(), next(), et prev().

10.2.22 end

end (array **array**)
end() déplace le pointeur interne du tableau **array** jusqu'au dernier élément.
Voir aussi: current(), each(), end(), next(), et reset().

10.2.23 extract

void extract (array **var_array**, int **extract_type** , string **prefix**)
Cette fonction sert à exporter un tableau vers la table des symboles. Elle prend un tableau associatif **var_array** et crée les variables dont les noms sont les index de ce tableau, et leur affecte la valeur associée. Pour chaque paire clé/valeur, cette fonction crée une variable, avec les paramètres **extract_type** et **prefix**.
extract() vérifie l'existence de la variable avant de la créer. La manière de traiter les collisions est déterminée 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.

Si **extract_type** est omis, extract() utilise EXTR_OVERWRITE par défault.
Notez que **prefix** n'est nécessaire que pour les valeurs de **extract_type** suivantes : EXTR_PREFIX_SAME et EXTR_PREFIX_ALL.
extract() vérifie que les clés constitue un nom de variable valide, et si c'est le cas, procède à son exportation.
Une utilisation possible de cette fonction est l'exportation vers la table des symboles de tableau de variables retourné par la fonction wddx_deserialize().

```
<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
blue, large, sphere, medium

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.

10.2.24 in_array

bool in_array (mixed **needle**, array **haystack**)
Recherche **needle** dans **haystack** et retourne vrai si il s'y trouve, ou faux sinon.

```
$os = array("Mac", "NT", "Irix", "Linux");
if (in_array("Irix", $os))
print "Irix trouve";
```

Note : *Cette fonction a été ajoutée dans PHP 4.0.*

10.2.25 key

mixed key (array **array**)
key() retourne l'index de la clé courante dans un tableau.
Voir aussi: current(), et next()

10.2.26 krsort

int krsort (array **array**)
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.

```
$fruits = array("d"=>"papaye","a"=>"orange","b"=>"banane","c"=>"ananas");
ksort($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 asort(), arsort(), ksort() sort(), et rsort().

10.2.27 ksort

int ksort (array **array**)
Trie un tableau suivant les clés, en maintenant la correspondance entre les clés et les valeurs. Cette fonction est pratique pour les tableaux associatifs.

```
$fruits = array("d"=>"papaye","a"=>"orange","b"=>"banane","c"=>"ananas");
ksort($fruits);
for(reset($fruits); $key = key($fruits); next($fruits)) {
echo "fruits[$key] = ".$fruits[$key]."\n";
}
```

Cet exemple va afficher : fruits[a] = orange fruits[b] = banane fruits[c] = ananas fruits[d] = citron
Voir aussi asort(), arsort(), sort(), et rsort().

10.2.28 list

void list
Tout comme array(), ce n'est pas une véritable fonction, mais une construction syntaxique, qui permet d'assigner une série de variable en une seule ligne.

```
<table>
 <tr>
  <th>Nom des emploies name</th>
  <th>Salaire</th>
```

```
 </tr>
<?php
$result = mysql($conn, "SELECT id, nom, salaire FROM employe");
while (list($id, $nom, $salaire) = mysql_fetch_row($result)) {
print(" <tr>\n".
        "  <td><a href=\"info.php3?id=$id\">$nom </a></td>\n".
        "  <td>$salaire</td>\n".
        " </tr>\n");
}
?></table>
```

Voir aussi: each(), array().

10.2.29 next

mixed next (array **array**)
retourne l'élément suivant du tableau, ou false si 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. Si le tableau contient des éléments vides ou des zéros, cette fonction retournera false pour ces éléments. Pour passer proprement en revue un tableau, il faut utiliser each().

Voir aussi: current(), end() prev() et reset()

10.2.30 pos

mixed pos (array **array**)
C'est une fonction alias de current().
Voir aussi: end(), next(), prev() et reset().

10.2.31 prev

mixed prev (array **array**)
Repositionne le pointeur interne de tableau à la dernière place qu'il occupait, ou bien retourne faux si il ne reste plus d'éléments. Si le tableau contient des éléments vides, cette fonction retournera faux 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()

10.2.32 range

array range (int **low**, int **high**)
range() range() retourne un tableau contenant tous les entiers depuis **low** jusqu'à **high**, inclus.
Voir shuffle() pour un exemple d'utilisation.

10.2.33 reset

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().

10.2.34 rsort

void rsort (array **array**)
Cette fonction effectue un trie en ordre décroissant (du plus grand au plus petit)

```
$fruits = array("papaye","orange","banane","ananas");
rsort($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: arsort(), asort(), ksort(), sort(), et usort().

10.2.35 shuffle

void shuffle (array **array**)
Cette fonction mélange les éléments d'un tableau.

```
$numbers = range (1,20);
srand (time());
shuffle ($numbers);
while (list(, $number) = each ($numbers)) {
echo "$number ";
}
```

Voir aussi arsort(), asort(), ksort(), rsort(), sort() and usort().

10.2.36 sizeof

int sizeof (array **array**)
Retourne le nombre d'élément d'un tableau.
Voir aussi: count()

10.2.37 sort

void sort (array **array**)
Cette fonction trie le tableau array. Les éléments seront triés du plus petit au plus grand.

```
$fruits = array("papaye","orange","banane","ananas");
sort($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 classé dans l'ordre alphabétique.
Voir aussi: arsort(), asort(), ksort(), rsort(), et usort().

10.2.38 uasort

void uasort (array **array**, function **cmp_function**)
Cette fonction trie un tableau en conservant la correspondance entre les index et leurs valeurs. Cette fonction sert essentiellement lors de tri de tableaux associatifs oú l'ordre des éléments est significatif. La fonction de comparaison utilisée est définie par l'utilisateur.

10.2.39 uksort

void uksort (array **array**, function **cmp_function**)
Cette fonction va trier les clés du tableau en utilisant une fonction définie par l'utilisateur. Si un tableau qui doit être trié avec un critère complexe, il est préférable d'utiliser cette fonction.

```
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: arsort(), asort(), uasort(), ksort(), rsort(), et sort().

10.2.40 usort

void usort (array **array**, function **cmp_function**)
Cette fonction 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 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.

```
function cmp($a,$b) {
if ($a == $b) return 0;
return ($a > $b) ? -1 : 1;
}
$a = array(3,2,5,6,1);
usort($a, cmp);
while(list($key,$value) = each($a)) {
echo "$key: $value\n";
}
```

Cet exemple affichera : 0: 6 1: 5 2: 3 3: 2 4: 1 Note : *Evidemment dans ce cas trivial, rsort() serait plus approprié.*
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: arsort(), asort(), ksort(), rsort() and sort().

10.3 Aspell

Les fonctions Aspell vous permettent de vérifier l'orthographe d'un mot, et d'offrir des suggestions de corrections. Plusieurs langues sont disponible, comme l'allemand, le suédois et le danois.
Vous avez besoin de la librairie Aspell, disponible à : http://aspell.sourceforge.net/

10.3.1 aspell_new

int 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.

```
$aspell_link=aspell_new("english");
```

10.3.2 aspell_check

boolean aspell_check (int **dictionary_link**, string **word**)
aspell_check() vérifie l'orthographe d'un mot et retourne TRUE si l'orthographe est correcte, et FALSE sinon.

```
$aspell_link=aspell_new("english");
if (aspell_check($aspell_link,"testt")) {
echo "L'orthographe est correcte.";
} else {
echo "Désolé, l'orthographe est incorrecte.";
```

```
}
```

10.3.3 aspell_check-raw

boolean aspell_check_raw (int **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.

```
$aspell_link=aspell_new("english");
if (aspell_check_raw($aspell_link,"testt")) {
echo "This is a valid spelling";
} else {
echo "Sorry, wrong spelling";
}
```

10.3.4 aspell_suggest

array aspell_suggest (int **dictionary_link**, string **word**)
aspell_suggest() retourne un tableau contenant les orthographes possibles d'un mot mal formé.

```
$aspell_link=aspell_new("english");
if (!aspell_check($aspell_link,"testt")) {
   $suggestions=aspell_suggest($aspell_link,"testt");
for($i=0; $i < count($suggestions); $i++) {
echo "Orthographe envisageable : " . $suggestions[$i] . "<br>";
   }
}
```

10.4 mathématiques sur des nombres de taille arbitraire

Ces fonctions ne sont disponibles que si l'option de configuration --enable-bcmath a été activée lors de la compilation.

10.4.1 bcadd

string bcadd (string **left operand**, string **right operand**, int **scale**)
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().

10.4.2 bccomp

int bccomp (string **left operand**, string **right operand**, int **scale**)
Compare l'opérande **left operand** avec l'opérande **right operand** et renvoie le résultat sous forme de valeur numérique (integer). 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.

10.4.3 bcdiv

string bcdiv (string **left operand**, string **right operand**, int **scale**)
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().

10.4.4 bcmod

string bcmod (string **left operand**, string **modulus**)

Retourne le reste de la division entre **left operand** en utilisant **modulus**.
Voir aussi bcdiv().

10.4.5 bcmul

string bcmul (string **left operand**, string **right operand**, int **scale**)
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().

10.4.6 bcpow

string bcpow (string **x**, string **y**, int **scale**)
Elève **x** à la puissance **y**. Le paramètre optionnel **scale** définit le nombre de chiffres après la virgule dans le résultat.
Voir aussi bcsqrt().

10.4.7 bcscale

string bcscale (int **scale**)
Cette fonction définit la précision par défaut pour toutes les fonctions mathématiques sur des nombres de taille arbitraire qui suivent et qui omettent le paramètre **scale**.

10.4.8 bcsqrt

string bcsqrt (string **operand**, int **scale**)
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().

10.4.9 bcsub

string bcsub (string **left operand**, string **right operand**, int **scale**)
Soustrait l'opérande **right operand** de 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().

10.5 de calendrier

Les fonctions de calendrier ne sont disponibles que si l'extension calendrier (calendar) a été compilée. Elle est située dans dl/calendar. Lisez le fichier dl/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 JC). 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.

10.5.1 JDToGregorian

string jdtogregorian (int **julianday**)
Converti le nombre de jours du calendrier Julien en une chaîne contenant une date du calendrier grégorien, au format "mois/jour/année".

10.5.2 GregorianToJD

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.

```
<?php
```

```
$jd = GregorianToJD(10,11,1970);
echo("$jd\n");
$gregorian = JDToGregorian($jd);
echo("$gregorian\n");
?>
```

10.5.3 JDToJulian

string jdtojulian (int **julianday**)
Converti le nombre de jours du calendrier Julien en une chaîne contenant la date du calendrier Julien, au format "mois/jour/année".

10.5.4 JulianToJD

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.

10.5.5 JDToJewish

string jdtojewish (int **julianday**)
Converti le nombre de jours du calendrier julien en date du calendrier juif.

10.5.6 JewishToJD

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.

10.5.7 JDToFrench

string jdtofrench (int **month**, int **day**, int **year**)
Converti le nombre de jours du calendrier julien en date du calendrier français républicain.

10.5.8 FrenchToJD

int frenchtojd (int **month**, int **day**, int **year**)
Converti 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.

10.5.9 JDMonthName

string jdmonthname (int **julianday**, int **mode**)
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	
0	Grégorien - abrégé	
1	Grégorien	
2	Julien - abrégé	
3	Julien	
4	Juif	
5	Républicain français	

10.5.10 JDDayOfWeek

mixed jddayofweek (int **julianday**, int **mode**)
Retourne le numéro du jour de la semaine. Peut retourner une chaîne ou un entier, en fonction du mode.

Mode	Signification
0	Retourne le numéro du jour comme un entier (0=dimanche, 1=lundi, etc.) @tab
1	Retourne une chaîne contenant le nom du jour (anglais grégorien) @tab
2	Retourne une chaîne contenant le nom abrégé du jour de la semaine (anglais grégorien). @tab

10.5.11 easter_date

int easter_date (int **year**)
Retourne un timestamp UNIX pour Pâques, à minuit, pour une année donnée. Si l'année n'est pas précisée, c'est l'année en cours qui est utilisée.
ATTENTION: cette fonction génére une alerte (Warning) si la date tombe hors de la zone de validité des timestamps UNIX (i.e. avant 1970 ou après 2037).

```
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 Pâques 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 Pâques avant 1970 et apres 2037.

10.5.12 easter_days

int easter_days (int **year**)
Retourne le nombre de jour entre le 21 Mars et Pâques, pour une année donnée. Si l'année n'est pas précisée, l'année en cours est utilisée par défaut.
Cette fonction peut être utilisée à la place de easter_date() pour calculer la date de Pâques, pour les années qui tombent hors de l'intervalle de validité des timestamps UNIX (i.e. avant 1970 ou après 2037).

```
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 Pâques 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().

10.6 API CCVS

Ces fonctions permettent d'accéder directement à CCVS depuis un script PHP. CCVS est la solution de RedHat au problème de l'intermédiaire ("middle-man") lors de l'utilisation de cartes de crédits. Elles vous permettent de vous adresser directement aux institutions de cartes de crédits, via votre "*nix box" (NDtraducteur : ??) et un modem. En utilisant le module CCVS de PHP, vous pouvez directement traiter les cartes de crédits depuis vos scripts PHP. Les explications suivantes vous éclaireront sur le processus à suivre.
Pour activer le support CCVS de PHP, vérifiez d'abord votre dossier d' installation CCVS. You devez ensuite

configurer PHP avec l'option --with-ccvs. Si vous utilisez cette option sans spécifier le chemin de votre installation CCVS, PHP essayera de la trouver dans le dossier d'installation par défaut (`/usr/local/ccvs'). Si votre installation CCVS n'est pas standard, configurez PHP avec l'option --with-ccvs=$ccvs_path, où $ccvs_path est le chemin de votre installation CCVS. Notez que CCVS requièrent l'existence de $ccvs_path/lib et $ccvs_path/include, ainsi que de cv_api.h dans le dossier `include', et libccvs.a dans le dossier `"lib"'.
De plus, un processus ccvsd doit tourner lors de l'utilisation de vos scripts PHP. Assurez vous que les processus PHP utilisent le m□me utilisateur que celui qui a installé CCVS (i.e. si vous avez installé CCVS avec l'utilisateur 'ccvs', vos processus PHP doivent tourner aussi sous l'utilisateur 'ccvs').
Plus d'informations sur CCVS sont disponibles à l'adresse suivante :
http://www.redhat.com/products/ccvs.
Cette documentation est en cours d'élaboration. Jusqu'à sa finalisation, RedHat entretient une documentation légèrement en retard, mais très utile à
http://www.redhat.com/products/ccvs/support/CCVS3.3docs/ProgPHP.html.

10.7 Classe/Objet

10.7.1 get_class_methods

array get_class_methods (string **class_name**)
get_class_methods() retourne un tableau contenant les noms des méthodes de la classe **class_name**.

10.7.2 get_class_vars

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**.

10.7.3 get_object_vars

array get_object_vars (object **obj**)
get_object_vars() retourne un tableau contenant les valeurs des attributs de la classe **class_name**.

10.7.4 method_exists

bool 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.

10.8 Support COM pour Windows

Ces fonctions ne sont disponibles que sous les versions Windows de PHP. Elles ont été ajoutées dans PHP4.

10.8.1 com_load

string com_load (string **module name**, string **server name**)

10.8.2 com_invoke

mixed com_invoke (resource **object**, string **function_name**, mixed **paramètres de fonction, ...**)

10.8.3 com_propget

mixed com_propget (resource **object**, string **property**)

10.8.4 com_get

mixed com_get (resource **object**, string **property**)

10.8.5 com_propput

void com_propput (resource **object**, string **property**, mixed **value**)

10.8.6 com_propset

void com_propset (resource **object**, string **property**, mixed **value**)
Cette fonction est un alias de com_propput().

10.8.7 com_set

void com_set (resource **object**, string **property**, mixed **value**)
Cette fonction est un alias de com_set().

10.9 ClibPDF

ClibPDF vous permet de créer des documents PDF avec PHP. Cette librairie est disponible à FastIO 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 module, 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().
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.

```
<?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 converti en script PHP qui utilise l'extension ClibPDF :

```
<?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 + $margin), 1.0);
cpdf_set_page_animation($pdf, 4, 0.5, 0, 0, 0);  /* wipe */
cpdf_translate($pdf, $radius + $margin, $radius + $margin);
cpdf_save($pdf);
cpdf_setrgbcolor($pdf, 0.0, 0.0, 1.0);
  /* minute strokes */
cpdf_setlinewidth($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($pdf);
  /* 5 minute strokes */
cpdf_setlinewidth($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);
  }
  $ltime = getdate();
  /* draw hour hand */
cpdf_save($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);
  /* draw minute hand */
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);
  /* draw second hand */
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($pdf);
  /* draw little circle at center */
cpdf_circle($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);
?>
```

10.9.1 cpdf_global_set_document_limits

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. Cette fonction doit être appelé avant cpdf_open() pour être effective. Elle fixe les limites de tous les documents ouverts après.
Voir aussi cpdf_open().

10.9.2 cpdf_set_creator

void cpdf_set_creator (string **creator**)
cpdf_set_creator() fixe le créateur d'un document PDF.
Voir aussi cpdf_set_subject(), cpdf_set_title(), cpdf_set_keywords().

10.9.3 cpdf_set_title

void cpdf_set_title (string **title**)
cpdf_set_title() fixe le titre d'un document PDF.
Voir aussi cpdf_set_subject(), cpdf_set_creator(), cpdf_set_keywords().

10.9.4 cpdf_set_subject

void cpdf_set_subject (string **subject**)
cpdf_set_subject() fixe le sujet d'un document PDF.
Voir aussi cpdf_set_title(), cpdf_set_creator(), cpdf_set_keywords().

10.9.5 cpdf_set_keywords

void cpdf_set_keywords (string **keywords**)
cpdf_set_keywords() fixe les mot clés d'un document PDF.
Voir aussi cpdf_set_title(), cpdf_set_creator(), cpdf_set_subject().

10.9.6 cpdf_open

int cpdf_open (int **compression**, string **filename**)
cpdf_open() ouvre un nouveau document PDF. Le premier paramètre 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(), cpdf_output_buffer().

10.9.7 cpdf_close

void cpdf_close (int **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().

10.9.8 cpdf_page_init

void cpdf_page_init (int **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().

10.9.9 cpdf_finalize_page

void cpdf_finalize_page (int **pdf document**, int **page number**)
cpdf_finalize_page() termine la page de numéro **page number**. Cette fonction fait que qu'une sauvegarde mémoire. Les pages terminées prennent moins de place, mais ne peuvent plus être modifiées.
Voir aussi cpdf_page_init().

10.9.10 cpdf_finalize

void cpdf_finalize (int **pdf document**)
cpdf_finalize() termine un document. Vous devez toujours appeler cpdf_close() après.
Voir aussi cpdf_close().

10.9.11 cpdf_output_buffer

void cpdf_output_buffer (int **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().

10.9.12 cpdf_save_to_file

void cpdf_save_to_file (int **pdf document**, string **filename**)
cpdf_save_to_file() 'crit un document PDF dans un fichier, si il a été créé en mémoire. Cette fonction n'est pas nécessaire si un nom de fichier a été fourni lors de l'appel à cpdf_open().
Voir aussi cpdf_output_buffer(), cpdf_open().

10.9.13 cpdf_set_current_page

void cpdf_set_current_page (int **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().

10.9.14 cpdf_begin_text

void cpdf_begin_text (int **pdf document**)
cpdf_begin_text() démarre une section de texte. Elle doit être terminée avec cpdf_end_text().

```
<?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().

10.9.15 cpdf_end_text

void cpdf_end_text (int **pdf document**)
cpdf_end_text() termine une section de texte, commencée avec cpdf_begin_text().

```
<?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().

10.9.16 cpdf_show

void cpdf_show (int **pdf document**, string **text**)
cpdf_show() imprime la chaîne **text**, à la position courante.
Voir aussi cpdf_text(), cpdf_begin_text(), cpdf_end_text().

10.9.17 cpdf_show_xy

void cpdf_show_xy (int **pdf document**, string **text**, double **x-koor**, double **y-koor**, 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().

10.9.18 cpdf_text

void cpdf_text (int **pdf document**, string **text**, double **x-koor**, double **y-koor**, 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().

10.9.19 cpdf_set_font

void cpdf_set_font (int **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.

10.9.20 cpdf_set_leading

void cpdf_set_leading (int **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().

10.9.21 cpdf_set_text_rendering

void cpdf_set_text_rendering (int **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.

10.9.22 cpdf_set_horiz_scaling

void cpdf_set_horiz_scaling (int **pdf document**, double **scale**)
cpdf_set_horiz_scaling() fixe l'échelle horizontale du texte à **scale** %.

10.9.23 cpdf_set_text_rise

void cpdf_set_text_rise (int **pdf document**, double **value**)
cpdf_set_text_rise() fixe l'élévation du texte à **value** unités.

10.9.24 cpdf_set_text_matrix

void cpdf_set_text_matrix (int **pdf document**, array **matrix**)
cpdf_set_text_matrix() fixe la matrice du texte, qui décrit la transformation appliquée à police.

10.9.25 cpdf_set_text_pos

void cpdf_set_text_pos (int **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().

10.9.26 cpdf_set_char_spacing

void cpdf_set_char_spacing (int **pdf document**, double **space**)
cpdf_set_char_spacing() fixe l'espacement des caractères.
Voir aussi cpdf_set_word_spacing(), cpdf_set_leading().

10.9.27 cpdf_set_word_spacing

void cpdf_set_word_spacing (int **pdf document**, double **space**)
cpdf_set_word_spacing() fixe l'espacement des caractères.
Voir aussi cpdf_set_char_spacing(), cpdf_set_leading().

10.9.28 cpdf_continue_text

void cpdf_continue_text (int **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().

10.9.29 cpdf_stringwidth

double cpdf_stringwidth (int **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().

10.9.30 cpdf_save

void cpdf_save (int **pdf document**)
cpdf_save() sauve l'environnement courant. Cette fonction 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().

10.9.31 cpdf_restore

void cpdf_restore (int **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.

```
<?php cpdf_save($pdf);
// plein de transformations, translations, ...
cpdf_restore($pdf) ?>
```

Voir aussi cpdf_save().

10.9.32 cpdf_translate

void cpdf_translate (int **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. Si il prend la valeur de 0 (ou si il est omis), c'est la valeur par défaut (72) qui est utilisé.

10.9.33 cpdf_scale

void cpdf_scale (int **pdf document**, double **x-scale**, double **y-scale**)
cpdf_scale() modifie l'echelle dans les deux direction.

10.9.34 cpdf_rotate

void cpdf_rotate (int **pdf document**, double **angle**)
cpdf_rotate() effectue une rotation, d'un angle de **angle** degrés.

10.9.35 cpdf_setflat

void cpdf_setflat (int **pdf document**, double **value**)
cpdf_setflat() fixe la platitude (flatness), entre 0 et 100.

10.9.36 cpdf_setlinejoin

void cpdf_setlinejoin (int **pdf document**, long **value**)
cpdf_setlinejoin() fixe le paramètre linejoin à une valeur entre 0 et 2. 0 = miter, 1 = round, 2 = bevel.

10.9.37 cpdf_setlinecap

void cpdf_setlinecap (int **pdf document**, int **value**)
cpdf_setlinecap() fixe le paramètre linecap à une valeur entre 0 et 2. 0 = butt end, 1 = round, 2 = projecting square.

10.9.38 cpdf_setmiterlimit

void cpdf_setmiterlimit (int **pdf document**, double **value**)
cpdf_setmiterlimit() Fixe le paramètre miter limit à une valeur supérieure ou égale à 1.

10.9.39 cpdf_setlinewidth

void cpdf_setlinewidth (int **pdf document**, double **width**)
cpdf_setlinewidth()Fixe la largeur de ligne à la valeur de **width**.

10.9.40 cpdf_setdash

void cpdf_setdash (int **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.

10.9.41 cpdf_moveto

void cpdf_moveto (int **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.

10.9.42 cpdf_rmoveto

void cpdf_rmoveto (int **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().

10.9.43 cpdf_curveto

void cpdf_curveto (int **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().

10.9.44 cpdf_lineto

void cpdf_lineto (int **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 utilisé.
Voir aussi cpdf_moveto(), cpdf_rmoveto(), cpdf_curveto().

10.9.45 cpdf_rlineto

void cpdf_rlineto (int **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 utilisé.
Voir aussi cpdf_moveto(), cpdf_rmoveto(), cpdf_curveto().

10.9.46 cpdf_circle

void cpdf_circle (int **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 utilisé.
Voir aussi cpdf_arc().

10.9.47 cpdf_arc

void cpdf_arc (int **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 utilisé.
Voir aussi cpdf_circle().

10.9.48 cpdf_rect

void cpdf_rect (int **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 utilisé.

10.9.49 cpdf_closepath

void cpdf_closepath (int **pdf document**)
cpdf_closepath() ferme le chemin courant.

10.9.50 cpdf_stroke

void cpdf_stroke (int **pdf document**)
cpdf_stroke() dessine une ligne le long du chemin.
Voir aussi cpdf_closepath(), cpdf_closepath_stroke().

10.9.51 cpdf_closepath_stroke

void cpdf_closepath_stroke (int **pdf document**)
cpdf_closepath_stroke() est une combinaison de cpdf_closepath() et cpdf_stroke().
Voir aussi cpdf_closepath(), cpdf_stroke().

10.9.52 cpdf_fill

void cpdf_fill (int **pdf document**)
cpdf_fill() rempli l'intérieur du chemin courant avec la couleur courante.
Voir aussi cpdf_closepath(), cpdf_stroke(), cpdf_setgray_fill(), cpdf_setgray(), cpdf_setrgbcolor_fill(), cpdf_setrgbcolor().

10.9.53 cpdf_fill_stroke

void cpdf_fill_stroke (int **pdf document**)
cpdf_fill_stroke() remplis 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(), cpdf_setrgbcolor().

10.9.54 cpdf_closepath_fill_stroke

void cpdf_closepath_fill_stroke (int **pdf document**)
cpdf_closepath_fill_stroke() remplis 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(), cpdf_setrgbcolor().

10.9.55 cpdf_clip

void cpdf_clip (int **pdf document**)
cpdf_clip() aligne les dessins sur le chemin courant.

10.9.56 cpdf_setgray_fill

void cpdf_setgray_fill (int **pdf document**, double **value**)
cpdf_setgray_fill() Choisit un niveau de gris comme couleur de remplissage.
Voir aussi cpdf_setrgbcolor_fill().

10.9.57 cpdf_setgray_stroke

void cpdf_setgray_stroke (int **pdf document**, double **gray value**)
cpdf_setgray_stroke() choisit un niveau de gris comme couleur de dessin.
Voir aussi cpdf_setrgbcolor_stroke().

10.9.58 cpdf_setgray

void cpdf_setgray (int **pdf document**, double **gray value**)

cpdf_setgray_stroke() choisit un niveau de gris comme couleur de dessin et de remplissage.
Voir aussi cpdf_setrgbcolor_stroke(), cpdf_setrgbcolor_fill().

10.9.59 cpdf_setrgbcolor_fill

void cpdf_setrgbcolor_fill (int **pdf document**, double **red value**, double **green value**, double **blue value**)
cpdf_setrgbcolor_fill() choisit une couleur rgb comme couleur de remplissage.
Voir aussi cpdf_setrgbcolor_stroke(), cpdf_setrgbcolor().

10.9.60 cpdf_setrgbcolor_stroke

void cpdf_setrgbcolor_stroke (int **pdf document**, double **red value**, double **green value**, double **blue value**)
cpdf_setrgbcolor_stroke() choisit une couleur rgb comme couleur de dessin.
Voir aussi cpdf_setrgbcolor_fill(), cpdf_setrgbcolor().

10.9.61 cpdf_setrgbcolor

void cpdf_setrgbcolor (int **pdf document**, double **red value**, double **green value**, double **blue value**)
cpdf_setrgbcolor_stroke() choisit une couleur rgb comme couleur de dessin et de remplissage.
Voir aussi cpdf_setrgbcolor_stroke(), cpdf_setrgbcolor_fill().

10.9.62 cpdf_add_outline

void cpdf_add_outline (int **pdf document**, string **text**)
cpdf_add_outline() ajoute un signet à la page courante, avec le texte **text** qui pointe sur la page courante.

```
<?php
$cpdf = cpdf_open(0);
cpdf_page_init($cpdf, 1, 0, 595, 842);
cpdf_add_outline($cpdf, 0, 0, 0, 1, "Page 1");
// ...
// quelques dessin
// ...
cpdf_finalize($cpdf);
Header("Content-type: application/pdf");
cpdf_output_buffer($cpdf);
cpdf_close($cpdf);
?>
```

10.9.63 cpdf_set_page_animation

void cpdf_set_page_animation (int **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.

10.9.64 cpdf_import_jpeg

int cpdf_import_jpeg (int **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().

10.9.65 cpdf_place_inline_image

void cpdf_place_inline_image (int **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().

10.9.66 cpdf_add_annotation

void cpdf_add_annotation (int **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.

10.10 CURL

PHP supporte libcurl, une librairie créée par Daniel Stenberg, qui vous permet de vosu connecter de commniquer avec de nombreux serveurs, gr‰ce 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'autentification par mot de passe et nom de compte.

Pour pouvoir utiliser les fonctions CURL, vous devez installer le package CURL. 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] o□ DIR est le chemin jusqu'au dossier contenant les dossier `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 :

```
<?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);
?>
```

10.10.1 curl_init

int 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().

```
<?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().

10.10.2 curl_setopt

bool curl_setopt (int **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 : (specifiée 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**: Choississez une valeur non nulle pour que CURL vous affiche tous les événements.
- **CURLOPT_HEADER**: Choississez une valeur non nulle pour que CURL inclus l'entête dans la valeur de retour.
- **CURLOPT_NOPROGRESS**: Choississez 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**: Choississez une valeur non nulle pour que le corps du transfert ne soit pas inclus dans la valeur de retour.
- **CURLOPT_FAILONERROR**: Choississez 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**: Choississez une valeur non nulle pour que PHP prépare un chargement.
- **CURLOPT_POST**: Choississez 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**: Choississez une valeur non nulle pour que PHP ne fasse que lister les noms d'un dossier FTP.
- **CURLOPT_FTPAPPEND**: Choississez une valeur non nulle pour que PHP concatène le fichier distant, plutôt que de l'écraser.

- **CURLOPT_NETRC**: Choississez 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**: Choississez une valeur non nulle pour suivre toutes les entêtes "Location: " que le serveur envoie dans les entêtes HTTP (notez que cette fonction est récursive, et que PHP suivra toutes les entêtes "Location: " qu'il trouvera).

- **CURLOPT_PUT**: Choississez 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**: Choississez 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'annuelera.

- **CURLOPT_RESUME_FROM**: Passez un a long as a parameter that contains the offset, in bytes, that you want the transfer to start from.

- **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 cha”ne de caractères pour les valeurs suivantes de **option** parameter:

- **CURLOPT_URL**: L'URL que PHP va récupérer. Vous pouvez aussi choisir cette v aleur lors de l'appel à curl_init(). function.

- **CURLOPT_USERPWD**: Passez une cha”ne de caractères au format [nom]:[mot de passe], pour que PHP l'utilise lors de la connexion.

- **CURLOPT_PROXYUSERPWD**: Passez une cha”ne 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 cha”ne de caractères qui représente la plage de valeur que vous désirez. Elle est au format "X-Y", o☐ 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 cha”ne de caractères qui contient toutes les données à passer lors d'une opération de HTTP POST.

- **CURLOPT_REFERER**: Passez une cha”ne de caractères qui contient l'entête de "REFERER", utilisé lors d'une requête HTTP.

- **CURLOPT_USERAGENT**: Passez une cha”ne de caractères qui contient l'entête "user-agent" utilisé dans une requête HTTP.

- **CURLOPT_FTPPORT**: Passez une cha”ne 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 cha”ne 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 cha”ne de caractères qui contiendra le contenu du cookie, à transmettre dans l'entête HTTP.
- **CURLOPT_SSLCERT**: Passez une cha”ne de caractères qui contiendra le nom de fichier du certificat, au format PEM.
- **CURLOPT_SSLCERTPASSWD**: Passez une cha”ne de caractères qui contient le mot de passe nécessaire pour utiliser le certificat CURLOPT_SSLCERT.
- **CURLOPT_COOKIEFILE**: Passez une cha”ne 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 entêtes HTTP écrites dans un fichier.
- **CURLOPT_CUSTOMREQUEST**: Passez une cha”ne 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 assurrer 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'entête de la sortie du transfert.
- **CURLOPT_STDERR**: Le fichier d'erreurs.

10.10.3 curl_exec

bool curl_exec (int **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**.

10.10.4 curl_close

void curl_close (int **ch**)
Cette fonction ferme une session CURL et libère toutes les ressources reservées. L'identifiant CURL, **ch**, est aussi effacé.

10.10.5 curl_version

string curl_version
curl_version() retourne une cha”ne avec la version courante de la librairie CURL.

10.11 de 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 PHP4.

10.11.1 cybercash_encr

array cybercash_encr (string **wmk**, string **sk**, string **inbuff**)
Cette fonction retourne un tableau associatif, contenant les éléments "errcode" et, si "errcode" vaut FALSE, "outbuff" (string), "outLth" (long) et "macbuff" (string).

10.11.2 cybercash_decr

array cybercash_decr (string **wmk**, string **sk**, string **inbuff**)
Cette fonction retourne un tableau associatif, contenant les éléments "errcode" et, si "errcode" vaut FALSE, "outbuff" (string), "outLth" (long) et "macbuff" (string).

10.11.3 cybercash_base64_encode

string cybercash_base64_encode (string **inbuff**)

10.11.4 cybercash_base64_decode

string cybercash_base64_decode (string **inbuff**)

10.12 de dates et heures

10.12.1 checkdate

int checkdate (int **month**, int **day**, int **year**)
Retourne TRUE si la date fournie est valide, et sinon FALSE. La date est considérée comme valide si :

- L'année est comprise entre entre 0 et 32767 inclus
- Le mois est compris entre 1 et 12 inclus
- Le jour est compris dans l'intervalle de date du mois. Les années bissextiles sont prises en compte.

10.12.2 date

string date (string **format**, int **timestamp**)
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)
- d - Jour du mois, sur deux chiffres (éventuellement avec un zéros) : "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)
- h - Heure, au format 12h, "01" à "12"
- H - heure, au format 24h, "00" à "23"
- g - Heure, au format 12h sans les zéros initiaux, "1" à "12"
- G - Heure, au format 24h sans les zéros initiaux, "0" à "23"
- i - Minutes; "00" à "59"
- 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 bisextile ("1") ou pas ("0")
- m - - Mois; i.e. "01" à "12"
- n - Mois sans les zéros initiaux; i.e. "1" à "12"
- M - Mois, en trois lettres (et en anglais) : par exemple "Jan" (pour Janvier)
- 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 jour dans le mois donné, i.e. "28" à "31"
- 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().

```
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 futur ou dans passé.

```
$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);
```

Pour formater des dates dans d'autres langues, utilisez les fonctions setlocale() et strftime().
Voir aussi gmdate() et mktime().

10.12.3 getdate

array getdate (int **timestamp**)
Retourne un tableau associatif contenant les informations de date et heure du timestamp, avec les champs suivants :

- "seconds" - secondes
- "minutes" - minutes
- "hours" - heures
- "mday" - jour du mois
- "wday" - jour de la semaine, numérique
- "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"

10.12.4 gettimeofday

array gettimeofday (void)
C'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

10.12.5 gmdate

string gmdate (string **format**, int **timestamp**)
Identique à la fonction date(), seulement 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".

```
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().

10.12.6 gmmktime

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.

10.12.7 gmstrftime

string gmstrftime (string **format**, int **timestamp**)
gmstrftime() se comporte exactement comme strftime() hormis le fait 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".

```
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().

10.12.8 microtime

string 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 gettimeofday().
Voir aussi time().

10.12.9 mktime

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.
Cette fonction 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".

```
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 oú 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 and 2037.
Le dernier jours 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".

```
$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().

10.12.10 strftime

string strftime (string **format**, int **timestamp**)
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 suivant 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. locale
- %d - jour du mois en numérique (intervalle 00 à 31)
- %H - heure de la journée en numérique, et sur 24-heures (intervalle 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
- %p - soit `am' ou `pm' en fonction de l'heure absolue, ou en fonction des valeurs enregistrées en local.
- %S - secondes en numérique
- %U - numéro de semaine dans l'année, en considérant le premier dimanche de l'année comme le premier jour de la première semaine.
- %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

```
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() et mktime().

10.12.11 time

int time
Retourne la heure courante, mesurée en secondes depuis le début de l'époque UNIX, (1er janvier 1970 00:00:00 GMT).
Voir aussi date().

10.13 dbm

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. (A ne pas confondre avec IBM's DB2 software, qui fonctionne avec 10.62 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.
Les bases suivantes sont supportées :

- 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.
- ndbm est un nouveau type de dbm plus flexible. Il a cependant la majorité des limitations du genre.
- gdbm est la base dbm GNU.
- db2 est DB2 de Sleepycat Software. 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".
- cdb rdy "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. Puisque c'est une base constante, elle ne supporte que la lecture.

```
<?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.

```
<?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);
?>
```

10.13.1 dba_close

void dba_close (int **handle**)
dba_close() ferme le lien établit 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().

10.13.2 dba_delete

string dba_delete (string **key**, int **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()

10.13.3 dba_exists

bool dba_exists (string **key**, int **handle**)
dba_exists() vérifie si la clé **key** existe dans la base identifiée par **handle**.
key est la clé qui doit être vérfiée.
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().

10.13.4 dba_fetch

string dba_fetch (string **key**, int **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().

10.13.5 dba_firstkey

string dba_firstkey (int **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().

10.13.6 dba_insert

bool dba_insert (string **key**, string **value**, int **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().

10.13.7 dba_nextkey

string dba_nextkey (int **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().

10.13.8 dba_popen

int dba_popen (string **path**, string **mode**, string **handler**, **...**)
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().

10.13.9 dba_open

int dba_open (string **path**, string **mode**, string **handler**, **...**)
dba_open() établit une connexion à la base repérée par **path** avec le mode **mode** et l'identifiant **handler**.
pathest 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().

10.13.10 dba_optimize

bool dba_optimize (int **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().

10.13.11 dba_replace

bool dba_replace (string **key**, string **value**, int **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().

10.13.12 dba_sync

bool dba_sync (int **handle**)
dba_sync() synchronise la base de données spécifiée par **handle**. Si accepté, cela va probablement lancer une opératio nde 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().

10.14 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 la même fichier dBase risque de rendre votre base de données incohérente.
A la différence des bases de données SQL, la définition des bases de données dBase, ne peut pas être changée. Une fois le fichier créé, la définition de la base est définitive. Il n'y a pas d'index qui accélèrent les recherches ou organisent vos données. Les fichiers dBase sont de simples fichiers séquentiels avec des 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 and Postgres 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 de et vers votre base des données web, maintenant que le format du fichier est communément géré 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.

10.14.1 dbase_create

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.

```
// "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>";
```

10.14.2 dbase_open

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).
Retourne un identifiant de base de données, ou FALSE si la base n'a pas pu être selectionnée.

10.14.3 dbase_close

bool dbase_close (int **dbase_identifier**)
Ferme la base associée à **dbase_identifier**.

10.14.4 dbase_pack

bool dbase_pack (int **dbase_identifier**)
Compacte la base de données spécifiée (effacement définitif de tous les enregistrements marqués pour l'effacement, avec la fonction dbase_delete_record()).

10.14.5 dbase_add_record

bool dbase_add_record (int **dbase_identifier**, array **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.

10.14.6 dbase_replace_record

bool dbase_replace_record (int **dbase_identifier**, array **record**, int **dbase_record_number**)
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()).

10.14.7 dbase_delete_record

bool dbase_delete_record (int **dbase_identifier**, int **record**)

Marque l'enregistrement **record** pour l'effacement, dans la base. Pour effacer réellement l'enregistrement, il faut utiliser aussi dbase_pack().

10.14.8 dbase_get_record

array dbase_get_record (int **dbase_identifier**, int **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).

10.14.9 dbase_get_record_with_names

array dbase_get_record_with_names (int **dbase_identifier**, int **record**)
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).

10.14.10 dbase_numfields

int dbase_numfields (int **dbase_identifier**)
Retourne le nombre de champs (colonnes) de la base de données spécifiée. 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).

```
$rec = dbase_get_record($db, $recno);
$nf  = dbase_numfields($db);
for ($i=0; $i < $nf; $i++) {
print $rec[$i]."<br>\n";
}
```

10.14.11 dbase_numrecords

int dbase_numrecords (int **dbase_identifier**)
Retourne le nombre d'enregistrements (lignes) dans la base spécifiée. 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).

10.15 dbm

Ces fonctions vous permettent d'écrire des lignes dans une base de donnée 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).

```
$dbm = dbmopen("dernier", "w");
if (dbmexists($dbm, $userid)) {
  $last_seen = dbmfetch($dbm, $userid);
} else {
dbminsert($dbm, $userid, time());
}
faire_quelquechose();
dbmreplace($dbm, $userid, time());
dbmclose($dbm);
```

10.15.1 dbmopen

int 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.
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 librairie 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 fait 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'.

10.15.2 dbmclose

bool dbmclose (int **dbm_identifier**)
Déverrouille et ferme la base de données dbm_identifier.

10.15.3 dbmexists

bool dbmexists (int **dbm_identifier**, string **key**)
Retourne TRUE si il y a une valeur associée à la clé **key**.

10.15.4 dbmfetch

string dbmfetch (int **dbm_identifier**, string **key**)
Retourne la valeur associée à la clé **key**.

10.15.5 dbminsert

int dbminsert (int **dbm_identifier**, string **key**, string **value**)
Ajoute la valeur **value** dans la base de données, avec la clé **key**.
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().)

10.15.6 dbmreplace

bool dbmreplace (int **dbm_identifier**, string **key**, string **value**)
Remplace une valeur courante par la valeur **value** pour la clé **key**, dans une base dbm.
Cette fonction va créer la clé, si elle n'existe pas dans la base.

10.15.7 dbmdelete

bool dbmdelete (int **dbm_identifier**, string **key**)
Efface la valeur de la clé **key**, dans la base dbm.
Retourne FALSE si la clé n'existe pas dans cette base.

10.15.8 dbmfirstkey

string dbmfirstkey (int **dbm_identifier**)
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 une table de hash.

10.15.9 dbmnextkey

string dbmnextkey (int **dbm_identifier**, string **key**)
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 :

```
$key = dbmfirstkey($dbm_id);
while ($key) {
echo "$key = " . dbmfetch($dbm_id, $key) . "\n";
   $key = dbmnextkey($dbm_id, $key);
}
```

10.15.10 dblist

string dblist (void)

10.16 Accès aux dossiers

10.16.1 chdir

int chdir (string **directory**)
Change le dossier courant de PHP en **directory**. Retourne FALSE si l'opération échoue, et TRUE sinon.

10.16.2 dir

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.

```
$d = dir("/etc");
echo "Pointeur: ".$d->handle."<br>\n";
echo "Chemin: ".$d->path."<br>\n";
while($entry=$d->read()) {
echo $entry."<br>\n";
}
$d->close();
```

10.16.3 closedir

void closedir (int **dir_handle**)
closedir() ferme le pointeur de dossier **dir_handle**. Le dossier devait avoir été ouvert avec opendir().

10.16.4 opendir

int opendir (string **path**)
opendir() retourne un pointeur sur un dossier pour être utilisé avec les fonctions closedir(), readdir() et rewinddir().

10.16.5 readdir

string readdir (int **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.

```
<?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 :

```
<?php
$handle=opendir('.');
while ($file = readdir($handle)) {
if ($file != "." && $file != "..") {
```

```
echo "$file\n";
   }
}
closedir($handle);
?>
```

10.16.6 rewinddir

void rewinddir (int **dir_handle**)
rewinddir() retourne à la première entrée du dossier : le prochain fichier lu sera le premier.

10.17 Chargement dynamique de fonctions

10.17.1 dl

int dl (string **library**)
Charge l'extension PHP définie dans la bibliothèque **library**. Voir aussi la directive de configuration 7.1.5.2 ini.extension-dir.

10.18 DOM XML

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-2.0.0 (la version beta ne fonctionne pas). Ces fonctions ont été ajoutées dans PHP4.
Note du traducteur : liste des fonctions à venir (13/6/2000) domxml_root domxml_add_root domxml_dumpmem domxml_attributes domxml_getattr domxml_setattr domxml_children domxml_new_child domxml_node domxml_new_xmldoc alias new_xmldoc.
Ce module définit les constantes suivantes :

Constant	Valeur	Déscription	
XML_ELEMENT_NODE	1	@tab	
XML_ATTRIBUTE_NODE	2	@tab	
XML_TEXT_NODE	3	@tab	
XML_CDATA_SECTION_NODE	4	@tab	
XML_ENTITY_REF_NODE	5	@tab	
XML_ENTITY_NODE	6	@tab	
XML_PI_NODE	7	@tab	
XML_COMMENT_NODE	8	@tab	
XML_DOCUMENT_NODE	9	@tab	
XML_DOCUMENT_TYPE_NODE	10	@tab	
XML_DOCUMENT_FRAG_NODE	11	@tab	
XML_NOTATION_NODE	12	@tab	
XML_GLOBAL_NAMESPACE	1	@tab	
XML_LOCAL_NAMESPACE	2	@tab	

Ce module définit un ensemble de classes. Les fonctions DOM XML retourne un arbre à partir d'un document XML, dont chaque noeud est un objet issu de ces classes.

10.18.1 xmldoc

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).

10.18.2 xmldocfile

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).

10.18.3 xmltree

object xmltree (string **str**)
xmltree() analyse le document XML **str** et retourne un arbre d'objets PHP qui représente le document analysé.

10.19 Fonction d'exécution de programmes

10.19.1 escapeshellcmd

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 9.7.6 Opérateur d'exécutions. Généralement, cette fonction est utilisée comme ceci :
system(EscapeShellCmd($cmd))

Voir aussi exec(), popen(), system(), et les 9.7.6 Opérateur d'exécutions.

10.19.2 exec

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 9.7.6 Opérateur d'exécutions.

10.19.3 passthru

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 9.7.6 Opérateur d'exécutions.

10.19.4 system

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.
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 dans aucune intervention, utilisez la fonction passthru().
Voir aussi exec(), PassThru(), popen(), EscapeShellCmd() et les 9.7.6 Opérateur d'exécutions.

10.20 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 : *Actuellement, Adobe ne fournit que libc5, compatible Linux. Les tests avec glibc2 ont donné des "segmentation fault". Si quelqu'un est arrivé à un résultat, envoyez nous vos infos.*
Note : *Si vous rencontrez des problèmes de configuration avec fdftk, vérifier que le header FdfTk.h et la librairie libFdfTk.so sont bien là oú ils doivent être. Ils devraient être placés dans le dossier `fdftk-dir/include' et `fdftk-dir/lib'. Cela ne sera pas le cas si vous avez simplement décompressé la distribution.*
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 plutot 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.
Les exemples suivants montre comme évaluer les 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.
// 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);
?>
```

10.20.1 fdf_open

int fdf_open (string **filename**)

La fonction 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.

```
<?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().

10.20.2 fdf_close

void fdf_close (int **fdf_document**)
fdf_close() ferme le document FDF.
Voir aussi fdf_open().

10.20.3 fdf_create

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.

```
<?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().

10.20.4 fdf_save

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().

10.20.5 fdf_get_value

string fdf_get_value (int **fdf_document**, string **fieldname**)

fdf_get_value() retourne la valeur d'un champs.
Voir aussi fdf_set_value().

10.20.6 fdf_set_value

void fdf_set_value (int **fdf_document**, string **fieldname**, string **value**, int **isName**)
fdf_set_value() fixe la valeur d'un champs. Le dernier paramètre determine 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().

10.20.7 fdf_next_field_name

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().

10.20.8 fdf_set_ap

void 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.

10.20.9 fdf_set_status

void fdf_set_status (int **fdf_document**, string **status**)
fdf_set_status() fixe la valeur de la clé /STATUS.
Voir aussi fdf_get_status().

10.20.10 fdf_get_status

string fdf_get_status (int **fdf_document**)
fdf_get_status() retourne la valeur de la clé /STATUS.
Voir aussi fdf_set_status().

10.20.11 fdf_set_file

void 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().

10.20.12 fdf_get_file

string fdf_get_file (int **fdf_document**)
fdf_set_file() lit la valeur de la clé /F.
Voir aussi fdf_set_file().

10.21 filePro

Ces fonctions permettent de lire des données enregistrées dans des bases non modifiables, sur des serveurs filePro.
filePro est une marque de Fiserv, Inc. Vous pouvez avoir plus de détails sur filePro à http://www.fileproplus.com/.

10.21.1 filepro

bool filepro (string **directory**)
Cette fonction 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.

10.21.2 filepro_fieldname

string filepro_fieldname (int **field_number**)

Retourne le nom du champs d'index **field_number**.

10.21.3 filepro_fieldtype

string filepro_fieldtype (int **field_number**)
Retourne le type du champs d'index **field_number**.

10.21.4 filepro_fieldwidth

int filepro_fieldwidth (int **field_number**)
Retourne la taille du champs d'index **field_number**.

10.21.5 filepro_retrieve

string filepro_retrieve (int **row_number**, int **field_number**)
Retourne la valeur du champs d'index **row_number**, et à la ligne **field_number**.

10.21.6 filepro_fieldcount

int filepro_fieldcount
Retourne le nombre de champs (ou colonnes) d'une base filePro.
Voir aussi filepro().

10.21.7 filepro_rowcount

int filepro_rowcount
Retourne le nombre de lignes dans une base filePro.
Voir aussi filepro().

10.22 Système de fichiers

10.22.1 basename

string basename (string **path**)
Cette fonction prend en paramètre le chemin complet d'un fichier et en extrait le nom du fichier.
Sous Windows, les caractères (/) et backslash (\) sont utilisés comme séparateur de dossier. Sous les autres OS, seul le caractère slash (/) est utilisé.

```
$path = "/home/httpd/html/index.php3";
$file = basename($path); // $file est affecté avec "index.php3"
```

Voir aussi: dirname().

10.22.2 chgrp

int chgrp (string **filename**, mixed **group**)
chgrp() essaie de changer le groupe propriétaire du fichier. Seul le superuser (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.
Renvoie TRUE en cas de succès, sinon renvoie FALSE.
Voir aussi chown() et chmod().
Note : *Sous Windows, cette fonction ne fait rien et renvoie TRUE dans tous les cas.*

10.22.3 chmod

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**):

```
chmod( "/somedir/somefile", 755 );   // notation décimale; probablement FALSE
chmod( "/somedir/somefile", 0755 );  // notation octale; valeur du mode correcte
```

Renvoie TRUE en cas de succès, FALSE sinon.
Voir aussi chown() et chgrp().
Note : *Cette fonction ne fonctionne pas sous Windows.*

10.22.4 chown

int chown (string **filename**, mixed **user**)
chown() change le groupe propriétaire du fichier. Seul le superutilisateur (root) peut changer le propriétaire d'un fichier.
Renvoie "TRUE" en cas de succès, "FALSE" sinon. Note : *Sous Windows, ne fait rien et retourne TRUE.*

Voir aussi chown() et chmod().
Note : *Cette fonction est inopérante sous Windows*

10.22.5 clearstatcache

void clearstatcache
L'appel à la fonction stat ou lstat est relativement couteux 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 oú 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_writeable(), is_readable(), is_executable(), is_file(), is_dir(), is_link(), filectime(), fileatime(), filemtime(), fileinode(), filegroup(), fileowner(), filesize(), filetype(), et fileperms().

10.22.6 copy

int copy (string **source**, string **dest**)
copy() fait une copie du fichier. Renvoie TRUE en cas de succès, FALSE sinon.

```
if ( !copy($file, $file.'.bak') ) {
print("La copie du fichier $file n'a pas réussi...<br>\n");
}
```

Voir aussi: rename().

10.22.7 delete

void delete (string **file**)
Ceci est une fausse entrée du manuelle pour ceux qui recherchent en fait la fonction unlink() ou unset().
Voir aussi: unlink() pour effacer des fichiers, unset() pour effacer des variables.

10.22.8 dirname

string dirname (string **path**)
Si **path** contient le chemin d'un fichier ou dossier, cette fonction retournera le nom du dossier qui le contient.
Sous Windows, les slash (/) et backslash (\) sont utilisés comme séparateurs de dossier. Dans les autres environnements, seul le slash (/) est utilisé.

```
$path = "/etc/passwd";
$file = dirname($path); // $file contient "/etc"
```

Voir aussi: basename().

10.22.9 diskfreespace

float diskfreespace (string **directory**)
diskfreespace() retournera le nombre d'octets disponible sur le disque correspondant contenant le dossier **directory**.

```
$df = diskfreespace("/"); // $df contient le nombre d'octets libres sur "/"
```

10.22.10 fclose

int fclose (int **fp**)
Ferme le fichier **fp**.
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().

10.22.11 feof

int feof (int **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().

10.22.12 fgetc

string fgetc (int **fp**)
fgetc() retourne une chaîne contenant un seul caractère, lu depuis le fichier pointé par **fp**. 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().

10.22.13 fgetcsv

array fgetcsv (int **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.

```
$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);
```

10.22.14 fgets

string fgets (int **fp**, int **length**)
fgets() retourne la chaîne lue jusqu'à la longueur **length** - 1 octets, ou bien la fin du fichier, ou encore un retour chariot (le premier des trois qui sera rencontré).

Si une erreur survient, 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 :

```
$fd = fopen ("/tmp/inputfile.txt", "r");
while (!feof($fd)) {
   $buffer = fgets($fd, 4096);
echo $buffer;
}
fclose ($fd);
```

Voir aussi fread(), fopen(), popen(), fgetc(), et fsockopen().

10.22.15 fgetss

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, PHP4B3.*

Voir aussi fgets(), fopen(), fsockopen(), popen(), et strip_tags().

10.22.16 file

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 et en la mettant à "1", si vous voulez rechercher aussi dans le dossier 7.1.1.13 ini.include-path.

```
<?php
// Lire une page web dans un tableau, et l'afficher
$fcontents = file( 'http://www.php.net' );
while ( list( $line_num, $line ) = each( $fcontents ) ) {
echo "<b>Line $line_num:</b> " . htmlspecialchars( $line ) . "<br>\n";
}
// lire une page web dans une chaîne
$fcontents = join( '', file( 'http://www.php.net' ) );
?>
```

Voir aussi readfile(), fopen() et popen().

10.22.17 file_exists

int file_exists (string **filename**)
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 cette fonction est mis en cache. Reportez vous à clearstatcache() pour plus de détails.

10.22.18 fileatime

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.
Le résultat de cette fonction est mis en cache. Reportez vous à clearstatcache() pour plus de détails.

10.22.19 filectime

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.
Le résultat de cette fonction est mis en cache. Reportez vous à clearstatcache() pour plus de détails.

10.22.20 filegroup

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.
Le résultat de cette fonction est mis en cache. Reportez vous à clearstatcache() pour plus de détails.
Note : *Cette fonction est inopérante sous Windows*

10.22.21 fileinode

int fileinode (string **filename**)
fileinode() renvoie le numéro d'inode du fichier **filename**, ou FALSE en cas d'erreur.
Le résultat de cette fonction est mis en cache. Reportez vous à clearstatcache() pour plus de détails.
Note : *Cette fonction est inopérante sous Windows*

10.22.22 filemtime

int filemtime (string **filename**)
filemtime() renvoie la date de dernière modification du fichier **filename**, ou FALSE en cas d'erreur.
Le résultat de cette fonction est mis en cache. Reportez vous à clearstatcache() pour plus de détails.

10.22.23 fileowner

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.
Le résultat de cette fonction est mis en cache. Reportez vous à clearstatcache() pour plus de détails.
Note : *Cette fonction est inopérante sous Windows*

10.22.24 fileperms

int fileperms (string **filename**)
fileperms() renvoie les permissions affectées au fichier **filename**, ou FALSE en cas d'erreur.
Le résultat de cette fonction est mis en cache. Reportez vous à clearstatcache() pour plus de détails.

10.22.25 filesize

int filesize (string **filename**)
filesize() renvoie la taille du fichier **filename**, ou FALSE en cas d'erreur.
Le résultat de cette fonction est mis en cache. Reportez vous à clearstatcache() pour plus de détails.

10.22.26 filetype

string filetype (string **filename**)
filetype() renvoie le type du fichier **filename**. Les réponses possibles sont : fifo, char, dir, block, link, file, et unknown.
Retourne FALSE en cas d'erreur.
Le résultat de cette fonction est mis en cache. Reportez vous à clearstatcache() pour plus de détails.

10.22.27 flock

bool 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 de verrous écriture / lecture simple, qui peut être utilisé sur n'importe quelle plateforme (Unix et Windows compris).
flock() retourne TRUE en cas de succès, et FALSE sinon. (le verrou n'a pas pu être obtenu).

10.22.28 fopen

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.
Attention, fopen() ne gère pas les redirections, 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, cette fonction 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 dans 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 au début du fichier et réduit la taille du fichier à 0. 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 inutile sous Unix). Si il n'est pas nécessaire, il sera ignoré.
Vous pouvez utiliser le troisième paramètre optionnel pour explorer le dossier 7.1.1.13 ini.include-path, en le mettant à 1.

```
$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 backslash utilisés dans le chemin du fichier, ou bien utilisez des slash.

```
$fp = fopen("c:\\data\\info.txt", "r");
```

Voir aussi fclose(), fsockopen() et popen().

10.22.29 fpassthru

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().

10.22.30 fputs

int fputs (int **fp**, string **str**, int **length**)
fputs() est un alias de fwrite(), et lui est identique en tout point. Notez que **length** est un paramètre optionnel, et si il n'est pas spécifié, toute la chaîne est écrite.

10.22.31 fread

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).

```
// Lit un fichier, et le place dans une chaîne
$filename = "/usr/local/something.txt";
$fd = fopen ($filename, "r");
$contents = fread ($fd, filesize ($filename));
fclose ($fd);
```

Voir aussi fwrite(), fopen(), fsockopen(), popen(), fgets(), fgetss(), file(), et fpassthru().

10.22.32 fseek

int fseek (int **fp**, int **offset**)
fseek() positionne le pointeur interne de fichier dans le fichier référencé par **fp** à l'offset **offset**. Equivalent à la fonction C fseek(fp, offset, SEEK_SET).
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.
Ne peut pas être utilisé sur les pointeurs retournés par fopen() si ils sont au format HTTP ou FTP.
Voir aussi ftell() et rewind().

10.22.33 ftell

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().

10.22.34 fwrite

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 7.1.1.17 ini.magic-quotes-runtime sera ignorée, et les slash seront conservés.
Voir aussi fread(), fopen(), fsockopen(), popen(), et fputs().

10.22.35 set_file_buffer

int fwrite (int **fp**, int **buffer**)
set_file_buffer() ajoute un buffer pour les opérations d'écriture du fichier **fp** avec un buffer de **buffer** octets. Si buffer est à 0 alors les opérations d'écriture ne seront pas bufferisées.
La fonction retourne 0 en cas de succès, ou EOF si ce n'est pas possible.
Notez que par défaut, la fonction fopen() utilise un buffer de 8ko.
Voir aussi fopen().

10.22.36 is_dir

bool is_dir (string **filename**)
Retourne TRUE si **filename** existe et est un dossier.
Le résultat de cette fonction est mis en cache. Voir la fonction clearstatcache() pour plus de détails.
Voir aussi is_file() et is_link().

10.22.37 is_executable

bool is_executable (string **filename**)
Retourne TRUE si **filename** existe et est exécutable.
Le résultat de cette fonction est mis en cache. Voir la fonction clearstatcache() pour plus de détails.
Voir aussi is_file() et is_link().

10.22.38 is_file

bool is_file (string **filename**)
Retourne TRUE si **filename** existe et est un fichier (et pas un dossier).
Le résultat de cette fonction est mis en cache. Voir la fonction clearstatcache() pour plus de détails.
Voir aussi is_dir() et is_link().

10.22.39 is_link

bool is_link (string **filename**)
Retourne TRUE si **filename** existe et est un lien symbolique.
Le résultat de cette fonction est mis en cache. Voir la fonction clearstatcache() pour plus de détails.
Voir aussi is_dir() et is_file().
Note : *Cette fonction est inopérante sous Windows*

10.22.40 is_readable

bool is_readable (string **filename**)
is_readable() retourne vrai 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 cette fonction est mis en cache. Voir la fonction clearstatcache() pour plus de détails.
Voir aussi is_writeable().

10.22.41 is_writeable

bool is_writeable (string **filename**)
is_writeable() 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 cette fonction est mis en cache. Voir la fonction clearstatcache() pour plus de détails.
Voir aussi is_readable().

10.22.42 link

int link (string **target**, string **link**)
Link() crée un lien.
Voir aussi symlink() pour créer des liens symboliques et et readlink() avec linkinfo().
Note : *Cette fonction est inopérante sous Windows*

10.22.43 linkinfo

int linkinfo (string **path**)
Linkinfo() renvoie les informations à propos d'un lien le champs st_dev de la structure d'information d' UNIX (comme en langage C). Cette fonction sert à vérifier si un lien (repéré par **path**) existe (en utilisant la même méthode que la macro S_ISLNK de stat.h). Retourne FALSE en cas d'erreur.
Voir aussi symlink(), link(), et readlink().
Note : *Cette fonction est inopérante sous Windows*

10.22.44 mkdir

int mkdir (string **pathname**, int **mode**)
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 :
mkdir ("/path/to/my/dir", 0700);

Retourne TRUE en cas de succès, et FALSE en cas d'échec.
Voir aussi rmdir().

10.22.45 pclose

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().
Retourne le statut final du processus exécuté.
Voir aussi popen().

10.22.46 popen

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 unidirectionel (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.
$fp = popen ("/bin/ls", "r");

Voir aussi pclose().

10.22.47 readfile

int readfile (string **filename**, int **use_include_path**)
readfile() lis le fichier **filename** et l'envoi à la sortie standard.
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.
ATTENTION : PHP ne gère pas les redirections, donc il faut penser à ajouter un "/" final pour les dossiers.
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 7.1.1.13 ini.include-path, en passant la valeur de 1.
Voir aussi fpassthru(), file(), fopen(), include(), require(), et virtual().

10.22.48 readlink

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 : *Cette fonction est inopérante sous Windows*

10.22.49 rename

int rename (string **oldname**, string **newname**)
rename() rente de renommer le fichier **oldname** en **newname**.
Retourne TRUE en cas de succès et FALSE sinon.

10.22.50 rewind

int rewind (int **fp**)
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().

10.22.51 rmdir

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, retourne FALSE.
Voir aussi mkdir().

10.22.52 stat

array stat (string **filename**)
stat() renvoie les informations à propos du fichier filename.
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 supporte le type st_blksize type¤les autres systèmes (i.e. Windows) retourne -1
Les résultats de cette fonction sont mis en cache . Reportez vous à la fonction clearstatcache() pour plus de détails.

10.22.53 lstat

array lstat (string **filename**)
Cette fonction est identique à stat() mais elle accepte aussi un lien symbolique comme argument.
Retourne un tableau avec les éléments suivants :

1. volume
2. inode
3. mode de protection du inode
4. nombre de liens
5. id de lÃutilisateur 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 supporte le type st_blksize type¤les autres systèmes (i.e. Windows) retourne -1
Les résultats de cette fonction sont mis en cache . Reportez vous à la fonction clearstatcache() pour plus de détails.

10.22.54 symlink

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 : *Cette fonction est inopérante sous Windows*

10.22.55 tempnam

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.
Le comportement de tempnam() dépend du système. Sous Windows, la variable d'environnement TMP sera prioritaire par rapport au paramètre **dir**; sous Linux la variable d'environnement TMPDIR aura la priorité, tandis que l'OS SVR4 utilisera toujours le paramètre **dir** si le dossier n'existe pas. Reportez vous à la documentation de votre système(tempnam(3)).
Retourne le nom du fichier temporaire, ou la chaîne NULL en cas d'échec.

```
$tmpfname = tempnam ("/tmp", "FOO");
```

10.22.56 touch

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 crée.
Retourne TRUE en cas de succès, et FALSE sinon.

```
if ( touch($FileName) ) {
print "$FileName modification time has been changed to todays date et time";
} else {
print "Sorry Could Not change modification time of $FileName";
```

}

10.22.57 umask

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.

10.22.58 unlink

int unlink (string **filename**)
unlink() efface **filename**. Identique à la fonction Unix C unlink().
Retourne FALSE en cas d'échec.
Voir aussi rmdir() pour supprimer des dossiers.
Note : *Cette fonction peut ne pas fonctionner sous Windows.*

10.23 FTP

FTP : File Transfer Protocol.
Les constantes suivantes sont définies dans le module FTP : FTP_ASCII, et FTP_BINARY.

10.23.1 ftp_connect

int ftp_connect (string **host**, int **port**)
Retourne un flot FTP en cas de succès, et FALSEsinon.
ftp_connect() ouvre une connexion FTP avec l'hôte **host**. Le paramètre **port** spécifie le port de connexion. Si il est omis, le port 21 sera utilisé.

10.23.2 ftp_login

int ftp_login (int **ftp_stream**, string **username**, string **password**)
Retourne TRUE en cas de succès, et FALSE sinon.
Authentifie le flot FTP.

10.23.3 ftp_pwd

int ftp_pwd (int **ftp_stream**)
Retourne le nom du dossier courant, ou FALSE en cas d'erreur.

10.23.4 ftp_cdup

int ftp_cdup (int **ftp_stream**)
Retourne TRUE en cas de succès, et FALSE sinon.
Change de dossier, et passe au dossier parent.

10.23.5 ftp_chdir

int ftp_chdir (int **ftp_stream**, string **directory**)
Retourne TRUE en cas de succès, et FALSE sinon.
Change le dossier courant en **directory**.

10.23.6 ftp_mkdir

string ftp_mkdir (int **ftp_stream**, string **directory**)
Retourne le nom du dossier ainsi créé en cas de succès, et FALSE sinon.
Crée le dossier nommé **directory**.

10.23.7 ftp_rmdir

int ftp_rmdir (int **ftp_stream**, string **directory**)
Retourne TRUE en cas de succès, et FALSE sinon.
Efface le dossier **directory**.

10.23.8 ftp_nlist

int ftp_nlist (int **ftp_stream**, string **directory**)
Retourne un tableau de nom de fichiers en cas de succès, et FALSE sinon.

10.23.9 ftp_rawlist

int ftp_rawlist (int **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 résutltat.

10.23.10 ftp_systype

int ftp_systype (int **ftp_stream**)
Retourne le type de serveur, ou FALSE en cas d'erreur.

10.23.11 ftp_pasv

int ftp_pasv (int **ftp_stream**, int **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.

10.23.12 ftp_get

int ftp_get (int **ftp_stream**, string **local_file**, string **remote_file**, int **mode**)
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.

10.23.13 ftp_fget

int ftp_fget (int **ftp_stream**, int **fp**, string **remote_file**, int **mode**)
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.

10.23.14 ftp_put

int ftp_put (int **ftp_stream**, string **remote_file**, string **local_file**, int **mode**)
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.

10.23.15 ftp_fput

int ftp_fput (int **ftp_stream**, string **remote_file**, int **fp**, int **mode**)
Retourne TRUE en cas de succès, et FALSEsinon.
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.

10.23.16 ftp_size

int ftp_size (int **ftp_stream**, string **remote_file**)
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 ficheir n'existe pas, la valeur -1 est retournée. Certains serveurs FTP ne supportent pas cette fonction.

10.23.17 ftp_mdtm

int ftp_mdtm (int **ftp_stream**, string **remote_file**)
Retourne un UNIX timestamp en cas de succès, et FALSEsinon.
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.

10.23.18 ftp_rename

int ftp_rename (int **ftp_stream**, string **from**, string **to**)
Retourne TRUE en cas de succès, et FALSE sinon.
ftp_rename() renomme le fichier **from** en **to**.

10.23.19 ftp_delete

int ftp_delete (int **ftp_stream**, string **path**)
Retourne TRUE en cas de succès, et FALSEsinon.
ftp_delete() efface le fichier **path** sur un serveur FTP.

10.23.20 ftp_site

int ftp_site (int **ftp_stream**, string **cmd**)
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.

10.23.21 ftp_quit

int ftp_quit (int **ftp_stream**)
ftp_connect() ferme la connexion **ftp_stream**.

10.24 Fonctions GNU Gettext

10.24.1 bindtextdomain

string bindtextdomain (string **domain**, string **directory**)
bindtextdomain() fixe le chemin du domaine **domain** à **directory**.

10.24.2 dcgettext

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.

10.24.3 dgettext

string dgettext (string **domain**, string **message**)
dgettext() remplace le domaine courant.

10.24.4 gettext

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.

```
<?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"));
?>
```

10.24.5 textdomain

int textdomain (string **library**)

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.

10.25 HTTP

Ces fonctions permettent de travailler sur les informations transmises au navigateur, via le protocole HTTP.

10.25.1 header

int header (string **string**)

La fonction Header() permet de spécifier un entête HTTP lors de l'envoi des fichiers HTML. Reportez vous à HTTP 1.1 Specification pour plus d'informations sur les entêtes HTTP. NB : 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 espace ou des lignes vides dans ce code qui produisent un début de sortie avant que header() n'ai été appelé.
Il y a cependant deux entêtes spéciaux. Le premier est "Location". Non seulement il renvoie un entête au client, mais en plus, il envoie un statut de redirectoin 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.

```
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 tous les entêtes qui commencent par "HTTP/" (la casse n'est pas importante). Par exemple, si vous avez votre page d'erreur 404 Apache qui pointent sur un script PHP, c'est une bonne idée que de vous assurez que le script PHP génére une erreur 404. La première chose à dans votre script est :

```
header("http/1.0 404 Not Found");
```

Les scripts PHP générent 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

```
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
```

10.25.2 setcookie

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 entêtes. Les cookies doivent passer avant tout autre entête (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 client. 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 prochaîne page.
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 :

```
setcookie("TestCookie","Test Value");
setcookie("TestCookie",$value,time()+3600);  /* expire in 1 hour */
setcookie("TestCookie",$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 :

```
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 cookie que votre tableau a d'éléments, mais lorsque les cookies seront reçus par PHP, les valeurs seront placées dans un tableau :

```
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.

10.26 Hyperwave

10.26.1 Introduction

Hyperwave a été developpé par IICM 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. 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 un 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 (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'objet 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.

10.26.2 Intégration avec Apache

L'extension Hyperwave est utilisée de manière optimale lorsqu 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, etcÖ

10.26.3 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.

10.26.4 hw_Array2Objrec

strin hw_array2objrec (array **object_array**)
Converti un **object_array** en un objet. Les attributs multiples tels que 'Title' en différentes langues seront traités correctement.
Voir aussi hw_objrec2array().

10.26.5 hw_Children

array hw_children (int **connection**, int **objectID**)
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.

10.26.6 hw_ChildrenObj

array hw_childrenobj (int **connection**, int **objectID**)
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

10.26.7 hw_Close

int hw_close (int **connection**)
Retourne false si la connexion n'est pas valide, et sinon, true. Ferme la connexion connection à un serveur Hyperwave.

10.26.8 hw_Connect

int hw_connect (string **host**, int **port**, string **username**, string **password**)
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().

10.26.9 hw_Cp

int hw_cp (int **connection**, array **object_id_array**, int **destination id**)
Copie les objet ayant les objects id 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().

10.26.10 hw_Deleteobject

int hw_deleteobject (int **connection**, int **object_to_delete**)
Efface l'objet dont l'identifiant est object_to_delete. Toutes les instances de l'objets seront effacées. Retourne TRUE si aucune erreur ne survient, et sinon, FALSE.
Voir aussi hw_mv().

10.26.11 hw_DocByAnchor

int hw_docbyanchor (int **connection**, int **anchorID**)
Retourne l'dentifiant d'objet de l'objet dans l'ancrage **anchorID**.

10.26.12 hw_DocByAnchorObj

string hw_docbyanchorobj (int **connection**, int **anchorID**)
Retourne les attributs du document qui correspond à **anchorID**.

10.26.13 hw_DocumentAttributes

string hw_documentattributes (int **hw_document**)
Retourne les attributs du document.
Voir aussi hw_DocumentBodyTag(), hw_DocumentSize().

10.26.14 hw_DocumentBodyTag

string hw_documentbodytag (int **hw_document**)
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().

10.26.15 hw_DocumentContent

string hw_documentcontent (int **hw_document**)
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().

10.26.16 hw_DocumentSetContent

string hw_documentsetcontent (int **hw_document**, string **content**)
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().

10.26.17 hw_DocumentSize

int hw_documentsize (int **hw_document**)
Retourne la taille du document en octets.
Voir aussi hw_DocumentBodyTag(), hw_DocumentAttributes().

10.26.18 hw_ErrorMsg

string hw_errormsg (int **connection**)
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.

10.26.19 hw_EditText

int hw_edittext (int **connection**, int **hw_document**)
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(), hw_GetText().

10.26.20 hw_Error

int hw_error (int **connection**)
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.

10.26.21 hw_Free_Document

int hw_free_document (int **hw_document**)
Détruit un document Hyperwave.

10.26.22 hw_GetParents

array hw_getparentsobj (int **connection**, int **objectID**)
Retourne un tableau indexé avec les identifiants des objets parents de **objectID**.

10.26.23 hw_GetParentsObj

array hw_getparentsobj (int **connection**, int **objectID**)
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**.

10.26.24 hw_GetChildColl

array hw_getchildcoll (int **connection**, int **objectID**)
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().

10.26.25 hw_GetChildCollObj

array hw_getchildcollobj (int **connection**, int **objectID**)
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().

10.26.26 hw_GetRemote

int hw_getremote (int **connection**, int **objectID**)
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, grâces 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 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().

10.26.27 hw_GetRemoteChildren

int hw_getremotechildren (int **connection**, string **object record**)
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().

10.26.28 hw_GetSrcByDestObj

array hw_getsrcbydestobj (int **connection**, int **objectID**)
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().

10.26.29 hw_GetObject

array hw_getobject (int **connection**, [int)array] **objectID**, string **query**|
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().

10.26.30 hw_GetAndLock

string hw_getandlock (int **connection**, int **objectID**)
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().

10.26.31 hw_GetText

int hw_gettext (int **connection**, int **objectID**, mixed **rootID/prefix**)
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.
Cette fonction n'est opérationnelle qu'avec des document 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().

10.26.32 hw_GetObjectByQuery

array hw_getobjectbyquery (int **connection**, string **query**, int **max_hits**)
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().

10.26.33 hw_GetObjectByQueryObj

array hw_getobjectbyqueryobj (int **connection**, string **query**, int **max_hits**)
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().

10.26.34 hw_GetObjectByQueryColl

array hw_getobjectbyquerycoll (int **connection**, int **objectID**, string **query**, int **max_hits**)
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().

10.26.35 hw_GetObjectByQueryCollObj

array hw_getobjectbyquerycollobj (int **connection**, int **objectID**, string **query**, int **max_hits**)
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().

10.26.36 hw_GetChildDocColl

array hw_getchilddoccoll (int **connection**, int **objectID**)
Retourne un tableau avec les id des documents fils d'une collection.
Voir aussi hw_GetChildColl().

10.26.37 hw_GetChildDocCollObj

array hw_getchilddoccollobj (int **connection**, int **objectID**)
Retourne un tableau contenant les attributs des documents fils du groupe **objectID**.
Voir aussi hw_ChildrenObj(), hw_GetChildCollObj().

10.26.38 hw_GetAnchors

array hw_getanchors (int **connection**, int **objectID**)
Retourne un tableau contenant les identifiants des ancrages du document **objectID**.

10.26.39 hw_GetAnchorsObj

array hw_getanchorsobj (int **connection**, int **objectID**)
Retourne un tableau d'attributs des ancrages du document **objectID**.

10.26.40 hw_Mv

int hw_mv (int **connection**, array **object id array**, int **source id**, int **destination id**)
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().

10.26.41 hw_Identify

int hw_identify (string **username**, string **password**)
Identifies un utilisateur, dont le nom d'utilisateur est **username** and 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().

10.26.42 hw_InCollections

array hw_incollections (int **connection**, array **object_id_array**, array **collection_id_array**, int **return_collections**)
Vérifie qu'un ensemble d'objet (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'identifiant 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.

10.26.43 hw_Info

string hw_info (int **connection**)
Retourne les informations de la connexion courante. La chaîne retournée a le format suivant :
<Serverstring>, <Host>, <Port>, <Username>, <Port of Client>, <Byte swapping>

10.26.44 hw_InsColl

int hw_inscoll (int **connection**, int **objectID**, array **object_array**)
Insère un nouveau groupe, avec les attributs **object_array** dans le groupe **objectID**.

10.26.45 hw_InsDoc

int hw_insdoc (int **connection**, int **parentID**, string **object_record**, string **text**)
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().

10.26.46 hw_InsertDocument

int hw_insertdocument (int **connection**, int **parent_id**, int **hw_document**)
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().

10.26.47 hw_InsertObject

int hw_insertobject (int **connection**, string **object rec**, string **parameter**)
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(), hw_InsColl().

10.26.48 hw_mapid

int hw_mapid (int **connection**, int **server id**, int **object id**)
Represente 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

10.26.49 hw_Modifyobject

int hw_modifyobject (int **connection**, int **object_to_change**, array **remove**, array **add**, int **mode**)
Cette commande permet d'effacer, d'ajouter ou de modifier des 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().

```
// $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.

```
// $connect is an existing connection to the Hyperwave server
// $objid is the ID of the object to modify
$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 :*

```
$remarr = array("Title" => "en:Books");
$addarr = array("Title" => "en:Articles");
$hw_modifyobject($connect, $objid, $remarr, $addarr);
```

ou

```
$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'.

```
$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 : *Seul 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.

10.26.50 hw_New_Document

int hw_new_document (string **object_record**, string **document_data**, int **document_size**)

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().

10.26.51 hw_Objrec2Array

array hw_objrec2array (string **object_record**)
Convertir 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().

10.26.52 hw_OutputDocument

int hw_outputdocument (int **hw_document**)
Affiche hw_document sans la balise BODY.

10.26.53 hw_pConnect

int hw_pconnect (string **host**, int **port**, string **username**, string **password**)
Retourne un index de connexion en cas de succès, et false si la connexion n'a pas pu être créée. 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 faites, (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().

10.26.54 hw_PipeDocument

int hw_pipedocument (int **connection**, int **objectID**)
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'attend pas la fin du transfert pour rendre la main).
Voir aussi hw_GetText() (insertions), hw_free_document(), hw_DocumentSize(), hw_DocumentBodyTag(), hw_OutputDocument().

10.26.55 hw_Root

int hw_root ()
Retour l' Object id de la racine. Actuellement, cette identifiant est toujours 0. L'ensemble des fils de la racine est celui du serveur courant.

10.26.56 hw_Unlock

int hw_unlock (int **connection**, int **objectID**)
Déverrouille un document, et laisse l'accès aux autres utilisateurs.
Voir aussi hw_GetAndLock().

10.26.57 hw_Who

int hw_who (int **connection**)
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 1 l'élément est l'utilisateur qui a appelé la fonction.

10.26.58 hw_Username

string hw_getusername (int **connection**)
Retourne le nom d'utilisateur utilisé par la connexion.

10.27 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!

10.27.1 ibase_connect

int ibase_connect (string **database**, string **username** , string **password**)
Ouvre une connexion à une base de données Interbase.

```
$dbh = ibase_connect ($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_pconnect().

10.27.2 ibase_pconnect

int ibase_connect (string **database**, string **username** , string **password**)
Ouvre une connexion persistante à une base de données Interbase.
Voir aussi ibase_connect().

10.27.3 ibase_close

int ibase_close (int **connection_id**)
Ferme une connexion à une base de données Interbase. Cette fonction prend comme argument un identifiant de connexion id retourné par ibase_connect().

10.27.4 ibase_query

int ibase_query (int **link_identifier** , string **query** , int **bind_args**)
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().

10.27.5 ibase_fetch_row

int ibase_fetch_row (**int result_identifier**)
Retourne la prochaîne ligne spécifiée dans le résultat obtenu de ibase_query().

10.27.6 ibase_fetch_object

int ibase_fetch_object (int **result_id**)
Lit une ligne dans une base Interbase et la place dans un pseudo objet. Cette fonction prend comme argument un identifiant de résultat obtenu de ibase_query() ou ibase_execute().

```
$dbh = ibase_connect ($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().

10.27.7 ibase_free_result

int ibase_free_result (int **result_identifier**)
Libère un résultat créé par ibase_query().

10.27.8 ibase_prepare

int ibase_prepare (int **link_identifier** , string **query**)
Prépare une requête pour lier les paramètres (avec ibase_bind()) et l'éxécuter (avec ibase_execute()).

10.27.9 ibase_bind

ibase_bind (**int query**)
Lie les paramètres avec une requête précédemment préparée avec ibase_prepare().
Note : *Ibase_bind() ne fonctionne actuellement pas avec PHP4*

10.27.10 ibase_execute

int ibase_execute (**int query**)
Execute une requête préparée (et éventuellement liée) par ibase_prepare() (et éventuellement ibase_bind()).

10.27.11 ibase_free_query

ibase_free_query (**int query**)
Libère la mémoire réservée par une requête préparée par ibase_prepare().

10.27.12 ibase_timefmt

ibase_timefmt (**string format**)
Fixe le format de date pour les prochaînes requêtes.
Note : *Ibase_timefmt() ne fonctionne pas sous PHP4.*

10.27.13 ibase_num_fields

int ibase_num_fields (int **result_id**)
Retourne le nombre de lignes dans un résultat.

```
$dbh = ibase_connect ($host, $username, $password);
$stmt = 'SELECT * FROM tblname';
$sth = ibase_query ($dbh, $stmt);
if ( ibase_num_rows($sth) > 0 ) {
while ($row = ibase_fetch_object ($sth)) {
print $row->email . "\n";
   }
} else {
die("Aucun résultat");
}
ibase_close ($dbh);
```

Note : *Ibase_timefmt() ne fonctionne pas encore sous PHP4.*

10.28 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.

10.28.1 icap_open

stream icap_open (string**calendar**, string**username**, string**password**, string**options**)
Retourn 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(????).

10.28.2 icap_close

int icap_close (int **icap_stream**, int **flags**)
Ferme le flot ICAP **icap_stream**.

10.28.3 icap_fetch_event

object icap_fetch_event (stream**icap_stream**, id**event id**, options**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

10.28.4 icap_list_events

array icap_list_events (stream **icap_stream**, datetime **begin_date**, datetime **end_date**)
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

10.28.5 icap_store_event

int icap_store_event (int **icap_stream**, 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.

10.28.6 icap_delete_event

int icap_delete_event (int **uid**)
icap_delete_event() efface l'événement d'identifiant **uid**.
Retourne TRUE.

10.28.7 icap_snooze

int icap_snooze (int **uid**)
icap_snooze() éteind l'alarme de l'événement identifié par l'UID **uid**.
Retourne TRUE.

10.28.8 icap_list_alarms

array icap_list_alarms (stream **icap_stream**, datetime **alarm_date**)
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

10.29 Informix

Le pilote d'accès à Informix pour Online (ODS) 7.x, SE 7.x et Universal Server (IUS) 9.x est implémenté 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 partiellment fini, de nouveau types sont disponibles, mais SLOB et CLOB sont toujours en développement.
Note : *Avant que vous ne lanciez le script "configure", assurez vous que la variable d'environnement "INFORMIXDIR" a été correctement paramétrée.*
Le script de configuration va détecter automatiquement les librairies disponibles, et inclure les dossiers si

vous lancer le script avec l'option "configure --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.
Note : *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 lancant 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).
Note : *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").
Note : *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.
Note : *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 celle qui contiennent la valeur NULL.*

10.29.1 ifx_connect

int ifx_connect (string **database** , string **userid** , string **password**)
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, si ils viennent à manquer, les valeurs par défaut seront prises dans le fichier 7.1 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().

```
$conn_id = ifx_pconnect ("mydb@ol_srv1", "imyself", "mypassword");
```

10.29.2 ifx_pconnect

int ifx_pconnect (string **database** , string **userid** , string **password**)
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 :
Cette fonction 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().

10.29.3 ifx_close

int ifx_close (int **link_identifier**)
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().

```
$conn_id = ifx_connect ("mydb@ol_srv", "itsme", "mypassword");
... some queries and stuff ...
ifx_close($conn_id);
```

10.29.4 ifx_query

int ifx_query (string **query**, int **link_identifier** , int **cursor_type** , mixed **BLOBidarray**)
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 par 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 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_connect().

```
ifx_textasvarchar(1);      // Utilisation du mode "text mode" pour les BLOBs
$res_id = ifx_query("select * from orders", $conn_id);
if (! $res_id) {
printf("Impossible de selectionner des lignes dans : %s\n<br>%s<br>\n", ifx_error());
ifx_errormsg();
die;
}
ifx_htmltbl_result($res_id, "border=\"1\"");
ifx_free_result($res_id);
```

```
                // créer un identifiant de BLOB pour une colonne de type BYTE et une de type TEXT
$textid = ifx_create_blob(0, 0, "Text column in memory");
$byteid = ifx_create_blob(1, 0, "Byte column in memory");
                // store BLOB id's in a BLOBid array
$BLOBidarray[] = $textid;
$BLOBidarray[] = $byteid;
                // launch query
$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);
```

10.29.5 ifx_prepare

int ifx_prepare (string **query**, int **conn_id**, int **cursor_def** , mixed **BLOBidarray**)
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().
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 affectérs (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().

10.29.6 ifx_do

int ifx_do (int **result_id**)
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.

10.29.7 ifx_error

string ifx_error
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
? : Undéfinie
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
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().

10.29.8 ifx_errormsg

string ifx_errormsg (int **errorcode**)
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
", ifx_errormsg(-201));

10.29.9 ifx_affected_rows

int ifx_affected_rows (int **result_id**)
result_id est un identifiant valide de résultat retourné par ifx_query() ou ifx_prepare().
Retourne le nombre de lignes affectées par la requête associée à **result_id**.
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.
Cette fonction est très pratique après ifx_prepare() pour limiter la taille des résultats.
Voir aussi : ifx_num_rows().

```
$rid = ifx_prepare ("select * from emp
where name like " . $name, $connid);
if (! $rid) {
   ... error ...
}
$rowcount = ifx_affected_rows ($rid);
if ($rowcount > 1000) {
printf ("Trop de lignes trouvées (%d)\n<br>", $rowcount);
die ("Ressayez avec une autre requête. <br>\n");
}
```

10.29.10 ifx_getsqlca

array ifx_getsqlca (int **result_id**)

result_id est un identifiant valide de résultat retourné par ifx_query() ou ifx_prepare().
Retourne une pseudo-ligne (tableau associatif) avec sqlca.sqlerrd[0] à sqlca.sqlerrd[5] après la requête associée **result_id**.
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 driver au moment approprié.

```
/* 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 column' ", $connid);
if (! $qid) {
   ... error ...
}
$sqlca = ifx_getsqlca ($qid);
$serial_value = $sqlca["sqlerrd1"];
echo "Le numéro de série de la valeur insérée est : " . $serial_value<br>\n";
```

10.29.11 ifx_fetch_row

array ifx_fetch_row (int **result_id**, mixed **position**)

Retourne un tableau associatif qui contient la ligne retournée, ou FALSE si il ne reste plus de lignes à lire, ou si 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. 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 spécifier 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 si il n'y a plus de ligne.

```
$rid = ifx_prepare ("select * from emp where name like " . $name,
                    $connid, IFX_SCROLL);
if (! $rid) {
   ... error ...
}
$rowcount = ifx_affected_rows($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);
```

10.29.12 ifx_htmltbl_result

int ifx_htmltbl_result (int **result_id**, string **html_table_options**)
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.

```
$rid = ifx_prepare ("select * from emp where name like " . $name,
                $connid, IFX_SCROLL);
if (! $rid) {
   ... error ...
}
$rowcount = ifx_affected_rows ($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);
```

10.29.13 ifx_fieldtypes

array ifx_fieldtypes (int **result_id**)
Retourne un tableau associatif avec les noms des champs comme clés, et les types SQL comme valeur. En cas d'erreur, retourne FALSE.

```
$types = ifx_fieldtypes ($resultid);
if (! isset ($types)) {
  ... error ...
}
for ($i = 0; $i < count($types); $i++) {
   $fname = key($types);
printf("%s :\t type =  %s\n", $fname, $types[$fname]);
next($types);
}
```

10.29.14 ifx_fieldproperties

array ifx_fieldproperties (int **result_id**)
Retourne un tableau associatif avec les nom des champs comme clé, et les données de propriétés des champs comme valeur. Retourne FALSE en cas d'erreur.
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"etcÖ. et ISNULLABLE = "Y" ou "N" (le champs peut contenir NULL ou pas : Oui ou Non).

```
$properties = ifx_fieldtypes ($resultid);
if (! isset($properties)) {
  ... error ...
}
for ($i = 0; $i < count($properties); $i++) {
   $fname = key ($properties);
printf ("%s:\t type =  %s\n", $fname, $properties[$fname]);
next ($properties);
}
```

10.29.15 ifx_num_fields

int ifx_num_fields (int **result_id**)
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.

10.29.16 ifx_num_rows

int ifx_num_rows (int **result_id**)
Compte le nombre de ligne déjà lues dans le résultat **result_id** après ifx_query() ou ifx_do().

10.29.17 ifx_free_result

int ifx_free_result (int **result_id**)
Libère les ressources prises par le résultat **result_id**. Retourne FALSE en cas d'erreur.

10.29.18 ifx_create_char

int ifx_create_char (string **param**)
Crée un objet char. **param** sera le contenu de l'objet.

10.29.19 ifx_free_char

int ifx_free_char (int **bid**)
Supprime l'objet char **bid**. Retourne FALSE en cas d'erreur, et sinon TRUE.

10.29.20 ifx_update_char

int ifx_update_char (int **bid**, string **content**)
Modifie le contenu de l'objet char repéré par son identifiant**bid**. **content** est une chaîne avec les nouvelles données. Retourne. FALSE en cas d'erreur, et sinon, TRUE.

10.29.21 ifx_get_char

int ifx_get_char (int **bid**)
Retourne le contenu de l'objet associé à l'identifiant **bid**.

10.29.22 ifx_create_blob

int ifx_create_blob (int **type**, int **mode**, string **param**)
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.
Retourne FALSE en cas d'erreur, et sinon, un identifiant de BLOB.

10.29.23 ifx_copy_blob

int ifx_copy_blob (int **bid**)
Retourne FALSE en cas d'erreur, et sinon, l'identifiant du nouvel objet.

10.29.24 ifx_free_blob

int ifx_free_blob (int **bid**)
Supprime l'objet BLOB **bid**. Retourne FALSE en cas d'erreur, et sinon TRUE.

10.29.25 ifx_get_blob

int ifx_get_blob (int **bid**)
Retourne le contenu de l'objet BLOB associé à **bid**.

10.29.26 ifx_update_blob

ifx_update_blob (int **bid**, string **content**)
Modifie le contenu de l'objet BLOB repéré par sont identifiant **bid**. **content** est une chaîne contenant les nouvelles données. Retourne FALSE en cas d'erreur, et sinon, TRUE.

10.29.27 ifx_blobinfile_mode

void ifx_blobinfile_mode (int **mode**)
Choisi 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.

10.29.28 ifx_textasvarchar

void ifx_textasvarchar (int **mode**)
Choisi 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.

10.29.29 ifx_byteasvarchar

void ifx_byteasvarchar (int **mode**)
Choisi 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.

10.29.30 ifx_nullformat

void ifx_nullformat (int **mode**)
Choisi le mode par défaut de lecture des valeurs. Le mode "0" retourne "", et le mode "1" retourne "NULL".

10.29.31 ifxus_create_slob

int ifxus_create_slob (int **mode**)
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. Retourne FALSE en cas d'erreur, et sinon, l'identifiant de l'objet SLOB.

10.29.32 ifx_free_slob

int ifxus_free_slob (int **bid**)
Supprime un objet SLOB. **bid** est l'identifiant de l'objet SLOB. Retourne FALSE en cas d'erreur, et sinon TRUE.

10.29.33 ifxus_close_slob

int ifxus_close_slob (int **bid**)
Ferme l'objet SLOB représenté par son identifiant **bid**. Retourne FALSE en cas d'erreur, et sinon, TRUE. TRUE.

10.29.34 ifxus_open_slob

int ifxus_open_slob (long **bid**, int **mode**)
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. Retourne FALSE en cas d'erreur, et sinon, l'identifiant du nouvel objet.

10.29.35 ifxus_tell_slob

int ifxus_tell_slob (long **bid**)
Retourne le fichier courant, ou la position courante d'un objet SLOB ouvert. **bid** est un identifiant d'objet SLOB. Retourne FALSE en cas d'erreur, et sinon, la position du pointeur de fichier.

10.29.36 ifxus_seek_slob

int ifxus_seek_slob (long **bid**, int **mode**, long **offset**)
Fixe 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. Retourne FALSE en cas d'erreur, et sinon, la position du pointeur de fichier.

10.29.37 ifxus_read_slob

int ifxus_read_slob (long **bid**, long **nbytes**)
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. Retourne FALSE en cas d'erreur, et sinon, une chaîne de caractères.

10.29.38 ifxus_write_slob

int ifxus_write_slob (long **bid**, string **content**)
Ecrit une chaîne dans un objet SLOB. **bid** est un identifiant d'objet SLOB et **content** sont les données à écrire. Retourne FALSE en cas d'erreur, et sinon, le nombre d'octets écrits.

10.30 sur les images

Vous pouvez utiliser les fonctions PHP pour obtenir les tailles des images aux formats JPEG, GIF, et PNG, et si vous avez la librairie GD (disponible à http://www.boutell.com/gd/) vous pourrez aussi créer et manipuler ces images.

10.30.1 GetImageSize

array getimagesize (string **filename**, array **imageinfo**)
GetImageSize() va déterminer la taille des images de type **GIF**, **JPG** ou **PNG** 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.
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".

```
<?php $size = GetImageSize("img/flag.jpg"); ?>
<IMG SRC="img/flag.jpg" <?php echo $size[3]; ?>>
```

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.xe.net/iptc/. Vous pouvez utiliser la fonction iptcparse() pour analyser ce marqueur, et obtenir des informations intelligibles.

```
<?php
   $size = GetImageSize("testimg.jpg",&$info);
if (isset($info["APP13"])) {
      $iptc = iptcparse($info["APP13"]);
var_dump($iptc);
   }
?>
```

Note : *Cette fonction ne requiert par la bibliothèque GD.*

10.30.2 ImageArc

int imagearc (int **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**.

10.30.3 ImageChar

int imagechar (int **im**, int **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).0
Voir aussi imageloadfont().

10.30.4 ImageCharUp

int imagecharup (int **im**, int **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().

10.30.5 ImageColorAllocate

int imagecolorallocate (int **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**.

```
$white = ImageColorAllocate($im, 255,255,255);
$black = ImageColorAllocate($im, 0,0,0);
```

10.30.6 ImageColorAt

int imagecolorat (int **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().

10.30.7 ImageColorClosest

int imagecolorclosest (int **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().

10.30.8 ImageColorExact

int imagecolorexact (int **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 palettre, retourne -1.
Voir aussi imagecolorclosest().

10.30.9 ImageColorResolve

int imagecolorresolve (int **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().

10.30.10 ImageColorSet

bool imagecolorset (int **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().

10.30.11 ImageColorsForIndex

array imagecolorsforindex (int **im**, int **index**)
imagecolorsforindex() retourne un tableau associatif avec les couleur rouge (red) , vert (green), bleu (blue) qui contiennent les valeurs de la couleur correspondante.
Voir aussi imagecolorat() et imagecolorexact().

10.30.12 ImageColorsTotal

int imagecolorstotal (int **im**)
imagecolorstotal() retourne le nombre de couleur de la palette.
Voir aussi imagecolorat() et imagecolorsforindex().

10.30.13 ImageColorTransparent

int imagecolortransparent (int **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 transparante est retourné.

10.30.14 ImageCopyResized

int imagecopyresized (int **dst_im**, int **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 se repère par rapport au coin supérieur gauche. Cete 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.

10.30.15 ImageCreate

int imagecreate (int **x_size**, int **y_size**)
imagecreate() retourne un identifiant d'image représentant une image blanche, de largeur size x_size et longueur y_size.

10.30.16 ImageCreateFromGif

int 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 :

```
function LoadGif($imgname)
{
  $im = @imagecreatefromgif($imgname); /* Tentative d'ouverture */
if ($im == "") { /* Echec ? */
    $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.*

10.30.17 ImageDashedLine

int imagedashedline (int **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().

10.30.18 ImageDestroy

int imagedestroy (int **im**)
imagedestroy() libère toute la mémoire associée avec l'image **im**. **im** est un identifiant d'image valide retourné par imagecreate().

10.30.19 ImageFill

int imagefill (int **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)).

10.30.20 ImageFilledPolygon

int imagefilledpolygon (int **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.

10.30.21 ImageFilledRectangle

int imagefilledrectangle (int **im**, int **x1**, int **y1**, int **x2**, int **y2**, int **col**)
imagefilledrectangle() dessine un rectangle de couleur **col** dans l'image **im**, en commencant par le sommet supérieur gauche (**x1**, **y1**) et finissant au sommet inférieur droit (**x2**, **y2**). Le coin supérieur gauche est l'origine (0, 0).

10.30.22 ImageFillToBorder

int imagefilltoborder (int **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)).

10.30.23 ImageFontHeight

int imagefontheight (int **font**)
imagefontheight() retourne la hauteur de la police en pixel.
Voir aussi imagefontwidth() et imageloadfont().

10.30.24 ImageFontWidth

int imagefontwidth (int **font**)
imagefontwidth() retourne la largeur de la police en pixels.
Voir aussi imagefontheight() et imageloadfont().

10.30.25 ImageGif

int imagegif (int **im**, string **filename**)
imagegif() crée un fichier image GIF avec le nom **filename** d'après l'iamge **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'ai 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 image de type image/gif content-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.*

10.30.26 ImageInterlace

int imageinterlace (int **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.
Cette fonction retourne l'état courant d'entrelacement de l'image.

10.30.27 ImageLine

int imageline (int **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().

10.30.28 ImageLoadFont

int 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.

position	Type de donnés 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) @tab
Octets 8-11	int	Largeur en pixel des caractères
Octets 12-15	int	Hauteur en pixel 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. @tab

Voir aussi ImageFontWidth() et ImageFontHeight().

10.30.29 ImagePolygon

int imagepolygon (int **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().

10.30.30 ImagePSBBox

array imagepsbbox (string **text**, int **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 ajouté 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 (? ? ?).
Les paramètres **space**, **tightness** et **angle** sont optionnels.
Le rectangle entourant est calculé en utilisant les informations disponibles sur les tailels de caractères, et, malheureusement, ont 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.
Cette fonction 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().

10.30.31 ImagePSEncodeFont

int imagepsencodefont (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 de disposent pas des caractères au-delà de 127, vous aurez surement besoin de les changer sur 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 7.1 Le fichier de configuration pour utiliser par défaut l'encodage correct.

10.30.32 ImagePSFreeFont

void imagepsfreefont (int **fontindex**)
Voir aussi imagepsloadfont().

10.30.33 ImagePSLoadFont

int imagepsloadfont (string **filename**)
Au cas oú 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.
Voir aussi imagepsfreefont().

10.30.34 ImagePSText

array imagepstext (int **image**, string **text**, int **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(), oú (x, y) définissait le coin supérieur gauche du premier caracètre. 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'espace caractère, ce qui vaut 1/1000ème d'un em-carré (? ? ?).
Les paramètres **space**, **tightness**, **angle** et **antialias** sont optionnels.
Cette fonction 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().

10.30.35 ImageRectangle

int imagerectangle (int **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 commencant au point supérieur gauche (**x1**,**y1**), et en finissant au point inférieur droit (**x2**,**y2**). Le coin supérieur gauche est l'origine (0,0).

10.30.36 ImageSetPixel

int imagesetpixel (int **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().

10.30.37 ImageString

int imagestring (int **im**, int **font**, int **x**, int **y**, string **s**, int **col**)
imagestring() dessine 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().

10.30.38 ImageStringUp

int imagestringup (int **im**, int **font**, int **x**, int **y**, string **s**, int **col**)
imagestringup() dessine une chaîne 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().

10.30.39 ImageSX

int imagesx (int **im**)
imagesx() retourne la largeur de l'image référencée par **im**.
Voir aussi imagecreate() et imagesy().

10.30.40 ImageSY

int imagesy (int **im**)
imagesy() retourne la hauteur de l'image référencée par **im**.
Voir aussi imagecreate() et imagesx().

10.30.41 ImageTTFBBox

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.
  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, représentants les 4 sommets du rectangle ainsi définis.

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.
Cette fonction requiert les bibliothèques GD et Freetype.
Voir aussi ImageTTFText().

10.30.42 ImageTTFText

array ImageTTFText (int **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 commancant 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 représentants les 4 points marquant 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'horizontal.
Cet exemple va générer une image GIF noire de 400x30 pixels, avec les mots "Test en cours..." en police blanche, Arial.

```
<?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, $white, "/path/arial.ttf", "Test en cours... Omega: &amp;#937;");
ImageGif($im);
ImageDestroy($im);
?>
```

Cette fonction requiert les bibliothèques GD et FreeType.
Voir aussi ImageTTFBBox().

10.31 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 developpées 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.

10.31.1 imap_append

int imap_append (int **imap_stream**, string **mbox**, string **message**, string**flags**)
Retourne TRUE en cas de succès, et FALSE en cas d'erreur.
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.
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.

10.31.2 imap_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.

10.31.3 imap_body

string imap_body (int **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) si il n'est pas déjà levé.
- FT_INTERNAL - La chaîne renvoyée est au format interne, et ne va pas canoniser les CRLF.

10.31.4 imap_check

object imap_check (int **imap_stream**)
imap_check() retourne les informations à propos de la boîte aux lettres courante. Retourne FALSE en cas d'échec.
La fonction 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 du message
Driver : Pilote
Mailbox : Nom de la boîte aux lettres
Nmsgs : Nombre de messages
Recent : Nombre de messages récents

10.31.5 imap_close

int imap_close (int **imap_stream**, int **flags**)
Termine un flot IMAP. Prend un argument optionnel **flag**, CL_EXPUNGE, qui va retirer automatiquement de la liste la boîte aux lettres.

10.31.6 imap_createmailbox

int imap_createmailbox (int **imap_stream**, string **mbox**)
imap_createmailbox() crée une nouvelle boîte aux lettres nommée **mbox** (voir imap_open() pour connaître le format des noms de **mbox**).
Retourne TRUE en cas de succès, et FALSE en cas d'erreur.
Voir aussi imap_renamemailbox() et imap_deletemailbox().

10.31.7 imap_delete

int imap_delete (int **imap_stream**, int **msg_number**)
Returns TRUE.
imap_delete() marque le fichier **msg_number** pour l'effacement, dans la boîte aux lettres courante. L'effacement réel n'interviendra que lors de l'appel de la fonction imap_expunge().

10.31.8 imap_deletemailbox

int imap_deletemailbox (int **imap_stream**, string **mbox**)
imap_deletemailbox() efface la boîte aux lettres (voir imap_open() pour connaître le format des noms de **mbox**).
Retourne TRUE en cas de succès, et FALSE en cas d'erreur.
Voir aussi imap_createmailbox() et imap_renamemailbox().

10.31.9 imap_expunge

int imap_expunge (int **imap_stream**)
imap_expunge() efface tous les messages marqués pour l'effacement par imap_delete().
Retourne TRUE.

10.31.10 imap_fetchbody

string imap_fetchbody (int **imap_stream**, int **msg_number**, string **part_number**, flags **flags**)
Cette fonction 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) si il n'est pas déjà levé.
- FT_INTERNAL - La chaîne renvoyée est au format interne, et ne va pas canoniser les CRLF.

10.31.11 imap_fetchstructure

object imap_fetchstructure (int **imap_stream**, int **msg_number**, int **flags**)
Lit la structure du message **msg_number**. Cette fonction dispose d'une option **flags**, qui une seule valeur, **FT_UID**, pour indiquer que l'argument **msg_number** est un **UID**. Cette fonction 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 :

type	Type primaire de corps
encoding	Codage de transfert du corps
ifsubtype	TRUE si il y a une chaîne de sous type
subtype	sous type**MIME**
ifdescription	TRUE si il y au ne chaîne de description
description	Chaîne de description du contenu
ifid	TRUE si il y a une chaîne d'identification
id	Chaîne d'identification
lines	Nombre de lignes
bytes	Nombre d'octets
ifdisposition	TRUE si il y a une chaîne de disposition
disposition	Chaîne de disposition
ifdparameters	TRUE s'il y a un tableau de paramètres dparameters
dparameters dparameters est un tableau d'objet oú chaque objet à un "attribut" et une "valeur". @tab tableau de disposition	
ifparameters	TRUE si le tableau de paramètres existe
paramètres parameter est un tableau d'objet oú chaque objet à un "attribut" et une "valeur". @tab Tableau de paramètres **MIME**	
parts parts est un tableau d'objets de même structure que l'objet supérieur, mais qui ne contient pas d'autres objets de même sorte. @tab Tableau d'objet décrivant chaque partie du message	

0	text
1	multipart
2	message
3	application
4	audio
5	image
6	video
7	other

0	7BIT
1	8BIT
2	BINARY
3	BASE64
4	QUOTED-PRINTABLE
5	OTHER

10.31.12 imap_header

object imap_header (int **imap_stream**, int **msg_number**, int **fromlength** , int **subjectlength** , string **defaulthost**)
Cette fonction retourne un objet avec divers éléments d'entête :
remail, date, Date, subject, Subject, in_reply_to, message_id,
newsgroups, followup_to, references
message flags:
Recent - 'R' si récent et lu
'N' si récent et pas lu
' ' if par récent
Unseen - 'U' si pas lu ET pas récent
' ' si pas lu OU pas lu et récent
Answered -'A' si répondu
' ' si non répondu
Deleted - 'D' si effacé
' ' si pas effacé
Draft - 'X' si brouillon
' ' si pas brouillon
Flagged - 'F' si marqué
' ' si non marqué
NOTE : le comportement de Récent/pas lu est un peu étrange. Si vous voulez
savoir si un message n'est pas lu, vous devez vérifier avec
Unseen == 'U' || Recent == 'N'
toaddress (ligne to: complète, jusqu'à 1024 caractères)
to[] (retourne un tableau d'objets, extrait de la ligne To , et contenant)
personal
adl
mailbox
host
fromaddress (ligne From: complète, jusqu'à 1024 caractères)
from[] (retourne un tableau d'objets, extrait de la ligne From , et contenant)
personal
adl
mailbox
host
ccaddress (ligne cc: complète, jusqu'à 1024 caractères)
cc[] (retourne un tableau d'objets, extrait de la ligne Cc , et contenant)
personal
adl
mailbox
host
bccaddress (ligne bcc: complète, jusqu'à 1024 caractère)
bcc[] (retourne un tableau d'objets, extrait de la ligne Bcc , et contenant)
personal
adl
mailbox
host
reply_toaddress (ligne reply_to: complète, jusqu'à 1024 caractères)
reply_to[] (retourne un tableau d'objets, extrait de la ligne Reply_to , et contenant)
personal
adl
mailbox
host
senderaddress (ligne sender_address: complète, jusqu'à 1024 caractères)
sender[] (retourne un tableau d'objets, extrait de la ligne Reply_to , et contenant)
personal
adl
mailbox
host
return_path (ligne return_path: complète, jusqu'à 1024 caractères)
return_path[] (retourne un tableau d'objets, extrait de la ligne return_path , et contenant)
personal

adl
mailbox
host
udate (date du message, au format unix (timestamps))
fetchfrom (Ligne From , limitée à **fromlength**caractères)
fetchsubject (Ligne de sujet, limitée à **subjectlength** caractères)

10.31.13 imap_headers

array imap_headers (int **imap_stream**)
imap_headers() retourne un tableau de chaîne contenant les entête des messages. Une chaîne par message.

10.31.14 imap_listmailbox

array imap_listmailbox (int **imap_stream**, string **ref**, string **pat**)
Retourne un tableau contenant les noms des boîtes aux lettres.

10.31.15 imap_getmailboxes

array imap_getmailboxes (int **imap_stream**, string **ref**, string **pat**)
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, oú 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.

10.31.16 imap_listsubscribed

array imap_listsubscribed (int **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 oú chercher et le nom des boîtes recherchées, sous la forme d'un masque.

10.31.17 imap_getsubscribed

array imap_getsubscribed (int **imap_stream**, string **ref**, string **pattern**)
Cette fonction est identique à imap_getmailboxes(), mais ne retourne que les boîtes aux lettres auxquelles l'utilisateur est inscrit.

10.31.18 imap_mail_copy

int imap_mail_copy (int **imap_stream**, string **msglist**, string **mbox**, int **flags**)
imap_mail_copy() retourne TRUE en cas de succès et FALSE en cas d'erreur.
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.
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.

10.31.19 imap_mail_move

int imap_mail_move (int **imap_stream**, string **msglist**, string **mbox**)
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.
Retourne TRUE en cas de succès, et FALSE en cas d'erreur.

10.31.20 imap_num_msg

int imap_num_msg (int **imap_stream**)
imap_num_msg() retourne le nombre de message dans la boîte aux lettres courante.

10.31.21 imap_num_recent

int imap_num_recent (int **imap_stream**)
imap_num_recent() retourne le nombre de message récents dans la boîte aux lettres courante.

10.31.22 imap_open

int imap_open (string **mailbox**, string **username**, string **password**, int **flags**)
Retourne un flot IMAP en cas de succès, et FALSE en cas d'erreur. Cette fonction peut aussi être utilisée pour ouvrir des flots sur des serveurs POP3 et NNTP. Pour se connecter à un serveur IMAP, on peut utiliser la commande suivante :
$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 :
$mbox = imap_open("{localhost/pop3:110}INBOX","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:
$nntp = imap_open("{localhost/nntp:119}comp.test","","");
Pour se connecter à un serveur distant, remplacez "localhost" par le nom ou l'adresse IP de la machine.
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 lettre.
- CL_EXPUNGE - Supprime automatiquement la boîte aux lettres de la liste, lors de la terminaison du flot.

10.31.23 imap_ping

int imap_ping (int **imap_stream**)
Retourne TRUE si le flot 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.

10.31.24 imap_renamemailbox

int imap_renamemailbox (int **imap_stream**, string **old_mbox**, string **new_mbox**)
Renomme une boîte aux lettres.
Retourne TRUE en cas de succès, et FALSE en cas d'erreur.
Voir aussi imap_createmailbox() et imap_deletemailbox().

10.31.25 imap_reopen

int imap_reopen (string **imap_stream**, string **mailbox**, string **flags**)

Retourne TRUE en cas de succès, et FALSE en cas d'erreur.
Cette fonction ré-ouvre le flot spécifié, mais vers 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 lettre.
- CL_EXPUNGE - Supprime automatiquement la boîte aux lettres de la liste, lors de la terminaison du flot.

10.31.26 imap_subscribe

int imap_subscribe (int **imap_stream**, string **mbox**)
Souscrit à la boîte aux lettres **mbox**.
Retourne TRUE en cas de succès, et FALSE en cas d'échec.

10.31.27 imap_undelete

int imap_undelete (int **imap_stream**, int **msg_number**)
Enlève la marque d'effacement du message **msg_number**, placée avec imap_delete().
Retourne TRUE en cas de succès, et FALSE en cas d'erreur.

10.31.28 imap_unsubscribe

int imap_unsubscribe (int **imap_stream**, string **mbox**)
Termine la souscription à la boîte aux lettres **mbox**.
Retourne TRUE en cas de succès, et FALSE en cas d'erreur.

10.31.29 imap_qprint

string imap_qprint (string **string**)
Convertit la chaîne à guillemets **string** en une chaîne à 8 bits.
Retourne une chaîne 8 bits (binaire).

10.31.30 imap_8bit

string imap_8bit (string **string**)
Convertit la chaîne à 8 bits en une chaîne à guillemets.
Retourne une chaîne à guillemets.

10.31.31 imap_binary

string imap_binary (string **string**)
Convertit la chaîne à 8 bits **string** en une chaîne à base64.
Retourne la chaîne codée.

10.31.32 imap_scanmailbox

array imap_scanmailbox (int **imap_stream**, string **string**)
Lis la liste des boîtes aux lettres, et y recherche la chaîne **string**.

10.31.33 imap_mailboxmsginfo

object imap_mailboxmsginfo (int **imap_stream**)
Retourne les informations à propos de la boîte aux lettres courante. 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 :
Date : date du message
Driver : lecteur
Mailbox : Nom de la boîte aux lettres
Nmsgs : nombre de messages
Recent : nombre de messages récents
Unread : nombre de messages non lus

Size : taille de la boîte aux lettres

10.31.34 imap_rfc822_write_address

string imap_rfc822_write_address (string **mailbox**, string **host**, string **personal**)
Retourne une adresse email proprement formatée, à partir du nom de la boîte aux lettre de l'hôte, et des informations personnelles.

10.31.35 imap_rfc822_parse_adrlist

string imap_rfc822_parse_adrlist (string **address**, string **default_host**)
Cette fonction analyse la chaîne **address** et essaie, pour chaque adresse, retourne un tableau d'objets. les 4 objets sont :
mailbox - Le nom de la boîte aux lettres (nom d'utilisateur)
host - Le nom de l'hôte
personal - Le nom de l'utilisateur
adl - La route jusqu'au domaine

10.31.36 imap_setflag_full

string imap_setflag_full (int **stream**, string **sequence**, string **flag**, string **options**)
Cette fonction 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

10.31.37 imap_clearflag_full

string imap_clearflag_full (int **stream**, string **sequence**, string **flag**, string **options**)
Cette fonction efface le flag spécifié dans les messages de la séquence **sequence**.
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

10.31.38 imap_sort

string imap_sort (int **stream**, int **criteria**, int **reverse**, int **options**)
Retourne un tableau de nombre de message, triés suivant les paramètres suivants :
Rev vaut 1 pour signifier : tri inverse (REVevse-sorting).
Les critères peuvent être un (et un seul) parmis 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

10.31.39 imap_fetchheader

string imap_fetchheader (int **imap_stream**, int **msgno**, int **flags**)
Cette fonction retourne l'entête brut et complet 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'entête. Cela réduit le RTT sur une connexion IMAP, si le message complet est souhaité. (e.g. dans une opératoin de sauvegarde dans un fichier).

10.31.40 imap_uid

int imap_uid (int **imap_stream**, int **msgno**)
Cette fonction retourne l'UID pour le message **msgno**. C'est la fonction contraire de imap_msgno().

10.31.41 imap_msgno

int imap_msgno (int **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().

10.31.42 imap_search

array imap_search (int **imap_stream**, string **criteria**, int **flags**)
imap_search() effectue une recherche dans la boîte aux lettres courantes, 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. Faites attention.
Les valeurs pour les flags sont SE_UID, qui fait que le tableau réponse contient les UIDs plutôt que les numéros de séquence.

10.31.43 imap_last_error

string imap_last_error (void)
Cette fonction 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.
ATTENTION : cette fonction n'est pas encore disponible sous PHP4.

10.31.44 imap_errors

array imap_errors (void)
Cette fonction 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.
ATTENTION : cette fonction n'est pas encore disponible sous PHP4.

10.31.45 imap_alerts

array imap_alerts (void)
Cette fonction 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'alerte est vidée.

10.31.46 imap_status

object imap_status (int **imap_stream**, string **mailbox**, int **options**)
Cette fonction retourne un objet contenant les informations de statut. Les flags 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 prochaîne 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.

10.31.47 imap_utf7_decode

string imap_utf7_decode (string **text**)
Décode la chaîne UTF-7 **text** en données 8 bits.
Retourne les données décodées en 8 bits, ou FALSE si la chaîne n'était pas une chaîne UTF-7 valide.

10.31.48 imap_utf7_encode

string imap_utf7_encode (string **data**)

Convertit les données 8 bits **data** en texte UTF-7 text. L'encodage UTF-7 est défini dans la RFC 2060. Retourne un texte UTF-7.

10.32 Options PHP & informations

10.32.1 error_log

int error_log (string **message**, int **message_type**, string **destination** , string **extra_headers**)

Envoie un message d'erreur dans les logs du serveur, sur un port **TCP**, ou bien à un fichier. Le premier paramètre, **message**, est le message qui doit être envoyé. Le deuxième paramètre, **message_type** indique l'endroit oú le message doit aller:

0	Le message **message** est envoyé au système de log du PHP, en utilisant la fonction système de logs, en fonction de la configuration de l'option 7.1.1.8 ini.error-log. @tab
1	**message** Le message est envoyé par mail à l'adresse donnée dans le paramètre **destination**. C'est le seul moment oú le quatrième paramètres, **extra_headers** est utilisé. Ce type de message utilise les mêmes fonctions internes que la fonction mail(). @tab
2	Le message **message** est envoyé au travers de la connexion de débuggage. Cette option est disponible uniquement si 4.2.7.13 --enable-debugger a été activée. Dans ce cas, le paramètre **destination** spécifie le nom de l'hôte ou l'adresse IP ainsi que, optionnellement, le numéro de port de la socket recevant les informations de débugage. @tab
3	Le message **message** est simplement écrit dans le fichier **destination**. @tab

```
// Envoie un message à l'administrateur si on ne peut
// pas se connecter à la base de donnée.
if (!Ora_Logon($username, $password)) {
error_log("Base de données Oracle indisponible!", 0);
}
// Envoie un mail à l'administrateur.
if (!($foo = allocate_new_foo()) {
error_log("Gros problème de connexion à la base!", 1,
          "operator@mydomain.com");
}
// Une autre manière d'appeler la fonction 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");
```

10.32.2 error_reporting

int error_reporting (int **level**)

établit le niveau d'erreur à prendre en compte et renvoie l'ancien niveau. Le niveau d'erreur est un champs de bits qui utilise les valeurs suivantes: (suivez les liens pour connaître les valeurs et leur signification)

Valeur	Nom interne	
1	C.3.3 E_ERROR @tab	
2	C.3.2 E_WARNING @tab	
4	C.3.4 E_PARSE @tab	
8	C.3.1 E_NOTICE @tab	
16	E_CORE_ERROR @tab	
32	E_CORE_WARNING @tab	

10.32.3 extension_loaded

bool extension_loaded (string **name**)

Retourne vraie 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.*

10.32.4 getenv

string getenv (string **varname**)
Retourne la valeur de la variable d'environnement **varname**, ou FALSE en cas d'erreur.

```
$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 trouvez la signification de chacune d'entre elles en consultant le site concernant CGI specification (en anglais>, et particulièment la page concernant les variables d'environnement..

10.32.5 get_cfg_var

string get_cfg_var (string **varname**)
Retourne la valeur courante de l'option PHP **varname**, ou bien FALSE en cas d'erreur.
Cette fonction 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 7.1 Le fichier de configuration, essayez de lire la valeur de cfg_file_path. Si cette valeur est disponbile, alors le fichier de configuration est utilisé.

10.32.6 get_current_user

string get_current_user (void)
Retourne le nom du possesseur du script courant.
Voir aussi getmyuid(), getmypid(), getmyinode(), et getlastmod().

10.32.7 get_magic_quotes_gpc

long get_magic_quotes_gpc (void)
Retourne le configuration actuelle de l'option 7.1.1.16 ini.magic-quotes-gpc (0 pour l'option désactivée, 1 pour l'option activée).
Voir aussi get_magic_quotes_runtime(), set_magic_quotes_runtime().

10.32.8 get_magic_quotes_runtime

long get_magic_quotes_runtime (void)
RRetourne la configuration actuelle de l'option 7.1.1.17 ini.magic-quotes-runtime. (0 pour option desactivée, 1 pour option sactivée).
Voir aussi get_magic_quotes_gpc(), set_magic_quotes_runtime().

10.32.9 getlastmod

int getlastmod (void)
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(). Retourne FALSE en cas d'erreur.

```
// 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().

10.32.10 getmyinode

int getmyinode (void)
Retourne l'inode du script, ou FALSE en cas d'erreur.
Voir aussi getmyuid(), get_current_user(), getmypid(), et getlastmod().
Note : *Cette fonction est inopérante sur les systèmes Windows.*

10.32.11 getmypid

int getmypid (void)
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 guarantit que deux invocations distinctes de la fonction donnent des résultats différents.
Voir aussi getmyuid(), get_current_user(), getmyinode(), et getlastmod().

10.32.12 getmyuid

int getmyuid (void)
Retourne l'UID du propriétaire du script actuel ou FALSE en cas d'erreur.
Voir aussi getmypid(), get_current_user(), getmyinode(), et getlastmod().

10.32.13 getrusage

array getrusage (int **who**)
Cette fonction 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.

```
$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.

10.32.14 phpinfo

int phpinfo (void)
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, entêtes HTTP, et licence GNU Public License.
Voir aussi phpversion().

10.32.15 phpversion

string phpversion (void)
Retourne le numéro de la version courante de PHP.

```
// affiche le numéro de version courante du PHP.
echo "PHP Version: ".phpversion();
```

Voir aussi phpinfo().

10.32.16 php_logo_guid

string php_logo_guid (void)
Note : *Cette fonctionnalité a été ajoutée dans PHP4 Beta 4.*

10.32.17 putenv

void putenv (string **setting**)
Fixe la valeur d'une variable d'environnement.

```
putenv("UNIQID=$uniqid");
```

10.32.18 set_magic_quotes_runtime

long set_magic_quotes_runtime (int **new_setting**)
Active/désactive l'option 7.1.1.17 ini.magic-quotes-runtime. (0 l'option est désactivée, 1 l'option est activée).
Voir aussi get_magic_quotes_gpc() et get_magic_quotes_runtime().

10.32.19 set_time_limit

void set_time_limit (int **seconds**)
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 7.1 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 7.1 Le fichier de configuration.

10.32.20 zend_logo_guid

string zend_logo_guid (void)
Note : *Cette fonctionnalité a été ajouté en PHP4 Beta 4.*

10.33 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 équipement 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.
Une telle référence en LDAP sera exprimée avec une autre syntaxe, la syntaxe à "noms distincts" ("distinguished names"), aussi appelé "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.

10.33.1 Exemple complet

Recupérer toutes les entrées dont le nom commence par "S" dans un serveur, et afficher le nom et l'adresse email.

```
// Structure d'une commande simple :
// connexion, lien, recherche, interpretation de la recherche
// résultat, déconnexion
<?php
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>";
}
?>
```

10.33.1.1 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 serveur fournisse un accès en lien anonymes ("anonymous bind") mais requièrent un mot de passe pour tout le reste).

Une séquence habituelle LDAP suivra le schéma suivant :

```
ldap_connect()    // établi 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"
```

10.33.1.2 Plus d'informations

Vous pouvez en apprendre encore plus, mais en anglais, aux adresses suivantes :

- Netscape
- University of Michigan
- OpenLDAP Project
- LDAP World

Le SDK Netscape contient un guide du programmeur au format HTML particulièrement pratique.

10.33.2 ldap_add

int ldap_add (int **link_identifier**, string **dn**, array **entry**)
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

<?php
$ds=ldap_connect("localhost");  // On suppose que le serveur LDAP est sur cet hote
if ($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 ";
}
?>
```

10.33.3 ldap_mod_add

int ldap_mod_add (int **link_identifier**, string **dn**, array **entry**)
Retourne TRUE en cas de succès, et FALSE sinon.
Cette fonction ajoute les attributs dans la nd spécifié. Cet ajout est fait au niveau attribut, et non pas au niveau objet. Les ajouts au niveau objet sont faits par la fonction ldap_add().

10.33.4 ldap_mod_del

int ldap_mod_del (int **link_identifier**, string **dn**, array **entry**)
Retourne TRUE en cas de succès, et FALSE sinon.
Cette fonction efface les attributs du nd **dn** spécifié. Cette modification est fait au niveau des attributs, et par opposition au niveau de l'objet. Les effacements au niveau objet sont effectués par la fonction ldap_delete().

10.33.5 ldap_mod_replace

int ldap_mod_replace (int **link_identifier**, string **dn**, array **entry**)
Retourne TRUE en cas de succès, et FALSE sinon.
Cette fonction remplace les attributs dans du nd **dn** spécifié. Cette modification est faite au niveau attribut, et non pas au niveau objet. Les modifications au niveau objet sont faites par la fonction ldap_modify().

10.33.6 ldap_bind

int ldap_bind (int **link_identifier**, string **bind_rdn**, string **bind_password**)
ldap_bind() lie un serveur LDAP avec le RDN et mot de passe spécifié (éventuellement). 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. Si ils manquent, la liaison se fera en mode anonyme.

10.33.7 ldap_close

int ldap_close (int **link_identifier**)
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().

10.33.8 ldap_connect

int ldap_connect (string **hostname**, int **port**)
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.

10.33.9 ldap_count_entries

int ldap_count_entries (int **link_identifier**, int **result_identifier**)
Retourne le nombre d'entrées en cas de succès, et FALSE sinon.
ldap_count_entries() retourne le nombre d'entrée placées dans le résultat par les recherches précédentes. **result_identifier** identifie un résultat LDAP interne.

10.33.10 ldap_delete

int ldap_delete (int **link_identifier**, string **dn**)
Retourne TRUE en cas de succès, et FALSE sinon.
ldap_delete() efface une entrée dans un dossier LDAP spécifié par le nd **dn**.

10.33.11 ldap_dn2ufn

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.

10.33.12 ldap_explode_dn

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.

10.33.13 ldap_first_attribute

string ldap_first_attribute (int **link_identifier**, int **result_entry_identifier**, int **ber_identifier**)
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().

10.33.14 ldap_first_entry

int ldap_first_entry (int **link_identifier**, int **result_identifier**)
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 prochaîne entrée.
Voir aussi ldap_get_entries().

10.33.15 ldap_free_result

int ldap_free_result (int **result_identifier**)
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 oú le script effectue plusieurs recherches successives, oú que les résultats retournés sont très grands, ldap_free_result() permet de réduire la consommation de mémoire.

10.33.16 ldap_get_attributes

array ldap_get_attributes (int **link_identifier**, int **result_entry_identifier**)
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.

```
// $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().

10.33.17 ldap_get_dn

string ldap_get_dn (int **link_identifier**, int **result_entry_identifier**)
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.

10.33.18 ldap_get_entries

array ldap_get_entries (int **link_identifier**, int **result_identifier**)
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 minuscule (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().

10.33.19 ldap_get_values

array ldap_get_values (int **link_identifier**, int **result_entry_identifier**, string **attribute**)
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.
Cette fonction nécessite un pointeur de résultat **result_entry_identifier**, ce qui implique qu'il ai 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

```
// $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>";
```

10.33.20 ldap_get_values_len

array ldap_get_values_len (int **link_identifier**, int **result_entry_identifier**, string **attribute**)
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, commencant à 0.
Cette fonction est utilisée exactement comme ldap_get_values() mais elle gère les données binaires, et non pas les chaînes.

10.33.21 ldap_list

int ldap_list (int **link_identifier**, string **base_dn**, string **filter**, array **attributes**)
Retourne TRUE en cas de succès, et FALSE sinon.
ldap_list() effecture une recherche avec le filtre 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.

```
// $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] ;
```

10.33.22 ldap_modify

int ldap_modify (int **link_identifier**, string **dn**, array **entry**)
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().

10.33.23 ldap_next_attribute

string ldap_next_attribute (int **link_identifier**, int **result_entry_identifier**, int **ber_identifier**)
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().

10.33.24 ldap_next_entry

int ldap_next_entry (int **link_identifier**, int **result_entry_identifier**)
Retourne l'identifiant de l'entrée suivante, dans le résultat qui a été initialisé par ldap_first_entry(). Si 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().

10.33.25 ldap_read

int ldap_read (int **link_identifier**, string **base_dn**, string **filter**, array **attributes**)
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".
Cette fonction dispose d'un quatrième argument optionnel. Reportez vous à ldap_search().

10.33.26 ldap_search

int ldap_search (int **link_identifier**, string **base_dn**, string **filter**, array **attributes**)
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.
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 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" oú 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.

```
// $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>";
```

10.33.27 ldap_unbind

int ldap_unbind (int **link_identifier**)
Retourne TRUE en cas de succès, et FALSE sinon.
ldap_unbind() termine la liaison avec le serveur LDAP.

10.33.28 ldap_err2str

string ldap_err2str (int **errno**)
Retourne un message d'erreur.
Cette fonction 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().

```
<?php
for($i=0; $i<100; $i++) {
printf("Error $i: %s<br>\n", ldap_str2err($i));
  }
?>
```

10.33.29 ldap_errno

int ldap_errno (int **link_id**)
Retourne le numéro d'erreur LDAP généré par la dernière commande.
Cette fonction 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.

```
<?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().

10.33.30 ldap_error

string ldap_error (int **link_id**)
Retourne un message d'erreur.
Cette fonction 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().

10.34 Fonction mail

mail() envoie du courrier éléctronique.

10.34.1 mail

bool mail (string **to**, string **subject**, string **message**, string **additional_headers**)
Mail() poste automatiquement le message **message** à destination de **to**. Les destinataires multiples doivent être séparés par des virgules.

```
mail("rasmus@lerdorf.on.ca", "My Subject", "Ligne 1\nLigne 2\nLigne 3");
```

Le quatrième argument passé sera inséré à la fin de l'entête. Typiquement, cela permet d'insérer des entêtes supplémentaires. Les entêtes multiples doivent être séparés par des virgules.

```
mail("nobody@aol.com", "Le sujet", $message,
    "From: webmaster@$SERVER_NAME\nReply-To: webmaster@$SERVER_NAME\nX-Mailer: PHP/" .
phpversion());
```

10.35 mathématiques

10.35.1 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 à 10.4 mathématiques sur des nombres de taille arbitraire.

10.35.1.1 Constantes mathématiques

Les valeurs suivantes sont définies comme des constantes dans PHP:

Constante	Valeur	Description	
M_PI	3.14159265358979323846	La valeur de PI	

10.35.2 Abs

mixed abs (mixed **number**)
Retourne la valeur absolue du nombre **number**. Si le nombre est de type float, le type retourné est float, sinon, c'est integer (entier).

10.35.3 Acos

float acos (float **arg**)
Retourne l'arc cosinus de **arg** (arg en radians).
Voir aussi asin() et atan().

10.35.4 Asin

float asin (float **arg**)
Retourne l'arc sinus de **arg** (arg en radians).
Voir aussi acos() et atan().

10.35.5 Atan

float atan (float **arg**)
Retourne l'arc tangent de **arg** (arg en radians).
Voir aussi acos() et atan().

10.35.6 Atan2

float atan2 (float **y**, float **x**)

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.
Cette fonction retourne un résultat en radians, entre -PI et PI (inclus).
Voir aussi acos() et atan().

10.35.7 base_convert

strin base_convert (string **number**, int **frombase**, int **tobase**)
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 représentées par les lettres de a à z, avec a = 10 et z = 36.

```
$binary = base_convert($hexadecimal, 16, 2);
```

10.35.8 BinDec

int bindec (string **binary_string**)
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().

10.35.9 Ceil

int ceil (float **number**)
Retourne l'entier supérieur du nombre **number**. Utiliser ceil() sur un entier ne sert à rien.
NOTE: ceil() sous PHP/FI 2 retournait un float. Utilisez: $new = (double)ceil($number); pour retrouver le comportement traditionnel.
Voir aussi floor() et round().

10.35.10 Cos

float cos (float **arg**)
Retourne le cosinus de arg (arg en radians).
Voir aussi sin() et tan().

10.35.11 DecBin

string decbin (int **number**)
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().

10.35.12 DecHex

string dechex (int **number**)
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().

10.35.13 DecOct

string decoct (int **number**)
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().

10.35.14 Exp

float exp (float **arg**)
Retourne l'exponentielle de **arg**, c'est à dire e élevé à la puissance **arg**.
Voir aussi pow().

10.35.15 Floor

int floor (float **number**)

Retourne l'entier inférieur du nombre **number**. Utiliser floor() sur un entier ne sert à rien.

NOTE: floor() sous PHP/FI retournait un float. Utilisez: $new = (double)floor($number); pour retrouver le comportement traditionnel.

Voir aussi ceil() et round().

10.35.16 getrandmax

int getrandmax (void)

Retourne la plus grande valeur aléatoire possible retournée par rand().

Voir aussi rand(), srand() mt_rand(), mt_srand() et mt_getrandmax().

10.35.17 HexDec

int hexdec (string **hex_string**)

Retourne une chaîne contenant la représentation décimal du nombre **hex_string**. Le nombre le plus grand qui puisse être converti est 7fffffff en décimal, ce qui donne "2147483647 ".

Voir aussi the dechex().

10.35.18 Log

float log (float **arg**)

Retourne le logarithme naturel de arg.

10.35.19 Log10

float log10 (float **arg**)

Retourne le logarithme en base 10 de **arg**.

10.35.20 max

mixed max (mixed **arg1**, mixed **arg2**, mixed **argn**)

max() retourne la 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'argument est alors illimité.

Si au moins une valeur est de type double, elle 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.

10.35.21 min

mixed min (mixed **arg1**, mixed **arg2**, mixed **argn**)

min() retourne la plus petite valeur numérique parmi les valeurs passées en paramètre.

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'argument est alors illimité.

Si au moins une valeur est de type double, elle 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.

10.35.22 mt_rand

int mt_rand (**int min** , **int max**)

De nombreux générateurs de nombre 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 nombre 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.

Appelé sans les arguments optionnels **min**, **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 nombre aléatoire avec mt_srand().
Note : *Dans les versions antérieure 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().

10.35.23 mt_srand

void mt_srand (int **seed**)
Initialise une meilleure valeur aléatoire avec **seed**.

```
// 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().

10.35.24 mt_getrandmax

int mt_getrandmax (void)
Retourne la plus grand valeur aléatoire possible que peut retourner mt_rand().
Voir aussi mt_rand(), mt_srand() rand(), srand() et getrandmax().

10.35.25 number_format

string number_format (float **number**, int **decimals**, string **dec_point**, string **thousands_sep**)
number_format() retourne une chaîne représentant **number** formaté. Cette fonction accepte un, deux ou 4 paramètres (mais pas trois).
Si un seul paramètre est donné, **number** sera formaté sans décimale, mais avec une virgule entre chaque série de 1000.
Avec deux paramètres, **number** sera formaté avec **decimals** décimales et un point ("."), une virgule entre chaque série de 1000.
Avec quatre paramètres, **number** sera formaté avec **decimals** décimales et **dec_point** à la place du point, une virgule entre chaque série de 1000.

10.35.26 OctDec

int octdec (string **octal_string**)
Retourne une chaîne contenant la représentation décimale du nombre **octal_string**. Le nombre le plus grand qui puisse être converti est 17777777777 en décimal, ce qui donnera "2147483647".
Voir aussi decoct().

10.35.27 pi

double pi (void)
Retourne la valeur de pi.

10.35.28 pow

float pow (float **base**, float **exp**)
Retourne **base** élevé à la puissance **exp**.
Voir aussi exp().

10.35.29 rand

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 antérieure 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().

10.35.30 round

double round (double **val**)
Retourne la valeur arrondie de **val**.
```
$foo = round( 3.4 );   // $foo == 3.0
$foo = round( 3.5 );   // $foo == 4.0
$foo = round( 3.6 );   // $foo == 4.0
```

Voir aussi ceil() et floor().

10.35.31 Sin

float sin (float **arg**)
Retourne le sinus de **arg** (arg in radians).
Voir aussi cos() et tan().

10.35.32 Sqrt

float sqrt (float **arg**)
Retourne la racine carrée de **arg**.

10.35.33 srand

void srand (int **seed**)
Initialise le générateur de nombres aléatoires avec **seed**.
```
// initialise avec les microsecondes depuis la dernière seconde entière
sret((double)microtime()*1000000);
$randval = rand();
```

Voir aussi rand(), getrandmax(), mt_rand(), mt_srand() et mt_getrandmax().

10.35.34 Tan

float tan (float **arg**)
Retourne la tangente de **arg** (arg en radians).
Voir aussi sin() et cos().

10.36 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, selectionnés 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 lorsque mcal est utilisée : MCAL_SUNDAY, MCAL_MONDAY, MCAL_TUESDAY, MCAL_WEDNESDAY, MCAL_THURSDAY, MCAL_FRIDAY, MCAL_SATURDAY, MCAL_RECUR_NONE, MCAL_RECUR_DAILY, MCAL_RECUR_WEEKLY, MCAL_RECUR_MONTHLY_MDAY, MCAL_RECUR_MONTHLY_WDAY, MCAL_RECUR_YEARLY, MCAL_JANUARY, MCAL_FEBRUARY, MCAL_MARCH, MCAL_APRIL, MCAL_MAY, MCAL_JUNE, MCAL_JULY, MCAL_AUGUGT, MCAL_SEPTEMBER, MCAL_OCTOBER, MCAL_NOVEMBER, et MCAL_DECEMBER. La plus part 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.

10.36.1 mcal_open

int mcal_open (string **calendar**, string **username**, string **password**, string **options**)
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 connnexion.

10.36.2 mcal_close

int mcal_close (int **mcal_stream**, int **flags**)
Ferme la connexion.

10.36.3 mcal_fetch_event

object mcal_fetch_event (int **mcal_stream**, int **event_id**, int **options**)
mcal_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.
- int recur_type - type de récurence
- int recur_interval - intervalle de récurence
- datetime recur_endate - date de fin de récurence
- int recur_data - données de récurence

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

10.36.4 mcal_list_events

array mcal_list_events (int **mcal_stream**, int **begin_year** , int **begin_month** , int **begin_day** , int **end_year** , int **end_month** , int **end_day**)
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.

10.36.5 mcal_append_event

int mcal_append_event (int **mcal_stream**)
mcal_append_event() enregistre l'événement global dans le calendrier MCAL **mcal_stream**.
Retourne l'uid de l'enregistement ainsi inséré.

10.36.6 mcal_store_event

int mcal_store_event (int **mcal_stream**)
mcal_store_event() enregistre l'événement global dans le calendrier MCAL **mcal_stream**.
Retourne TRUE en cas de succès, et FALSE en cas d'erreur.

10.36.7 mcal_delete_event

int mcal_delete_event (int **uid**)
mcal_delete_event() efface l'événement d'identifiant **uid**.
Retourne TRUE.

10.36.8 mcal_snooze

int mcal_snooze (int **uid**)
mcal_snooze() éteind l'alarme de l'événement identifié par l'UID **uid**.
Retourne TRUE.

10.36.9 mcal_list_alarms

array mcal_list_events (int **mcal_stream**, int **begin_year** , int **begin_month** , int **begin_day** , int **end_year** , int **end_month** , int **end_day**)
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.

10.36.10 mcal_event_init

int mcal_event_init (int **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.
Retourne TRUE.

10.36.11 mcal_event_set_category

int mcal_event_set_category (int **stream**, string **category**)
mcal_event_set_category() fixe la catégorie de la structure globale à la valeur de **category**.
Retourne TRUE.

10.36.12 mcal_event_set_title

int mcal_event_set_title (int **stream**, string **title**)
mcal_event_set_title() fixe le titre de la structure globale à la valeur de **title**.
Retourne TRUE.

10.36.13 mcal_event_set_description

int mcal_event_set_description (int **stream**, string **description**)
mcal_event_set_description() fixe la catégorie de la structure globale à la valeur de **description**.
Returns TRUE.

10.36.14 mcal_event_set_start

int mcal_event_set_start (int **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.
Retourne TRUE.

10.36.15 mcal_event_set_end

int mcal_event_set_end (int **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.
Returns TRUE.

10.36.16 mcal_event_set_alarm

int mcal_event_set_alarm (int **stream**, int **alarm**)

mcal_event_set_alarm() fixe l'alarme de la structure globale, à un nombre de minutes avant déclenchement.
Retourne TRUE.

10.36.17 mcal_event_set_class

int mcal_event_set_class (int **stream**, int **class**)
mcal_event_set_class() Fixe la classe de la structure globale. La classe vaut 0 pour public, et 1 pour privée.
Retourne TRUE.

10.36.18 mcal_is_leap_year

int mcal_is_leap_year (int **year**)
mcal_is_leap_year() retoune 1 si l'année **year** est bissextile, et 0 sinon.

10.36.19 mcal_days_in_month

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**.

10.36.20 mcal_date_valid

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.

10.36.21 mcal_time_valid

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.

10.36.22 mcal_day_of_week

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**.

10.36.23 mcal_day_of_year

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**.

10.36.24 mcal_date_compare

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.

10.36.25 mcal_next_recurrence

int mcal_next_recurrence (int **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. Retourne un objet date vide si l'événement n'a pas de réoccurence, ou si quelquechose est invalide. Utilisez **weekstart** pour déterminer le premier jour.

10.36.26 mcal_event_set_recur_none

int mcal_event_set_recur_none (int **stream**)
mcal_event_set_recur_none() supprime la récurence de la structure globale (event->recur_type est mis à MCAL_RECUR_NONE).

10.36.27 mcal_event_set_recur_daily

int mcal_event_set_recur_daily (int **stream**, int **year**, int **month**, int **day**, int **interval**)
mcal_event_set_recur_daily() fixe la récurence quotidienne de la structure globale, jusqu'à la date passée en paramètre. (la date de début est celle de la structure).

10.36.28 mcal_event_set_recur_weekly

int mcal_event_set_recur_weekly (int **stream**, int **year**, int **month**, int **day**, int **interval**, int **weekdays**)
mcal_event_set_recur_weekly() fixe la récurence hedbomadaire de la structure globale, jusqu'à la date passée en paramètre. (la date de début est celle de la structure).

10.36.29 mcal_event_set_recur_monthly_mday

int mcal_event_set_recur_monthly_mday (int **stream**, int **year**, int **month**, int **day**, int **interval**)
mcal_event_set_recur_monthly_mday() fixe la récurence de la structure globale, jusqu'à la date passée en paramètre. (la date de début est celle de la structure).

10.36.30 mcal_event_set_recur_monthly_wday

int mcal_event_set_recur_monthly_wday (int **stream**, int **year**, int **month**, int **day**, int **interval**)
mcal_event_set_recur_monthly_wday() fixe la récurence mensuelle de la structure globale, jusqu'à la date passée en paramètre. (la date de début est celle de la structure).

10.36.31 mcal_event_set_recur_yearly

int mcal_event_set_recur_yearly (int **stream**, int **year**, int **month**, int **day**, int **interval**)
mcal_event_set_recur_yearly() fixe la récurence annuelle de la structure globale, jusqu'à la date passée en paramètre. (la date de début est celle de la structure).

10.36.32 mcal_fetch_current_stream_event

int mcal_fetch_current_stream_event (int **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 récurence
- int recur_interval - intervalle de récurence
- datetime recur_endate - date de fin de récurence
- int recur_data - données de récurence

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

10.37 de cryptage

Ces fonctions utilisent mcrypt.
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".
Pour l'utiliser, téléchargez la librairie libmcrypt-x.x.tar.gz grâce par ici 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.

```
<?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.
mcrypt peut opérer en 4 modes de cryptage (CBC, OFB, CFB, and 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 by 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 alétaoires, 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.

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'age.

- MCRYPT_BLOWFISH
- MCRYPT_DES
- MCRYPT_TripleDES

- MCRYPT_ThreeWAY
- MCRYPT_GOST
- MCRYPT_CRYPT
- MCRYPT_DES_COMPAT
- MCRYPT_SAFER64
- MCRYPT_SAFER128
- MCRYPT_CAST128
- MCRYPT_TEAN
- MCRYPT_RC2
- MCRYPT_TWOFISH (pour les versions mcrypt 2.x)
- MCRYPT_TWOFISH128 (TWOFISHxxx est disponible dans les nouvelles versions de mcrypt 2.x)
- MCRYPT_TWOFISH192
- MCRYPT_TWOFISH256
- MCRYPT_RC6
- MCRYPT_IDEA

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).

10.37.1 mcrypt_get_cipher_name

string mcrypt_get_cipher_name (int **cipher**)
mcrypt_get_cipher_name() retourne le nom du chiffrement utilisé.
mcrypt_get_cipher_name() prend le numéro de chiffrement comme paramètre, et retourne le nom du chiffrement, ou FALSE, si ce chiffrement n'existe pas.

```
<?php
$cipher = MCRYPT_TripleDES;
print mcrypt_get_cipher_name($cipher);
?>
```

L'exemple ci dessus va donner :
TripleDES

10.37.2 mcrypt_get_block_size

int mcrypt_get_block_size (int **cipher**)
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().

10.37.3 mcrypt_get_key_size

int mcrypt_get_key_size (int **cipher**)
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().

10.37.4 mcrypt_create_iv

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.

```
<?php
$cipher = MCRYPT_TripleDES;
$block_size = mcrypt_get_block_size($cipher);
$iv = mcrypt_create_iv($block_size, MCRYPT_DEV_RANDOM);
?>
```

10.37.5 mcrypt_cbc

int mcrypt_cbc (int **cipher**, string **key**, string **data**, int **mode**, string **iv**)
mcrypt_cbc() crypte ou décrypte (suivant le mode **mode**) les données **data** avec le chiffrement **cipher** et la clé **key** en mode CBC et retourne la chaîne résultat.
cipher est une des constantes de type MCRYPT_ciphername.
key est la clé à fournir à l'algorithme. Cette clé doit rester secrète.
data sont les données à crypter/décrypter.
mode vaut MCRYPT_ENCRYPT or MCRYPT_DECRYPT.
iv est le vecteur d'initialisation.
Voir aussi: mcrypt_cfb(), mcrypt_ecb(), mcrypt_ofb().

10.37.6 mcrypt_cfb

int mcrypt_cfb (int **cipher**, string **key**, string **data**, int **mode**, string **iv**)
mcrypt_cfb() crypte ou décrypte (suivant le mode **mode**) les données **data** avec le chiffrement **cipher** et la clé **key** en mode CFB et retourne la chaîne résultat.
cipher est une des constantes de type MCRYPT_ciphername.
key est la clé à fournir à l'algorithme. Cette clé doit rester secrète.
data sont les données à crypter/décrypter.
mode vaut MCRYPT_ENCRYPT ou MCRYPT_DECRYPT.
iv est le vecteur d'initialisation.
Voir aussi: mcrypt_cbc(), mcrypt_ecb(), mcrypt_ofb().

10.37.7 mcrypt_ecb

int mcrypt_ecb (int **cipher**, string **key**, string **data**, int **mode**)
mcrypt_ecb() crypte ou décrypte (suivant le mode **mode**) les données **data** avec le chiffrement **cipher** et la clé **key** en mode ECB et retourne la chaîne résultat.
cipher est une des constantes de type MCRYPT_ciphername.
key est la clé à fournir à l'algorithme. Cette clé doit rester secrète.
datasont les données à crypter/décrypter.
mode vaut MCRYPT_ENCRYPT ou MCRYPT_DECRYPT.
Voir aussi: mcrypt_cbc(), mcrypt_cfb(), mcrypt_ofb().

10.37.8 mcrypt_ofb

int mcrypt_ofb (int **cipher**, string **key**, string **data**, int **mode**, string **iv**)
mcrypt_ofb() crypte ou décrypte (suivant le mode **mode**) les données **data** avec le chiffrement **cipher** et la clé **key** en mode OFB et retourne la chaîne résultat.
cipher est une des constantes de type MCRYPT_ciphername.
key est la clé à fournir à l'algorithme. Cette clé doit rester secrète.
datasont les données à crypter/décrypter.
mode vaut MCRYPT_ENCRYPT ou MCRYPT_DECRYPT.
iv est le vecteur d'initialisation.
Voir aussi : mcrypt_cbc(), mcrypt_cfb() et mcrypt_ecb().

10.38 Hash

Ces fonctions ont été prévues pour fonctionner avec mhash.
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 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 vérifications, des signatures de messages, etc...

```
<?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

10.38.1 mhash_get_hash_name

string mhash_get_hash_name (int **hash**)
mhash_get_hash_name() sert à connaitre 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.

```
<?php
```

```
$hash = MHASH_MD5;
print mhash_get_hash_name($hash);
?>
```

L'exemple ci dessus va afficher :
MD5

10.38.2 mhash_get_block_size

int mhash_get_block_size (int **hash**)
mhash_get_block_size() sert à connaitre la taille de bloc du hash specifié **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.

10.38.3 mhash_count

int mhash_count (void)
mhash_count() retourne l'identifiant de hash maximal. Les hash sont numérotés de 0 jusqu'à cet identifiant.

```
<?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));
}
?>
```

10.38.4 mhash

string mhash (int **hash**, string **data**)
mhash() applique la fonction de hash **hash** aux données **data** et retourne le résultat.

10.39 diverses

Ces fonctions ont été placées là, car elles ne rentraient dans aucune catégorie adéquate.

10.39.1 connection_aborted

int connection_aborted (void **)**
Retourne TRUE si le client a abandonné la connexion. Reportez vous à 8.1 Gestion des connexions du chapitre 8 Caractéristiques.

10.39.2 connection_status

int connection_status (void **)**
Retourne les bits de status de la connexion. Reportez vous à la section 8.1 Gestion des connexions pour plus de détails.

10.39.3 connection_timeout

int connection_timeout (void **)**
Retourne TRUE si le script a expiré. Reportez vous à la section 8.1 Gestion des connexions pour plus de détails.

10.39.4 define

int define (string **name**, mixed **value**, int **case_insensitive**)
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 donnée par le paramètre **name**; sa valeur est donnée par **value**.
Le troisième paramètre optionnel **case_insensitive** peut prendre la valeur de *1*. Dans ce cas, le nom de la constante sera insensible à la casse (c'est la valeur par défaut). Cela signifie que, par défaut, CONSTANT et Constant représentent des valeurs différentes.

```
<?php
define ("CONSTANT", "Bonjour le monde.");
echo CONSTANT; // affiche "Bonjour le monde."
?>
```

define() retoune TRUE en cas de succès et FALSE sinon.
Voir aussi defined() et la section sur les 9.2 Les constantes.

10.39.5 defined

int defined (string **name**)
Retourne TRUE si la constante nommée **name** a été définie, et FALSE sinon.
Voir aussi define() et la section sur les 9.2 Les constantes.

10.39.6 die

void die (string **message**)
Cette fonction affiche la chaîne passée en paramètre, puis termine l'exécution du script. Il ne retourne rien de plus.

```
<?php
$filename = '/path/to/data-file';
$file = fopen ($filename, 'r')
or die("impossible d'ouvrir le fichier ($filename)");
?>
```

10.39.7 eval

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.

```
<?php
$string = 'tasse';
$name = 'cafe';
$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 cafe dedans.

10.39.8 exit

void exit
Cette fonction termine l'analyse d'un script en cours d'exécution. Elle ne renvoie aucune valeur.

10.39.9 func_get_arg

int func_get_arg (int **arg_num**)
Retourne l'argument à la **arg_num**-ième position d'une fonction définie par l'utilisateur. Les positions sont indexées en commencant à zéro. func_get_arg() générera une alerte (warning) si elle est appelée hors d'une définition de fonction.

Si **arg_num** est plus grand que le nombre d'arguments une alerte sera générée, et func_get_arg() retournera FALSE.

```
<?php
function foo() {
    $numargs = func_num_args();
echo "Nombre d'arguments: $numargs<br>\n";
if ($numargs >= 2) {
echo "Deuxième argument : " . func_get_arg (1) . "<br>\n";
    }
}
foo (1, 2, 3);
?>
```

func_get_arg() peut être utilisé conjointement avec func_num_args() et func_get_args() pour permettre aux fonctions définies par les utilisateurs d'accepter un nombre variable d'argument.
Note : *Cette fonction a été ajoutée dans PHP 4.*

10.39.10 func_get_args

int func_get_args (void **)**
Retourne un tableau dont les éléments sont les arguments de la fonction courante. func_get_args() générera une alerte (warning) si elle est appelée hors d'une définition de fonction.

```
<?php
function foo() {
   $numargs = func_num_args();
echo "Nombre d'arguments: $numargs<br>\n";
if ($numargs >= 2) {
echo "Deuxième argument : " . func_get_arg (1) . "<br>\n";
   }
   $arg_list = func_get_args();
for ($i = 0; $i < $numargs; $i++) {
echo "Argument $i : " . $arg_list[$i] . "<br>\n";
   }
}
foo (1, 2, 3);
?>
```

func_get_args() peut être utilisé conjointement avec func_num_args() et func_get_args() pour permettre aux fonctions définies par les utilisateurs d'accepter un nombre variable d'argument.
Note : *Cette fonction a été ajoutée dans PHP 4.*

10.39.11 func_num_args

int func_num_args (void **)**
Retourne le nombre d'arguments passé à la fonction courante. func_num_args() génèrera une alerte (warning) si elle est appelée hors d'une définition de fonction.

```
<?php
function foo() {
   $numargs = func_num_args();
echo "Nombre d'arguments: $numargs\n";
}
foo (1, 2, 3);  // affiche : 'Nomber d'arguments: 3'
?>
```

func_num_args() peut être utilisé conjointement avec func_get_arg() et func_get_args() pour permettre aux fonctions définies par les utilisateurs d'accepter un nombre variable d'arguments.
Note : *Cette fonction a été ajoutée dans PHP 4.*

10.39.12 function_exists

int function_exists (string **function_name**)
Vérifie la liste des fonctions définies, et recherche la fonction **function_name**. Retourne TRUE si cette fonction est trouvée, sinon FALSE.

10.39.13 get_browser

object get_browser (string **user_agent**)
get_browser() essaie de determiner 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, and 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 :

```
<?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>
```

crawler:
authenticodeupdate:
msn:

Pour fonctionner, votre configuration 7.1.13 Directives de configuration du navigateur. doit mener au fichier `browscap.ini'.
Pour plus d'informations, (y compris pour les endroits oú charger le fichier `browscap.ini'), suivez la FAQ PHP à http://www.php.net/FAQ.html.
Note : *Browscap a été ajouté dans PHP version 3.0b2.*

10.39.14 ignore_user_abort

int ignore_user_abort (int **setting**)
Cette fonction 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.

10.39.15 iptcparse

array iptcparse (string **iptcblock**)
Analyse un bloc binaire IPTC et recherche les balises simples. Elle retourne un tableau avec les balises comme index, et les valeurs 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.

10.39.16 leak

void leak (int **bytes**)
Leak() crée une fuite de mémoire.
Cette fonction est pratique pour débugger le gestionnaire de mémoire, qui doit nettoyer automatiquement les fuites de mémoire après chaque requête.

10.39.17 pack

string pack (string **format**, mixed **args** ...)
Compacte les arguments dans une chaîne binaire, suivant le format **format**. Retourne la chaîne binaire. L'idée vient du Perl, et tout le formatage fonctionne de la même fonction qu'en Perl, mais quelques formats manquent encore (comme, "u"). La chaîne de format est composée d'une série de code de formats, suivis un quantificateur optionnel. Le quantificateur peut être un entier, ou * pour la répétition indéfinie. Pour a, A, h et H le quantificateur spécifie combien de caractères d'un argument sont pris; pour @, c'est la position absolue oú placer les données, et pour le reste, c'est le nombre de répétitions. Actuellement, les formats suivants sont implémentés :

- une chaîne complétée avec NUL
- 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 (always 32 bits, ordre des bits dépendant de la machine)
- N entier long non signé (toujours 16 bits, ordre des bits big endian)
- V ntier 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 NUL
- X recule d'un octet
- rempli avec NUL, jusqu'à une position absolue

```
$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.

10.39.18 register_shutdown_function

int register_shutdown_function (string **func**)
Enregistre la fonction **func** pour exécution à la fin du script.
Erreur commune :
Etant donné qu'aucun affichage n'est autorisé depuis cette fonction, vous ne pourrez pas débugger en utilisant print ou echo dans cette fonction.

10.39.19 serialize

string serialize (mixed **value**)
serialize() retourne une chaîne contenant un flot d'octets représentant la valeur **value**, et qui pourra être relue.
Ceci est très utile pour enregistrer ou passer des valeurs à PHP sans perdre leur type ou leur structure.
Pour relire la chaîne dans PHP, utilisez unserialize(). serialize() accepte les integer, double, string, array (multidimensionel) et object (les propriétés de l'objet seront sauvegardées, mais pas ses méthodes).

```
// $session_data contient un tableau multi-dimensionnel avec des informations
// de session concernant l'utilisateur courant. On utilise serialize() pour
// enregistrer les informations dans une base de donnée à la fin de la requête.
$conn = odbc_connect("webdb", "php", "poulet");
$stmt = odbc_prepare($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)) {
       /* Un os ! Souffre et potasse. */
   }
}
```

10.39.20 sleep

void sleep (int **seconds**)
La fonction sleep retarde l'exécution du programme pendant **seconds** secondes.
Voir aussi usleep().

10.39.21 uniqid

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 :

```
$token = md5 (uniqid ("")); // pas de section aléatoire.
$better_token = md5 (uniqid (rand())); // mieux, difficile à deviner
```

Ceci va créer un identifiant de 32 caractère (un nombre hexadécimal de 128) qui sera très difficile à prédire.

10.39.22 unpack

array unpack (string **format**, string **data**)
Unpack() déconditionne des données depuis une chaîne binaire avec le format **format**. 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 (/).

```
$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 si il a été déconditionné avec l'option " non signé ".

10.39.23 unserialize

mixed unserialize (string **str**)
unserialize() prend une variable créée avec serialize() et la convertie en valeur PHP. La valeur lue est retournée, et elle peut être de type integer, double, string, array or object. Si un objet a été sérialisé, ses méthodes ont été perdues, et ses attributs ont été conservés.

```
// Ici, on utilise unserialize() pour lire les données concernant
// la session d'un utilisateur dans $session_data. Cet exemple complète
// celui décrit dans  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)) {
   // En cas d'erreur, initialisation à tableau vide.
$session_data = array();
} else {
   // On a maintenant les information dans $tmp[0].
```

```
   $session_data = unserialize($tmp[0]);
if (!is_array($session_data)) {
      // Un os ! initialisation à tableau vide
      $session_data = array();
   }
}
```

10.39.24 usleep

void usleep (int **micro_seconds**)
La sleep() retarde l'exécution du programme pendant **micro_seconds** micro-secondes.
Voir aussi sleep().
Note : *Cette fonction est inopérante sous Windows*

10.40 functions mSQL

10.40.1 msql

int msql (string **database**, string **query**, int **link_identifier**)
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. Si l'identifiant de connexion 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.

10.40.2 msql_affected_rows

int msql_affected_rows (int **query_identifier**)
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().

10.40.3 msql_close

int msql_close (int **link_identifier**)
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().

10.40.4 msql_connect

int msql_connect (string **hostname**)
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().

10.40.5 msql_create_db

int msql_create_db (string **database name**, int **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().

10.40.6 msql_createdb

int msql_createdb (string **database name**, int **link_identifier**)
Identique à msql_create_db().

10.40.7 msql_data_seek

int msql_data_seek (int **query_identifier**, int **row_number**)
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().

10.40.8 msql_dbname

string msql_dbname (int **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 nom de bases disponibles.

10.40.9 msql_drop_db

int msql_drop_db (string **database_name**, int **link_identifier**)
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().

10.40.10 msql_dropdb

Voir msql_drop_db().

10.40.11 msql_error

string msql_error (**)**
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.

10.40.12 msql_fetch_array

int msql_fetch_array (int **query_identifier**, int **result_type**)
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().

10.40.13 msql_fetch_field

object msql_fetch_field (int **query_identifier**, int **field_offset**)
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().

10.40.14 msql_fetch_object

int msql_fetch_object (int **query_identifier**, int **result_type**)
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.
Cette fonction 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().

10.40.15 msql_fetch_row

array msql_fetch_row (int **query_identifier**)
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().

10.40.16 msql_fieldname

string msql_fieldname (int **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**.

10.40.17 msql_field_seek

int msql_field_seek (int **query_identifier**, int **field_offset**)
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().

10.40.18 msql_fieldtable

int msql_fieldtable (int **query_identifier**, int **field**)
Retourne le nom de la table d'ou est le champs **field** a été extrait.

10.40.19 msql_fieldtype

string msql_fieldtype (int **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".

10.40.20 msql_fieldflags

string msql_fieldflags (int **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).

10.40.21 msql_fieldlen

int msql_fieldlen (int **query_identifier**, int **i**)
Msql_fieldlen() retourne la longueur d'un champs.

10.40.22 msql_free_result

int msql_free_result (int **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.

10.40.23 msql_freeresult

Voir msql_free_result()

10.40.24 msql_list_fields

int msql_list_fields (string **database**, string **tablename**)
Msql_list_fields() lit les informations de la table donnée. Les arguments sont le nom de la base de données, et le nom de la table. 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().

10.40.25 msql_listfields

Voir msql_list_fields().

10.40.26 msql_list_dbs

int msql_list_dbs
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.

10.40.27 msql_listdbs

Voir msql_list_dbs().

10.40.28 msql_list_tables

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.

10.40.29 msql_listtables

Voir msql_list_tables().

10.40.30 msql_num_fields

int msql_num_fields (int **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().

10.40.31 msql_num_rows

int msql_num_rows (int **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().

10.40.32 msql_numfields

int msql_numfields (int **query_identifier**)
Identique à msql_num_fields().

10.40.33 msql_numrows

int msql_numrows
Identique à msql_num_rows().

10.40.34 msql_pconnect

int msql_pconnect (string **hostname**)
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, cette fonction 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'.

10.40.35 msql_query

int 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().

10.40.36 msql_regcase

Voir sql_regcase().

10.40.37 msql_result

int msql_result (int **query_identifier**, int **i**, mixed **field**)
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().

10.40.38 msql_select_db

int msql_select_db (string **database_name**, int **link_identifier**)

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().

10.40.39 msql_selectdb

Voir msql_select_db().

10.40.40 msql_tablename

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.

```
<?php
msql_connect ("localhost");
$result = msql_list_tables ("wisconsin");
$i = 0;
while ($i < msql_numrows ($result)) {
   $tb_names[$i] = msql_tablename ($result, $i);
echo $tb_names[$i] . "<BR>";
   $i++;
}
?>
```

10.41 Microsoft SQL Server

10.41.1 mssql_close

int mssql_close (int **link_identifier**)
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().

10.41.2 mssql_connect

int mssql_connect (string **servername** , string **username** , string **password**)
Retourne un identifiant positif de lien en cas de succès, et FALSE sinon.
Mssql_connect() établit une connexion à un serveur MS SQL server. 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(), mssql_close().

10.41.3 mssql_data_seek

int mssql_data_seek (int **result_identifier**, int **row_number**)
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().

10.41.4 mssql_fetch_array

int mssql_fetch_array (int **result**)
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().

10.41.5 mssql_fetch_field

object mssql_fetch_field (int **result**, int **field_offset**)
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().

10.41.6 mssql_fetch_object

int mssql_fetch_object (int **result**)
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().

10.41.7 mssql_fetch_row

array mssql_fetch_row (int **result**)
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().

10.41.8 mssql_field_length

int mssql_field_length (int **result**, int **offset**)

10.41.9 mssql_field_name

int mssql_field_name (int **result**, int **offset**)

10.41.10 mssql_field_seek

int mssql_field_seek (int **result**, int **field_offset**)

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().

10.41.11 mssql_field_type

string mssql_field_type (int **result**, int **offset**)

10.41.12 mssql_free_result

int mssql_free_result (int **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().

10.41.13 mssql_get_last_message

string mssql_get_last_message (void **)**

10.41.14 mssql_min_error_severity

void mssql_min_error_severity (int **severity**)

10.41.15 mssql_min_message_severity

void mssql_min_message_severity (int **severity**)

10.41.16 mssql_num_fields

int mssql_num_fields (int **result**)
Mssql_num_fields() retourne le nombre de champs dans un résultat.
Voir aussi : mssql_query(), mssql_fetch_field() et mssql_num_rows().

10.41.17 mssql_num_rows

int mssql_num_rows (string **result**)
Mssql_num_rows() retourne le nombre de lignes dans un résultat.
Voir aussi : mssql_query() et mssql_fetch_row().

10.41.18 mssql_pconnect

int mssql_pconnect (string **servername** , string **username** , string)
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'.

10.41.19 mssql_query

int mssql_query (string **query**, int **link_identifier**)
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().

10.41.20 mssql_result

int mssql_result (int **result**, int **i**, mixed **field**)
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().

10.41.21 mssql_select_db

int mssql_select_db (string **database_name**, int **link_identifier**)
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 fait dans cette base.
Voir aussi : mssql_connect(), mssql_pconnect() et mssql_query().

10.42 MySQL

Ces fonctions vous permettent d'accéder aux bases de données MySQL.
Plus d'informations sont disponibles à http://www.mysql.com/.

10.42.1 mysql_affected_rows

int mysql_affected_rows (int **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, cette fonction utilise la dernière connexion ouverte.
Si la dernière requête était un DELETE sans clause WHERE, tous les enregistrements ont été effacés, mais cette fonction va retourner 0.
Cette commande n'est pas possible après un SELECT, car elle ne fonctionne qu'après des commandes qui modifient les enregistrements. Pour connaître le nombre de ligne retournées par un SELECT, utilisez mysql_num_rows().

10.42.2 mysql_change_user

int mysql_change_user (string **user**, string **password**, string **database** , int **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 : *Cette fonction a été introduite dans PHP 3.0.13 et requiert MySQL 3.23.3 ou plus récent.*

10.42.3 mysql_close

int mysql_close (int **link_identifier**)
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().

```
<?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().

10.42.4 mysql_connect

int mysql_connect (string **hostname :port :/path/to/socket** , string **username** , string **password**)
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().

```
<?php
   $link = mysql_connect ("kraemer", "marliesle", "secret") {
or die ("Connexion impossible");
   }
print ("Connexion réussie");
mysql_close ($link);
?>
```

Voir aussi mysql_pconnect() et mysql_close().

10.42.5 mysql_create_db

int mysql_create_db (string **database name**, int **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.

```
<?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().

10.42.6 mysql_data_seek

int mysql_data_seek (int **result_identifier**, int **row_number**)
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.

```
<?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);
?>
```

10.42.7 mysql_db_query

int mysql_db_query (string **database**, string **query**, int **link_identifier**)
Retourne un identifiant de résultat si la requête réussi, 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, la fonction prendra par défaut la dernière base de données 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.

10.42.8 mysql_drop_db

int mysql_drop_db (string **database_name**, int **link_identifier**)
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.

10.42.9 mysql_errno

int mysql_errno (int **link_identifier**)
Les erreurs qui sont remontées depuis le serveur MySQL ne sont plus des alertes. A la place, il faut utiliser cette fonction 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().

10.42.10 mysql_error

string mysql_error (int **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().

10.42.11 mysql_fetch_array

array mysql_fetch_array (int **result**, int **result_type**)
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().

```
<?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);
?>
```

10.42.12 mysql_fetch_field

object mysql_fetch_field (int **result**, 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()

10.42.13 mysql_fetch_lengths

array mysql_fetch_lengths (int **result**)
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().

10.42.14 mysql_fetch_object

object mysql_fetch_object (int **result**, int **result_typ**)
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_typ** est une constante qui peut prendre les valeurs suivantes : MYSQL_ASSOC, MYSQL_NUM, et MYSQL_BOTH.
Concernant la vitesse, cette fonction est aussi rapide que mysql_fetch_array(), et presque aussi rapide que mysql_fetch_row() (la différence est insignifiante)

```
<?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().

10.42.15 mysql_fetch_row

array mysql_fetch_row (int **result**)
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().

10.42.16 mysql_field_name

string mysql_field_name (int **result**, int **field_index**)
mysql_field_name() retourne le nom d'une colonne. Les arguments de la fonction sont un identifiant de résultat **result** et l'index du champs, ie.mysql_field_name($result,2);
Retournera le nom du deuxième champs dans le résultat associé à $result.
Pour des raisons de compatibilité ascendante, mysql_fieldname() peut encore être utilisé.

10.42.17 mysql_field_seek

int mysql_field_seek (int **result**, 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().

10.42.18 mysql_field_table

string mysql_field_table (int **result**, int **field_offset**)
Retourne le nom de la table oú se trouve une colonne. Pour des raisons de compatibilité ascendante, mysql_fieldtable() peut encore être utilisé.

10.42.19 mysql_field_type

string mysql_field_type (int **result**, 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.

```
<?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.

10.42.20 mysql_field_flags

string mysql_field_flags (int **result**, 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.

10.42.21 mysql_field_len

int mysql_field_len (int **result**, int **field_offset**)
mysql_field_len() retourne la taille du champs spécifié.
Pour des raisons de compatibilité ascendante, mysql_fieldlen() peut encore être utilisé.

10.42.22 mysql_free_result

int mysql_free_result (int **result**)
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.

10.42.23 mysql_insert_id

int mysql_insert_id (int **link_identifier**)
mysql_insert_id() retourne le dernier identifiant généré par un champs de type AUTO_INCREMENTED. Cette fonction ne prend aucun argument. Elle retourne le dernier identifiant généré par la dernière fonction INSERT effectuée.

10.42.24 mysql_list_fields

int mysql_list_fields (string **database_name**, string **table_name**, int **link_identifier**)
mysql_list_fields() recherche les informations à propos de la table spécifiée. 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 disponble.

10.42.25 mysql_list_dbs

int mysql_list_dbs (int **link_identifier**)
mysql_list_dbs() retournera un identifiant de résultat, qui contiendra les noms des bases de données disponibles sur le serveur MySQL. Utilisez la fonction mysql_tablename() pour lire toutes les bases de données.
Pour des raisons de compatibilité ascendante, mysql_listdbs() est encore disponible.

10.42.26 mysql_list_tables

int mysql_list_tables (string **database**, int **link_identifier**)
mysql_list_tables() prend le nom d'une base de données comme argument, et retourne un identifiant de résultat, qui contiendra la liste des tables. 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 disponble.

10.42.27 mysql_num_fields

int mysql_num_fields (int **result**)
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.

10.42.28 mysql_num_rows

int mysql_num_rows (int **result**)
mysql_num_rows() retourne le nombre de ligne d'une résultat.
Voir aussi mysql_db_query(), mysql_query() et mysql_fetch_row().
Pour des raisons de compatibilité ascendante mysql_numrows() est encore disponible.

10.42.29 mysql_pconnect

int 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 uen 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 dans en version 3.0B4. Le support de ":/path/to/socket" a été ajouté dans en 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'.

10.42.30 mysql_query

int mysql_query (string **query**, int **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:

```
<?php
$result = mysql_query ("SELECT * WHERE 1=1")
or die ("Invalid query");
?>
```

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 :

```
<?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().

10.42.31 mysql_result

int mysql_result (int **result**, int **row**, mixed **field**)
mysql_result() retourne le contenu d'un champs dans un résultat MySQL. L'argument de champs **row** peut être un offset de champs, ou le nom du 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().

10.42.32 mysql_select_db

int mysql_select_db (string **database_name**, int **link_identifier**)
Retourne TRUE en cas de succès, FALSE sinon.
mysql_select_db() change la base de données active sur la connexion représentée par l'identifiant de connexion. 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.

10.42.33 mysql_tablename

string mysql_tablename (int **result**, 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.

```
<?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++;
}
?>
```

10.43 réseau

10.43.1 checkdnsrr

int checkdnsrr (string **host**, string **type**)
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).

10.43.2 closelog

int closelog
closelog() ferme le pointeur qui sert à écrire dans l'historique système. L'utilisation de closelog() est optionnelle.

10.43.3 debugger_off

int debugger_off
Inactive le debuggeur interne de PHP. Le debuggeur est toujours en cours de développement.

10.43.4 debugger_on

int debugger_on (string **address**)
Active le debugger interne de PHP, et le connecte à l'adresse **address**. Le debuggeur est toujours en cours de développement.

10.43.5 fsockopen

int fsockopen (string **hostname**, int **port**, int **errno**, string **errstr**, double **timeout**)
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**. 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 : set_socket_blocking().

```
$fp = fsockopen("www.php.net", 80, &$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);
}
```

Voir aussi pfsockopen()

10.43.6 gethostbyaddr

string gethostbyaddr (string **ip_address**)
Retourne le nom d'hôte correspondant à l'IP **ip_address**. Si une erreur survient, retourne **ip_address**.
Voir aussi gethostbyname().

10.43.7 gethostbyname

string gethostbyname (string **hostname**)
Retourne l'adresse IP correspondant à l'hôte **hostname**.
Voir aussi gethostbyaddr().

10.43.8 gethostbynamel

array gethostbynamel (string **hostname**)
Retourne la liste d'IP correspondant à l'hôte **hostname**.
Voir aussi gethostbyname(), gethostbyaddr(), checkdnsrr(), getmxrr() et la page 8 du manuel.

10.43.9 getmxrr

int getmxrr (string **hostname**, array **mxhosts**, array **weight**)
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.

10.43.10 getprotobyname

int getprotobyname (string **name**)
getprotobyname() retourne le numéro de protocole associé avec le nom de protocole **name**, comme dans `/etc/protocols'.
Voir aussi getprotobynumber().

10.43.11 getprotobynumber

string getprotobynumber (int **number**)
getprotobynumber() retourne le numéro de protocole associé avec le nom de protocole **name**, comme dans `/etc/protocols'.
Voir aussi getprotobyname().

10.43.12 getservbyname

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().

10.43.13 getservbyport

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().

10.43.14 openlog

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 dans la section suivante. 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.
Voir aussi syslog() et closelog().

10.43.15 pfsockopen

int pfsockopen (string **hostname**, int **port**, int **errno**, string **errstr**, int **timeout**)
Cette fonction se comporte exactement comme fsockopen() mais la connexion ouverte le reste, même après la fin du script. C'est la version persistante de fsockopen().

10.43.16 set_socket_blocking

int set_socket_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.

10.43.17 syslog

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.

Pour plus d'informations sur l'historique, reportez vous au manuel Unix (syslog).
Avec Windows NT, l'historique est pris en charge par Event Log.

10.44 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/NISIntroduction to YP/NIS (en anglais). Il existe un livre en anglais Managing NFS and NIS" par Hal Stern.
Pour ajouter ces fonctionnalitées, vous devez compiler PHP avec l'option --with-yp.

10.44.1 yp_get_default_domain

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.

```
<?php
   $domain = yp_get_default_domain();
echo "Le domaine par défaut est : " . $domain;
?>
```

10.44.2 yp_order

int yp_order (string **domain**, string **map**)
yp_order() retourne le numéro d'ordre d'une carte ou FALSE.

```
<?php
   $number = yp_order($domain,$mapname);
echo "Le numéro d'ordre de cette carte est : " . $order;
?>
```

Voir aussi yp-get-default-domain().

10.44.3 yp_master

string yp_master (string **domain**, string **map**)
yp_master() retourne le nom de la machine maître d'une carte.

```
<?php
   $number = yp_master($domain, $mapname);
echo "Master for this map is: " . $master;
?>
```

Voir aussi yp-get-default-domain().

10.44.4 yp_match

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.

```
<?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()

10.44.5 yp_first

string[] yp_first (string **domain**, string **map**)
yp_first() retourne le premier couple (clé ; valeur) d'une carte donnée, ou FALSE.

```
<?php
   $entry = yp_first($domain, "passwd.byname");
   $key = key($entry);
echo "La première entrée de cette carte est " . $key
        . " et sa valeur est " . $entry[$key];
?>
```

Voir aussi yp-get-default-domain()

10.44.6 yp_next

string[] 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.

```
<?php
   $entry = yp_next($domain, "passwd.byname", "joe");
if(!$entry) {
echo yp_errno() . ": " . yp_err_string();
   }
   $key = key($entry);
echo "L'entree suivante après joe a la cle  " . $key
        . " et sa valeur " . $entry[$key];
?>
```

Voir aussi yp-get-default-domain().

10.45 Oracle 8 functions

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.

10.45.1 OCIDefineByName

int OCIDefineByName (int **stmt**, string **Column-Name**, mixed &**variable**, int **type**)
OCIDefineByName() copie les valeurs issues de SQL-Columns dans des variables PHP. Méfiez-vous des colonnes Oracle qui sont toutes en majuscule, tandis que dans les SELECT, vous pouvez aussi les écrire en minuscule. OCIDefineByName() s'attend à ce que **Column-Name** soit en majuscule. 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().

```
<?php
/* Exemple OCIDefineByPos par thies@digicol.de (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);
?>
```

10.45.2 OCIBindByName

int OCIBindByName (int **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).

```
<?php
/* Exemple OCIBindByPos par thies@digicol.de (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);
/* delete our "junk" from the emp table.... */
$stmt = OCIParse($conn,"delete from emp where empno in (1111,2222,3333)");
OCIExecute($stmt);
OCIFreeStatement($stmt);
OCILogoff($conn);
?>
```

10.45.3 OCILogon

int OCILogon (string **username**, string **password**, string **db**)

OCILogon() retourne un identifiant de connexion, nécessaire à la plus part 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 :

```
<?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 values('$conn' || ' ' || to_char(sysdate,'DD-MON-YY
HH24:MI:SS'))");
ociexecute($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().

10.45.4 OCIPLogon

int 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().

10.45.5 OCINLogon

int 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.

```
<?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 values('$conn' || ' ' || to_char(sysdate,'DD-MON-YY
HH24:MI:SS'))");
ociexecute($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().

10.45.6 OCILogOff

int OCILogOff (int **connection**)
OCILogOff() ferme la connexion Oracle.

10.45.7 OCIExecute

int OCIExecute (int **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.

10.45.8 OCICommit

int OCICommit (int **connection**)
OCICommit() valide toutes les transactions en cours sur la connexion Oracle **connection**.

10.45.9 OCIRollback

int OCIRollback (int **connection**)
OCIRollback() annule les transactions en cours sur la connexion Oracle **connection**.

10.45.10 OCINewDescriptor

string OCINewDescriptor (int **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.

```
<?php
   /* Ce script est fait pour être appelé dans un formulaire HTML
    * Il attends les variables $user, $password, $table, $where, et $commitsize
    * Le scrip efface alors les lignes selectionnées 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 attend les variables $user, $password, $table, $where, et $commitsize,
    * données par le formulaire. Le script efface
    * les lignes selectionnées 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 $table (id, the_blob) values(my_seq.NEXTVAL, EMPTY_BLOB())
returning the_blob into :the_blob");
OCIBindByName($stmt, ':the_blob', &$lob, -1, OCI_B_BLOB);
OCIExecute($stmt);
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);
 }
?>
```

10.45.11 OCIRowCount

int OCIRowCount (int **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.

```
<?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>";
?>
```

10.45.12 OCINumCols

int OCINumCols (int **stmt**)
OCINumCols() retourne le nombre de colonnes dans un résultat

```
<?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";
?>
```

10.45.13 OCIResult

mixed OCIResult (int **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).

10.45.14 OCIFetch

int OCIFetch (int **statement**)

OCIFetch() place la prochaîne ligne (d'une commande SELECT) dans le pointeur interne de résultat.

10.45.15 OCIFetchInto

int OCIFetchInto (int **stmt**, array &**result**, int **mode**)
OCIFetchInto() retourne la ligne suivante (pour une commande SELECT) dans le tableau **result**.
OCIFetchInto() crasera le contenu de **result**. Par défaut, **result** sera un tableau à index numérique, commencant 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 (ie 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.

10.45.16 OCIFetchStatement

int OCIFetchStatement (int **stmt**, array &**variable**)
OCIFetchStatement() retourne toutes les lignes d'un résultat dans le tableau variable.
OCIFetchStatement() retourne le nombre de lignes retournées.

```
<?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);
?>
```

10.45.17 OCIColumnIsNULL

int OCIColumnIsNULL (int **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**.

10.45.18 OCIColumnSize

int OCIColumnSize (int **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**.

```
<?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().

10.45.19 OCIServerVersion

string OCIServerVersion (int **conn**)
OCIServerVersion() retourne une chaîne contenant les informations de version du serveur

```
<?php
  $conn = OCILogon("scott","tiger");
print "Version du serveur : " . OCIServerVersion($conn);
OCILogOff($conn);
?>
```

10.45.20 OCIStatementType

string OCIStatementType (int **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"

```
<?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>";
?>
```

10.45.21 OCINewCursor

int OCINewCursor (int **conn**)
OCINewCursor() alloue un nouveau pointeur de commande, pour la connexion **conn**.

```
<?php
// suppose your stored procedure info.output returns a ref cursor in :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($data);
}
OCIFreeCursor($stmt);
OCIFreeStatement($curs);
OCILogoff($conn);
?>
```

```
<?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);
?>
```

10.45.22 OCIFreeStatement

int OCIFreeStatement (int **stmt**)
OCIFreeStatement() retourne TRUE en cas de succès, et FALSE en cas d'échec.

10.45.23 OCIFreeCursor

int OCIFreeCursor (int **stmt**)
OCIFreeCursor() retourne TRUE en cas de succès, et FALSE en cas d'échec.

10.45.24 OCIColumnName

string OCIColumnName (int **stmt**, int **col**)
OCIColumnName() retourne le nom de la colonne correspondant au numéro de colonne passé en paramètre (l'indexation des colonnes commence à 1).

```
<?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().

10.45.25 OCIColumnType

mixed OCIColumnName (int **stmt**, int **col**)
OCIColumnType() retourne le type de données de la colonne correspondant au numéro de colonne (les colonnes sont indexées à partir de 1).

```
<?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().

10.45.26 OCIParse

int OCIParse (int **conn**, strint **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.

10.45.27 OCIError

int 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.

10.45.28 OCIInternalDebug

void OCIInternalDebug (int **onoff**)
OCIInternalDebug() active l'affichage des informations de debuggage. Pour les afficher, mettez onoff à 0, ou sinon à 1 pour les cacher.

10.46 Oracle

10.46.1 Ora_Bind

int ora_bind (int **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 sorties, 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";
?>
```

10.46.2 Ora_Close

int ora_close (int **cursor**)
Retourne TRUE si la fermeture a bien eu lieu, et FALSE sinon. Les erreurs sont accessibles avec les fonctions ora_error() et ora_errorcode().
Cette fonction termine les pointeurs ouverts avec la fonction ora_open().

10.46.3 Ora_ColumnName

string Ora_ColumnName (int **cursor**, int **column**)
Ora_ColumnName() retourne le nom du champs **column** du pointeur **cursor**. Le nom retourné sera en majuscule.

10.46.4 Ora_ColumnType

string Ora_ColumnType (int **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"

10.46.5 Ora_Commit

int ora_commit (int **conn**)
Retourne TRUE si la validation a bien eu lieu, et FALSE sinon. Les erreurs sont accessibles avec les fonctions ora_error() et ora_errorcode().
Cette fonction 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.

10.46.6 Ora_CommitOff

int ora_commitoff (int **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().
Inactive la validation automatique après chaque ora_exec().

10.46.7 Ora_CommitOn

int ora_commiton (int **conn**)
ora_commiton() active la validation automatique après chaque ora_exec().
Retourne TRUE si l'activation a bien eu lieu, et FALSE sinon. Les erreurs sont accessibles avec les fonctions ora_error() et ora_errorcode().

10.46.8 Ora_Error

string Ora_Error (int **cursor_or_connection**)
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'erreurs 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 .

10.46.9 Ora_ErrorCode

int Ora_ErrorCode (int **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.*

10.46.10 Ora_Exec

int ora_exec (int **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().

10.46.11 Ora_Fetch

int ora_fetch (int **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.

10.46.12 Ora_GetColumn

mixed ora_getcolumn (int **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").

10.46.13 Ora_Logoff

int ora_logoff (int **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().
Déconnecte l'utilisateur, et se déconnecte.

10.46.14 Ora_Logon

int 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 :
$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.
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().

10.46.15 Ora_Open

int ora_open (int **connection**)
ora_open() ouvre un pointeur Oracle sur la connexion.
Retourne TRUE si la fermeture a bien eu lieu, et FALSE sinon. Les erreurs sont accessibles avec les fonctions ora_error() et ora_errorcode().

10.46.16 Ora_Parse

int ora_parse (int **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.

10.46.17 Ora_Rollback

int ora_rollback (int **connection**)
ora_rollback() annule une transaction Oracle. (Voir ora_commit() pour la définition d'une transaction).
Retourne TRUE si la fermeture a bien eu lieu, et FALSE sinon. Les erreurs sont accessibles avec les fonctions ora_error() et ora_errorcode().

10.47 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 alphanumérique ou n'est pas un backslash (\). Si un délimiteur doit être utilisé dans l'expression, il faudra l'échapper avec un backslash.
Le délimiteur final peut être suivi d'options qui affecteront la recherche. 10.47.7 Options de recherche.

- /<\/\w+>/
- |(\d{3})-\d+|Sm
- /^(?i)php[34]/
- /href='(.*)' - délimiteur final manquant
- /\w+\s*\w+/J - option 'J' inconnue

- 1-\d3-\d3-\d4| - délimiteur initial manquant

10.47.1 preg_match

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..
Retourne TRUE si la recherche à réussie, et FALSE sinon (notamment en cas d'erreur).

```
if (preg_match("/page\s+#(\d+)/i", "Go to page #9.", $parts))
print "La page suivante est $parts[1]";
else
print "Page introuvable.";
```

Voir aussi preg_match_all(), preg_replace() et preg_split().

10.47.2 preg_match_all

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..
- preg_match_all("|<[^>]+>(.*)</[^>]+>|U", "example: a test ", $out, PREG_PATTERN_ORDER);
- print $out[0][0].", ".$out[0][1]."\n";
- print $out[1][0].", ".$out[1][1]."\n"
  Cet exemple va afficher :
  example: , >div align=left>ceci est un test
  example: , this is a 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...
- preg_match_all("|<[^>]+>(.*)</[^>]+>|U", "exemple: a test ", $out, PREG_SET_ORDER);
- print $out[0][0].", ".$out[0][1]."\n";
- print $out[1][0].", ".$out[1][1]."\n"

Cet exemple va afficher :
example: , exemple:
this is a test , this is a 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, etcÖ. De même, $matches[1] contient le texte qui satisfait le masque complet, etc. *

Si **order** est omis, PREG_PATTERN_ORDER est utilisé par défaut.
Retourne le nombre de résultat qui satisfont le masque complet, ou FALSE en cas d'échec ou d'erreur.

```
preg_match_all("/\(?  (\d{3})?  \)?  (?(1)  [\-\s] ) \d{3}-\d{4}/x",
            "Call 555-1212 or 1-800-555-1212", $phones);
```

Voir aussi preg_match(), preg_replace() et preg_split().

10.47.3 preg_replace

mixed preg_replace (mixed **pattern**, mixed **replacement**, mixed **subject**)
preg_replace() analyse **subject** pour trouver l'expression **pattern** et remplace les résultats par **replacement**.
replacement peut contenir des références de la forme *n*. 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 correspond au texte de qui satisfait le masque complet. Les parenthèses ouvrantes sont comptées de gauche à droit (en commencant 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** a 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. Note : *Cette option a été ajoutée en PHP 4.0.*

```
$patterns = array("/(19|20\d{2})-(\d{1,2})-(\d{1,2})/", "/^\s*{(\w+)}\s*=/");
$replace = array("\\3/\\4/\\1", "$\\1 =");
print preg_replace($patterns, $replace, "{startDate} = 1999-5-27");
```

Cet exemple va afficher :
    $startDate = 5/27/1999

```
preg_replace("/(<\/?)(\w+)([^>]*>)/e", "'\\1'.strtoupper('\\2').'\\3'", $html_body);
```

Cela va mettre en majuscule toutes les balises HTML du texte.

Voir aussi preg_match(), preg_match_all() et preg_split().

10.47.4 preg_split

array preg_split (string **pattern**, string **subject**, int **limit**, int **flags**)
Note : *Le paramètre **flags** a été ajouté dans PHP Beta 3.*
Retourne un tableau contenant les sous chaînes de **subject**, séparées par les chaînes qui vérifient **pattern**.
Si **limit** est donné, seules les **limit** premières chaînes seront retournées.
Si le flag est PREG_SPLIT_NO_EMPTY, alors seul les sous chaînes non nulles seront retournées.

```
$keywords = preg_split("/[\s,]+/", "hypertext language, programming");
```

Voir aussi preg_match(), preg_match_all() et preg_replace().

10.47.5 preg_quote

string preg_quote (string **str**)
preg_quote() ajoute un backslash 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.
Les caractères spéciaux qui seront échappés :
. \\ + * ? [^] $ ( ) { } = ! < > | :
Note : *Cette fonction a été ajoutée dans PHP 3.0.9.*

10.47.6 preg_grep

array preg_grep (string **pattern**, array **input**)
preg_grep() retourne un tableau qui contient les éléments de **input**qui satisfont le masque **pattern**.

preg_grep("/^(\d+)?\.\d+$/", $array); // recherche les nombres à virgule flottante

Note : *Cette fonction a été ajoutée dans PHP 4.0.*

10.47.7 Options de recherche

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 E ne soit
- mise). C'est le même fonctionnement qu'en Perl.

- Lorsque cette option est mise, " 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.
- Si il n'y a pas de caractère de nouvelle ligne "\n" dans la chaîne sujet,
- ou si 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 caractère 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 caractère, et
- tous les caractères entre # non échappés et en dehors d'une classe de
- caractère, 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 \\ 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.
- Note :
- *Cette option a été ajoutée dans PHP 4.0.*

- *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 mise. 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 mise, 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 backslash suivis d'une lettre qui n'aurai pas de signification
- particulière cause une erreur, permettant la réservation de ces combinaisons
- pour des ajouts fonctionnels ultérieurs. Par défaut, comme en Perl, les
- backslash suivi d'une lettre sans signification particulière sont traités
- comme des valeurs litérales. Actuellement, cette option ne déclenche pas
- d'autres fonctions.
-

-

10.47.8 Syntaxe des masques

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 comportement 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) 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) Si PCRE_DOLLAR_ENDONLY est mis, et que PCRE_MULTILINE n'est pas mis, le méta caractère $ ne s'applique qu'à la fin physqiue de la chaîne, et non pas avant les caractères de nouvelle ligne.
(c) Si PCRE_EXTRA est mis, un backslash suivi d'une lettre sans signification spéciale est considérée comme une erreur.
(d) SI PCRE_UNGREEDY est mis, la "gourmandise" des quantificateurs de répétition est inversées, ce qui est rend non gourmand par défaut, mais si ils sont suivis de ?, il seront gourmands.
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, qui est appliqué sur une chaîne sujet, de gauche à droite. La plus part 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.
La puissance des expressions régulières provient de leur capacité à autoriser des alternatives et des quantificateur de répétitions dans le masque. Ils sont encodés dans le masque par des *meta-characters*,

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 oú dans un masque, hormis entre crochets, et ceux qui sont reconnus entre crochets. A l'extérieur des crochets, les méta caractères sont :

\ Caractère d'échappement, avec de multiples usages.
^ Le début de la chaîne sujet (ou de ligne, en mode multiligne)
$ La fin de la chaîne sujet (ou de ligne, en mode multiligne)
. Remplace n'importe quel caractère, hormis le caractère de nouvelle ligne (par défaut) ;
[Caractère de début de définition de classe
] Caractère de fin de définition de classe
| Caractère de début d'alternative
(Caractère de début de sous masque
) Caractère de fin de sous masque
? Etend le sens de (

mais aussi quantificateur de 0 ou 1
mais aussi quantificateur de minimisation

* Quantificateur de 0 ou plus
+ Quantificateur de 1 ou plus
{ Caractère de début 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 seul méta caractères autorisés sont

\ Caractère d'échappement, avec de multiples usages
^ négation de la classe, mais uniquement si placé tout au début de la classe
- indique un intervalle de caractères
] termine la classe de caractères

La section suivante décrit l'utilisation de chaque méta caractères :
BACKSLASH
Le caractère backslash a de nombreuses utilisations. En premier lieu, si il est suivi d'un caractère non alpha numérique, il ne prendra pas la signification spéciale qui y est rattachée. Cette utilisation du backslash 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 sur pour s'assurer qu'un caractère sera recherché pour sa valeur litérale, plutot que pour sa valeur spéciale. En particulier, pour rechercher les backslash, 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. Le backslash peut être utilisé pour échapper et ainsi rechercher un espace ou un dièse.
La deuxième utilité du backslash 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 plutot 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 si il y a déjà eu au moins autant de parenthèses ouvrantes auparavant, la séquence est prise pour une *référence de retour*. 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 du backslash, et génére 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 est une autre manière d'écrire un espace

\40 est identique, dans la mesure oú il n'y a pas 40 parenthèses ouvrantes auparavant.

\7 est toujours une référence de retour.

\11 peut être une référence de retour, ou une tabulation, tandis que \011 est toujours une tabulation

\011 est 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 référence de arrière)

\377 est un octet dont tous les bits sont à 1

\81 peut être soit une référence de arrière, soit le caractère NULL, suivi des caractères "8" et "1"

Les valeurs octales supérieures ou égales à 100 ne doivent pas être introduite 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 caractère, 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 backslash 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 mis 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 du backslash 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 backslash 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 caractère (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 oú 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.

CIRCUMFLEX et DOLLAR

En dehors d'une classe de caractère, 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 caractère, ^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 caractère, ^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 CRE_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 ligne. 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.

FULL STOP (PERIOD, DOT)
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.

SQUARE BRACKETS
Un crochet ouvrant introduit une classe de caractère, 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 backslash.
Une classe de caractère remplace un seul caractère dans la chaîne sujet, à moins que le premier caractère de la classe soit un ^, 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 backslash.
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 caractère représentent en même temps la majuscule et la minuscule. Par exemple,

[aeiou] représentera "A" ou "a", et [^aeiou] n'acceptera pas "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 caractère, 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 backslash, 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 "]" comme fin d'intervalle. Un masque tel que [W-]46] est compris comme la classe de caractère 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 backslash, le masque [W-\]46] est interprété comme une classe d'un seul caractère, contenant un intervalle de caractère. La valeur octale ou hexadécimale de "]" peuvent aussi être utilisée pour déterminer les limites de l'intervalle.
Les intervalles travaillent sur des séquences ASCII. Elles 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, [W-c] est équivalent é [][\^_`wxyzabc], avec l'option d'insensibilité de casse.
Si la table locale de caractère est "fr", [\xc8-\xcb] correspond aux caractères accentués.

Les types de caractères \d, \D, \s, \S, \w, et \W peuvent aussi intervenir dans les classes de caractères. Par exemple, [\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 alphanumériques autres que \, -, ^ (placé en début de chaîne) et] n'ont pas de significations particulière, mais ils ne perdront rien à être échappés.

VERTICAL BAR
La barre verticale sert à séparer des alternatives. Par exemple, dans le masque dupont|martin
recherche soit "dupont", soit " martin ". Le nombre d'alternative 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 droit, 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.

INTERNAL OPTION SETTING
Les options PCRE_CASELESS, PCRE_MULTILINE, PCRE_DOTALL, et PCRE_EXTENDED peuvent
être changée depuis le masque lui-même, avec des séquences mises "(?" et ")".
Les options sont
i pour PCRE_CASELESS
m pour PCRE_MULTILINE
s pour PCRE_DOTALL
x pour 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, mettre 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. Si il y a plusieurs séquences d'option qui portent sur la même optin, 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'option 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 la mettre au début du masque.

SUBPATTERNS
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 cat(aract|erpillar|) acceptera les mots "char", "chardon", ou "charmant". Sans les parenthèses, il n'accepterait que "chardon", "mant" ou la chaîne vide.
2. Le sous masque est considéré comme capturante : 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, (commencant 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 oú 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 pattern
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)
(?:(?i) samedi | dimanche)
De plus, comme les séquences d'options sont valables sur toute une alternative, le masque ci dessus acceptera aussi "DIMANCHE" que "Dimanche".

REPETITION
les Répétitions sont spécifiées avec des quantificateurs, qui peuvent être placés à la suite des caractères suivants :
Un caractère unique, même s'il s'agit d'un méta caractère
Une classe de caractères
Une référence de retour (Voir section suivante)
Un sous masque avec parenthèses (a 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 petit 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 :

```
[aeiou]{3,}
```

accepte n'importe quelle succession d'au moins 3 voyelles minuscules, tandis que

```
\d{8}
```

n'accepte que 8 chiffres exactements. Une accolade ouvrante qui apparaît à une position oú un quantificateur n'est pas accepté, ou si la syntaxe des quantificateurs n'est pas respectée, elle 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 oú 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 "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 commentaire d'un programme en C. Les commentaires apparaissent entre les séquences /* et */ 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 oú 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 mise, (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 répétition, qui soit plus grand que 1, ou avec un maximum de répétition, 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 mise, 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 oú on sait d'avance qu'il n'y aura pas de caractère de nouvelle ligne, mettre 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, si 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 Références arrières (back references)
En dehors des classes de caractères, un backslash 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 capturant précédent.
Cependant, si le nombre décimal suivant le backslash est plus petit que 10, il sera toujours considéré comme une référence arrière, et cela générera 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 "Backslash" pour avoir de plus amples détails à propos du nombre de chiffres qui suivent le backslash.
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 \1menttrouvera "calme et calmement " et "rapide et rapidement ", mais pas " calme et rapidement ". Si la recherche tiens 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 le backslash 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 mise, 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.

Les 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 décrite précédemment. Il existe cependant un type d'assertion plus complexe, codées sous la forme de sous masques. Il en existe deux types : ceux qui travaille au dela de la position courante, et ceux qui travaille 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 des assertions négatives. Par exemple :
foo(?!bar)
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 "foo"; 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.
Ces assertions 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 tel façon que les chaînes qu'il utilise sont toujours de la même taille. Cependant, lorsqu'il y a plusieurs alternatives, elles n'ont pas besoin d'être de la même taille.
(?<=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 caractère 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
(?<=\d{3})(?<!999)foo
recherche les chaînes "foo" précédée par trois chiffres qui 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".
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 (ONCE-ONLY SUBPATTERNS)
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 fait 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, commencant avec (?>comme ceci :
(?>\d+)bar
Après avoir tenté d'utiliser les 6 chiffres suivi de "foo" qui fait 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, commencant 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 performance et de temps de recherche.
Les sous masques conditionnels (CONDITIONAL SUBPATTERNS)
Il est possible de lier un sous masque à un 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)
(?(condition) masque positif | masque négatif)
Si les conditions sont satisfaites, le masque positif est utilisé, sinon, le masque négatif est utilisé, si présent. Si 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 chiffre, alors la condition est satisfaite si le sous masque correspondant à ce numéro a réussi. Considérons le masque suivant, qui contient des espaces non significatif pour le rendre plus compréhensible (on supposera l'option PCRE_EXTENDED mise) 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 contient pas de parenthèses. La troisième partie est conditionnée à la première, et s'assure que si 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 lettre, éventuellement mis entre parenthèse.
Si la condition n'est pas une séquence de nombre, 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 caractère 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 chiffresCOMMENTS
La séquence (?# marque le début des commentaires, qui se termine à la prochaîne 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 mise, les caractères dièses # non échappés en dehors d'une classe de caractères introduisent un commentaire qui continuera jusqu'à la prochaîne ligne dans le masque.
PERFORMANCE
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 généralement 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 mise, 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 mise, PCRE ne peut faire aucune optimisation ca 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.

10.48 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;, the TIFF library, pour compiler cette librairie.
Si vous utilisez pdflib 2.01, vérifiez l'installation de la librairie. Il ne devrait y avoir qu'un fichier (ou lien) : libPDF.so. La version 2.01 ne fait que créer une librairie appelée libpdf2.01.so, qui ne peut être trouvée lors du link du programme de test, lors de la configuration. Il vous faudra faire ce lien vous même, de libPDF.so vers libpdf2.01.so.
La version 2.20 de pdflib a introduit beaucoup de modifications dans ses API pour supporter les polices chinoises et japonaises. Cela modifie malheureusement les API du module PHP4 (mais pas le 3). Utilisez pdflib 2.20 qui gère correctement la mémoire lors de la génération de fichier PDF. Jusqu'à la publication de la version 3.0, pdflib peut être un peu instable. Le paramètre d'encodage de pdf_set_font() a été changé en chaîne. Ainsi, à la place de '4', vous devrez utiliser 'winansi'.
Si vous utilisez pdflib 2.30 la fonction pdf_set_text_matrix() a disparu. Elle n'est plus supportée. En général, c'est une bonne idée de consulter les notices de publications, pour connaître les modifications qui sont intervenues.
Reportez vous à l'excellente documentation de pdflib, disponible avec la distribution de pdflib ou à : http://www.ifconnection.de/~tm/software/pdflib/PDFlib-0.6.PDF. C'est une introduction très pratique des capacités de pdflib. La plus part des 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's ClibPDF. Les API sont légèrement différentes. Reportez vous à la section 10.9 ClibPDF pour plus de détails.
Actuellement, toutes les versions de la bibliothèque pdflib sont supportées. Il est préférable d'utiliser la nouvelle version, étant donné qu'elle comporte moins de bugs, et propose plus de fonctions que l'ancienne version. Malheureusement, les modifications dans les API de la bibliothèque ont été si importantes que certaines fonctions PHP ont dues être modifiées. Voici une petite liste des changements :

- La structure Info n'existe plus. Par conséquent, pdf_get_info() est obsolète et les fonctions pdf_set_info_creator(), pdf_set_info_title(), pdf_set_info_author(), pdf_set_info_subject() et pdf_set_info_keywords() ne prennent plus de structure info comme premier paramètre, mais le document PDF. Cela signifie aussi que le document PDF doit être ouvert pour pouvoir appeler ces fonctions.

- La manière d'ouvrir un document a changé. La fonction pdf_open() ne prend plus qu'un paramètre, qui est le pointeur de fichier, obtenu avec la fonction fopen().

Il y a aussi quelques nouveautés avec la version 2.01 de pdflib qui devrait être couvertes par PHP. Quelques fonctions ne sont plus utilisées (e.g. pdf_put_image()). Ne soyez pas surpris d'obtenir une alerte...
Le module PDF introduit deux nouveaux types de variables (avec pdflib 2.x, seulement un nouveau type) C'est pdfdoc et pdfinfo (pdfinfo n'existe pas avec pdflib 2.x. pdfdoc est un pointeur de document PDF, et presque toutes les fonctions en ont besoin comme premier paramètre. pdfinfo contient des méta données à propos du document PDF document. Il doit exister avant d'appeler (**pdfinfo** n'existe pas si pdflib 2.x est utilisé. **pdfdoc** est un pointeur sur un document PDF et presque toutes les fonctions le requière comme premier paramètre. **pdfinfo** contient des méta-données sur les documents PDF. Elles doivent être fixée avant d'utiliser pdf_open().
Note : *La suite n'est vraie qu'avec pdflib 0.6. Lisez le manuel de pdflib pour les nouveautés des nouvelles versions.*

Pour sauver des données dans un fichier PDF, vous devez fournir un fichier de type afm pour chaque police de caractère. Les fichiers Afm contiennent les définitions des polices Postscript. Par défaut, ces fichiers afm sont recherché dans un dossier nommé 'fonts', près du dossier qui contient le script éxécuté. (Encore une fois, ceci était vrai avec pdflib 0.6, mais ne le sera pas nécessairement avec les nouvelles versions).
La plupart des fonctions sont plutôt simples à utiliser. Le plus difficile est probablement la création d'un document PDF simple. Les exemples suivant vous aideront à repérer les premiers éléments. Ils utilisent les fonctions accessibles avec pdflib 0.6. Ils créent un exemple de test, test.pdf, d'une seule page. La page contient le texte "Times-Roman" en police renforcée, souligné, de 30 pt.

```
<?php
$fp = fopen("test.pdf", "w");
$info = PDF_get_info();
pdf_set_info_author($info, "Uwe Steinmann");
PDF_set_info_title($info, "Test for PHP wrapper of PDFlib 0.6");
PDF_set_info_author($info, "Name of Author");
pdf_set_info_creator($info, "See Author");
pdf_set_info_subject($info, "Testing");
$pdf = PDF_open($fp, $info);
PDF_begin_page($pdf, 595, 842);
PDF_add_outline($pdf, "Page 1");
pdf_set_font($pdf, "Times-Roman", 30, 4);
pdf_set_text_rendering($pdf, 1);
PDF_show_xy($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($pdf);
fclose($fp);
echo "<A HREF=getPDF.php3>finished</A>";
?>
```

Le script getPDF.php3 ne fait que produire un fichier PDF.

```
<?php
$fp = fopen("test.pdf", "r");
header("Content-type: application/pdf");
fpassthru($fp);
fclose($fp);
?>
```

La même chose avec pdflib 2.x ressemble à ce qui suit :

```
<?php
$fp = fopen("test.pdf", "w");
$pdf = PDF_open($fp);
pdf_set_info_author($pdf, "Uwe Steinmann");
PDF_set_info_title($pdf, "Test for PHP wrapper of PDFlib 2.0");
PDF_set_info_author($pdf, "Name of Author");
pdf_set_info_creator($pdf, "See Author");
pdf_set_info_subject($pdf, "Testing");
PDF_begin_page($pdf, 595, 842);
PDF_add_outline($pdf, "Page 1");
pdf_set_font($pdf, "Times-Roman", 30, 4);
pdf_set_text_rendering($pdf, 1);
PDF_show_xy($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($pdf);
fclose($fp);
echo "<A HREF=getPDF.php3>finished</A>";
```

```
?>
```

Ce script est identique à celui ci-dessus.

La distribution de pdflib contient d'autres exemples plus élaborés, qui crée des pages plus consistantes. Cet exemple convertit en PHP, et en utilisant pdflib 2.x ressemble à ce qui suit : (Vous pouvez retrouver cet exemple dans la documentation de 10.9 ClibPDF:

```
<?php
$pdffilename = "clock.pdf";
$radius = 200;
$margin = 20;
$pagecount = 40;
$fp = fopen($pdffilename, "w");
$pdf = pdf_open($fp);
pdf_set_info_creator($pdf, "pdf_clock.php3");
pdf_set_info_author($pdf, "Uwe Steinmann");
pdf_set_info_title($pdf, "Analog Clock");
while($pagecount-- > 0) {
pdf_begin_page($pdf, 2 * ($radius + $margin), 2 * ($radius + $margin));
pdf_set_transition($pdf, 4);  /* wipe */
pdf_set_duration($pdf, 0.5);
pdf_translate($pdf, $radius + $margin, $radius + $margin);
pdf_save($pdf);
pdf_setrgbcolor($pdf, 0.0, 0.0, 1.0);
   /* minutes */
pdf_setlinewidth($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($pdf);
   /* 5 minutes */
pdf_setlinewidth($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);
   }
   $ltime = getdate();
   /* Dessine les heures */
pdf_save($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);
   /* draw minute hand */
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);
```

```
    /* draw second hand */
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($pdf);
    /* draw little circle at center */
pdf_circle($pdf, 0, 0, $radius/30);
pdf_fill($pdf);
pdf_restore($pdf);
pdf_end_page($pdf);
}
$pdf = pdf_close($pdf);
fclose($fp);
echo "<A HREF=getPDF.php3?filename=".$pdffilename.">finished</A>";
?>
```

Le script getPDF.php3 ne fait qu'afficher le fichier PDF.

```
<?php
$fp = fopen($filename, "r");
header("Content-type: application/pdf");
fpassthru($fp);
fclose($fp);
?>
```

10.48.1 PDF_get_info

info pdf_get_info (string **filename**)
PDF_get_info() retourne une structure d'information de document PDF. Cette structure est la structure par défaut. Elle devra être remplie avec des informations adéquates, telle que l'auteur, le sujet etc.... Note : *Cette fonction n'est pas disponible si pdflib 2.x est activée.*

Voir aussi PDF_set_info_creator(), PDF_set_info_author(), PDF_set_info_keywords(), PDF_set_info_title(), PDF_set_info_subject().

10.48.2 PDF_set_info_creator

void pdf_set_info_creator (info **info**, string **creator**)
PDF_set_info_creator() affecte le champs titre de la structure info d'un document PDF. Cette fonction doit être appelée après PDF_get_info() et avant PDF_open(). L'appeler après PDF_open() sera sans effet sur le document. Note : *Cette fonction ne fait pas partie de PDF library.*
Note : *Cette fonction prend un autre type de premier paramètre si pdflib 2.x est activé. Le premier paramètre doit alors être un identifiant de document PDF document, retourné par pdf_open(). Par conséquent, pdf_open() doit avoir été appelé avant cette fonction.*

Voir aussi PDF_get_info(), PDF_set_info_keywords(), PDF_set_info_title(), PDF_set_info_subject().

10.48.3 PDF_set_info_title

void pdf_set_info_title (info **info**, string **title**)
PDF_set_info_title() modifie le sujet d'un document PDF. Cette fonction doit être appelée après PDF_get_info() mais avant PDF_open(). Si vous l'appelez après PDF_open(), elle n'aura pas d'effet sur le document. Note : *Cette fonction ne fait pas partie de PDF library.*
Note : *Cette fonction prend un autre type de premier paramètre si pdflib 2.x est activé. Le premier paramètre doit alors être un identifiant de document PDF document, retourné par pdf_open(). Par*

conséquent, pdf_open() doit avoir été appelé avant cette fonction.

Voir aussi PDF_get_info(), PDF_set_info_creator(), PDF_set_info_author(), PDF_set_info_keywords(), PDF_set_info_subject().

10.48.4 PDF_set_info_subject

void pdf_set_info_subject (info **info**, string **subject**)
PDF_set_info_subject() modifie le sujet d'un document PDF. Cette fonction doit être appelée après PDF_get_info() et avant PDF_open(), sinon, elle sera sans effet. Note : *Cette fonction ne fait pas partie de PDF library.*
Note : *Cette fonction prend un autre type de premier paramètre si pdflib 2.x est activé. Le premier paramètre doit alors être un identifiant de document PDF document, retourné par pdf_open(). Par conséquent, pdf_open() doit avoir été appelé avant cette fonction.*

Voir aussi PDF_get_info(), PDF_set_info_creator(), PDF_set_info_author(), PDF_set_info_title(), PDF_set_info_keywords().

10.48.5 PDF_set_info_keywords

void pdf_set_info_keywords (info **info**, string **keywords**)
PDF_set_info_subject() affecte le champs mots-clé de la structure d'un document PDF. Cette fonction doit être appelée après PDF_get_info() et avant PDF_open(), sinon, elle sera sans effet.
Note : *Cette fonction ne fait pas partie de PDF library.*
Note : *Cette fonction prend un autre type de premier paramètre si pdflib 2.x est activé. Le premier paramètre doit alors être un identifiant de document PDF document, retourné par pdf_open(). Par conséquent, pdf_open() doit avoir été appelé avant cette fonction.*

Voir aussi PDF_get_info(), PDF_set_info_creator(), PDF_set_info_author(), PDF_set_info_title(), PDF_set_info_subject().

10.48.6 PDF_set_info_author

void pdf_set_info_author (info **info**, string **author**)
PDF_set_info_author() affecte le champs auteur de la structure d'un document PDF. Cette fonction doit être appelée après PDF_get_info() et avant PDF_open(), sinon, elle sera sans effet. Note : *Cette fonction ne fait pas partie de PDF library.*
Note : *Cette fonction prend un autre type de premier paramètre si pdflib 2.x est activé. Le premier paramètre doit alors être un identifiant de document PDF document, retourné par pdf_open(). Par conséquent, pdf_open() doit avoir été appelé avant cette fonction.*

Voir aussi PDF_get_info(), PDF_set_info_creator(), PDF_set_info_keywords(), PDF_set_info_title(), PDF_set_info_subject().

10.48.7 PDF_open

int pdf_open (int **file**, int **info**)
PDF_open() ouvre un nouveau document PDF. Le fichier correspondant doit avoir été ouvert avec fopen() et le pointeur de fichier est passé en argument **file**. **info** est une structure info créée par pdf_get_info(). Cette structure info sera effacée à partir de cette fonction. Note : *Cette fonction ne fait pas partie de PDF library.*
Note : *Cette fonction prend un autre type de premier paramètre si pdflib 2.x est activé. Le premier paramètre doit alors être un identifiant de document PDF document, retourné par pdf_open(). Par conséquent, pdf_open() doit avoir été appelé avant cette fonction.*

Voir aussi fopen(), PDF_get_info(), PDF_close().

10.48.8 PDF_close

void pdf_close (int **pdf document**)
PDF_close() ferme un document PDF. Note : *A cause de la programmation buggée de pdflib 0.6 fermer un document PDF ferme aussi le fichier associé. Cela ne devrait pas être le cas, puisque pdflib n'a pas ouvert le fichier, mais s'attend à un fichier déjà ouvert, lorsque PDF_open() est appelée. Par conséquent, il ne devrait pas fermer le fichier. Afin de résoudre de problème, mettez en commentaire la ligne 190 dans le fichier p_basic.c de pdflib 0.6 jusqu'à ce qu'une nouvelle version corrige ce bug.*
Note : *Cette fonction n'a pas besoin de patch avec pdflib, si pdflib 2.0 est activée.*

Voir aussi PDF_open(), fclose().

10.48.9 PDF_begin_page

void pdf_begin_page (int **pdf document**, double **height**, double **width**)
PDF_begin_page() commence une nouvelle page avec la taille **height** et la largueur **width**.
Voir aussi PDF_end_page().

10.48.10 PDF_end_page

void pdf_end_page (int **pdf document**)
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().

10.48.11 PDF_show

void pdf_show (int **pdf document**, string **text**)
PDF_show() affiche le texte **text** avec la position courante, et avec la police courante.
Voir aussi PDF_show_xy(), PDF_set_text_pos(), PDF_set_font().

10.48.12 PDF_show_boxed

void pdf_show_boxed (int **pdf document**, string **text**, double **x-coor**, double **y-coor**, double **width**, double **height**, string **mode**)
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 peuvant prendre les valeurs de "justify" (justification) ou "fulljustify" (justification complète).
Voir aussi PDF_show(), PDF_show_xy().

10.48.13 PDF_show_xy

void pdf_show_xy (int **pdf document**, 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().

10.48.14 PDF_set_font

void pdf_set_font (int **pdf document**, 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 polices (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 : *Cette fonction doit être appelée après PDF_begin_page() pour créer un document PDF valide.*

10.48.15 PDF_set_leading

void pdf_set_leading (int **pdf document**, double **distance**)
PDF_set_leading() permet de choisir la distance entre les lignes du texte. Cette valeur sera utilisée si du texte est affiché par PDF_continue_text().
Voir aussi PDF_continue_text().

10.48.16 PDF_set_parameter

void pdf_set_parameter (int **pdf document**, string **name**, string **value**)
PDF_set_parameter() fixe certaines valeurs de pdglib.

10.48.17 PDF_set_text_rendering

void pdf_set_text_rendering (int **pdf document**, int **mode**)
PDF_set_text_rendering() determine le rendu du texte. Les valeurs possibles pour **mode** sont 0=fill text (texte plein), 1=stroke text (???), 2=fill and 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.

10.48.18 PDF_set_horiz_scaling

void pdf_set_horiz_scaling (int **pdf document**, double **scale**)
PDF_set_horiz_scaling() fixe l'échelle horizontale du texte, à **scale** en pourcentage.

10.48.19 PDF_set_text_rise

void pdf_set_text_rise (int **pdf document**, double **rise**)
PDF_set_text_rise() fixe l'élévation du texte à **rise** points.

10.48.20 PDF_set_text_matrix

void pdf_set_text_matrix (int **pdf document**, array **matrix**)
PDF_set_text_matrix() choisi la matrice de texte : la matrice de texte determine les transformations de la police courante. La matrice est un tableau de 6 éléments.

10.48.21 PDF_set_text_pos

void pdf_set_text_pos (int **pdf document**, double **x-coor**, double **y-coor**)
PDF_set_text_pos() choisi la position du texte qui sera utilisée lors du prochain pdf_show().
Voir aussi PDF_show(), PDF_show_xy().

10.48.22 PDF_set_char_spacing

void pdf_set_char_spacing (int **pdf document**, double **space**)
PDF_set_char_spacing() fixe l'espacement des caractères.
Voir aussi PDF_set_word_spacing(), PDF_set_leading().

10.48.23 PDF_set_word_spacing

void pdf_set_word_spacing (int **pdf document**, double **space**)
PDF_set_word_spacing() fixe l'espacement des mots.
Voir aussi PDF_set_char_spacing(), PDF_set_leading().

10.48.24 PDF_skew

void pdf_skew (int **pdf document**, 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.

10.48.25 PDF_continue_text

void pdf_continue_text (int **pdf document**, 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(), PDF_set_text_pos().

10.48.26 PDF_stringwidth

double pdf_stringwidth (int **pdf document**, 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().

10.48.27 PDF_save

void pdf_save (int **pdf document**)
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().

10.48.28 PDF_restore

void pdf_restore (int **pdf document**)
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.

```
<?php PDF_save($pdf);
// tout un lot de rotations, translations, transformations...
PDF_restore($pdf) ?>
```

Voir aussi PDF_save().

10.48.29 PDF_translate

void pdf_translate (int **pdf document**, 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.

```
<?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);
?>
```

10.48.30 PDF_scale

void pdf_scale (int **pdf document**, double **x-scale**, double **y-scale**)
PDF_scale() choisi 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.

```
<?php PDF_scale($pdf, 72.0, 72.0);
PDF_lineto($pdf, 1, 1);
PDF_stroke($pdf);
```

?>

10.48.31 PDF_rotate

void pdf_rotate (int **pdf document**, double **angle**)
PDF_rotate() choisi la rotation en degré.

10.48.32 PDF_setflat

void pdf_setflat (int **pdf document**, double **value**)
PDF_setflat() choisi la platitude, et lui affecte une valeur entre 0 et 100.

10.48.33 PDF_setlinejoin

void pdf_setlinejoin (int **pdf document**, long **value**)
PDF_setlinejoin() choisi le paramètre "linejoin", et lui affecte une valeur entre 0 et 2.

10.48.34 PDF_setlinecap

void pdf_setlinecap (int **pdf document**, int **value**)
PDF_setlinecap() affecte au paramètre linecap une valeur entre 0 et 2.

10.48.35 PDF_setmiterlimit

void pdf_setmiterlimit (int **pdf document**, double **value**)
PDF_setmiterlimit() choisi la "miter limit" et lui affecte une valeur supérieure à 1.

10.48.36 PDF_setlinewidth

void pdf_setlinewidth (int **pdf document**, double **width**)
PDF_setlinewidth() affecte à largeur de ligne la valeur **width**.

10.48.37 PDF_setdash

void pdf_setdash (int **pdf document**, double **white**, double **black**)
PDF_setdash() choisi 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.

10.48.38 PDF_moveto

void pdf_moveto (int **pdf document**, double **x-coor**, double **y-coor**)
PDF_moveto() déplace le point courant à la position (**x-coor**, **y-coor**).

10.48.39 PDF_curveto

void pdf_curveto (int **pdf document**, double **x1**, double **y1**, double **x2**, double **y2**, double **x3**, double **y3**)
PDF_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**).
Voir aussi PDF_moveto(), PDF_lineto(), PDF_stroke().

10.48.40 PDF_lineto

void pdf_lineto (int **pdf document**, 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(), PDF_stroke().

10.48.41 PDF_circle

void pdf_circle (int **pdf document**, 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().

10.48.42 PDF_arc

void pdf_arc (int **pdf document**, 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().

10.48.43 PDF_rect

void pdf_rect (int **pdf document**, 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().

10.48.44 PDF_closepath

void pdf_closepath (int **pdf document**)
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.

10.48.45 PDF_stroke

void pdf_stroke (int **pdf document**)
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().

10.48.46 PDF_closepath_stroke

void pdf_closepath_stroke (int **pdf document**)
PDF_closepath_stroke() est une combinaison de PDF_closepath() et PDF_stroke(). Elle ferme aussi le chemin.
Voir aussi PDF_closepath(), PDF_stroke().

10.48.47 PDF_fill

void pdf_fill (int **pdf document**)
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().

10.48.48 PDF_fill_stroke

void pdf_fill_stroke (int **pdf document**)
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().

10.48.49 PDF_closepath_fill_stroke

void pdf_closepath_fill_stroke (int **pdf document**)
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().

10.48.50 PDF_endpath

void pdf_endpath (int **pdf document**)
PDF_endpath() ferme le chemin courant mais ne le clos pas.
Voir aussi PDF_closepath().

10.48.51 PDF_clip

void pdf_clip (int **pdf document**)
PDF_clip() aligne tous les dessins sur le chemin courant.

10.48.52 PDF_setgray_fill

void pdf_setgray_fill (int **pdf document**, double **gray value**)
PDF_setgray_fill() choisi la couleur grise comme couleur de remplissage.
Voir aussi PDF_setrgbcolor_fill().

10.48.53 PDF_setgray_stroke

void pdf_setgray_stroke (int **pdf document**, double **gray value**)
PDF_setgray_stroke() Fixe la couleur de dessin à un niveau de gris.
Voir aussi PDF_setrgbcolor_stroke().

10.48.54 PDF_setgray

void pdf_setgray (int **pdf document**, double **gray value**)
PDF_setgray() choisi la couleur grise comme couleur de remplissage et de dessin.
Voir aussi PDF_setrgbcolor_stroke(), PDF_setrgbcolor_fill().

10.48.55 PDF_setrgbcolor_fill

void pdf_setrgbcolor_fill (int **pdf document**, 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().

10.48.56 PDF_setrgbcolor_stroke

void pdf_setrgbcolor_stroke (int **pdf document**, double **red value**, double **green value**, double **blue value**)
PDF_setrgbcolor_stroke() choisi la couleur rgb comme couleur de remplissage.
Voir aussi PDF_setrgbcolor_stroke().

10.48.57 PDF_setrgbcolor

void pdf_setrgbcolor (int **pdf document**, double **red value**, double **green value**, double **blue value**)
PDF_setrgbcolor_stroke() choisi la couleur rgb comme couleur de remplissage.
Voir aussi PDF_setrgbcolor_stroke(), PDF_setrgbcolor_fill().

10.48.58 PDF_add_outline

int pdf_add_outline (int **pdf document**, string **text**, int **parent**, int **open**)
PDF_add_outline() ajoute un signet de nom **text** sur la page courante, et au point courant. Le signet est inséré comme un fils de la page **parent** et est ouvert par défaut si **open** n'est pas 0. La valeur retournée est un identifiant du signet, qui pourra être réutilisé comme parent d'autres signets. Vous pouvez ainsi créer

des hiérarchies de signets.
Malheureusement, pdflib ne copie pas la chaîne, ce qui force PHP à allouer la mémoire. Actuellement, cette mémoire n'est jamais libérée par une fonction PDF, mais prise en charge par le gestionnaire PHP.

10.48.59 PDF_set_transition

void pdf_set_transition (int **pdf document**, int **transition**)
PDF_set_transition() Fixe le mode de transition entre les pages. La valeur de **transition** peut être :

- 0 : aucune transition,
- 1 : deux lignes en travers de l'écran représente la page,
- 2 : plusieurs lignes en travers de l'écran représente la page,
- 3 : une boite représente la page,
- 4 : une seule ligne en travers de l'écran représente la page,
- 5 : l'ancienne page se dissout pour revéler la nouvelle,
- 6 : l'effet page d'un coin à l'autre
- 7 : l'ancienne page est simplement remplacée par la nouvelle (valeur par défaut)

Voir aussi PDF_set_duration().

10.48.60 PDF_set_duration

void pdf_set_duration (int **pdf document**, double **duration**)
PDF_set_duration() choisi la durée de transition, en secondes, entre deux pages.
Voir aussi PDF_set_transition().

10.48.61 PDF_open_gif

int pdf_open_gif (int **pdf document**, string **filename**)
PDF_open_gif() ouvre et charge l'image GIF du fichier **filename**. Le format doit être GIF. La fonction retourne un identifiant d'image PDF :

```
<?php
$im = PDF_open_gif($pdf, "test.gif");
pdf_place_image($pdf, $im, 100, 100, 1);
pdf_close_image($pdf, $im);
?>
```

Voir aussi PDF_close_image(), PDF_open_jpeg(), PDF_open_memory_image(), PDF_execute_image(), PDF_place_image(), PDF_put_image().

10.48.62 PDF_open_memory_image

int pdf_open_memory_image (int **pdf document**, int **image**)
PDF_open_memory_image() prend comme argument une image créée avec les fonctions PHP, et la rend disponible opur le document PDF. La fonction retourne un identifiant PDF d'image.

```
<?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(), PDF_execute_image(), PDF_place_image(), PDF_put_image().

10.48.63 PDF_open_jpeg

int pdf_open_jpeg (int **pdf document**, string **filename**)
PDF_open_jpeg() ouvre et charge l'image JPEG du fichier **filename**. Le format de l'image doit être JPEG. La fonction retourne un identifiant d'image PDF.
Voir aussi PDF_close_image(), PDF_open_gif(), PDF_open_memory_image(), PDF_execute_image(), PDF_place_image(), PDF_put_image().

10.48.64 PDF_close_image

void pdf_close_image (int **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_memory_image().

10.48.65 PDF_place_image

void pdf_place_image (int **pdf document**, int **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.
Voir aussi PDF_put_image().

10.48.66 PDF_put_image

void pdf_put_image (int **pdf document**, int **image**)
PDF_put_image() enregistre une image dans un fichier PDF pour utilisation ultérieure. L'image n'est pas visible. On peut l'afficher avec la fonction PDF_execute_image(), et aussi souvent que désiré. Cela permet d'utiliser plusieurs fois la même image, sans augmenter la taille du fichier.
PDF_put_image() et PDF_execute_image() est fortement recommandé pour les images de grande taille (plusieurs ko) si elles sont affichées plus d'une fois dans le document. Note : *Cette fonction n'a plus de sens avec la version 2.01 de pdflib. Elle ne fera que retourner une alerte.*

Voir aussi PDF_put_image(), PDF_place_image(), PDF_execute_image().

10.48.67 PDF_execute_image

void pdf_execute_image (int **pdf document**, int **image**, double **x-coor**, double **y-coor**, double **scale**)
PDF_execute_image() affiche une image qui a été enregistrée dans le document PDF, avec la fonction PDF_put_image(). L'image est implantée dans la page courante, et aux coordonnées données. L'image peut être changée d'échelle en même temps qu'elle est affichée. Une échelle de 1.0 affichera l'image à sa taille d'origine. Note : *Cette fonction n'a plus de sens avec la version 2.01 de pdflib. Elle ne fera que retourner une alerte.*

```
<?php
$im = ImageCreate(100, 100);
$col1 = ImageColorAllocate($im, 80, 45, 190);
ImageFill($im, 10, 10, $col1);
$pim = PDF_open_memory_image($pdf, $im);
pdf_put_image($pdf, $pim);
pdf_execute_image($pdf, $pim, 100, 100, 1);
pdf_execute_image($pdf, $pim, 200, 200, 2);
```

```
pdf_close_image($pdf, $pim);
?>
```

10.48.68 pdf_add_annotation

void pdf_add_annotation (int **pdf document**, double **llx**, double **lly**, double **urx**, double **ury**, string **title**, string **content**)
pdf_add_annotation() ajoute une note, dont le coin inférieur droit est (**llx**, **lly**) et le coin supérieur droit est (**urx**, **ury**).

10.49 Fonctions de paiement Verisign

Cette extension vous permet de traiter des transactions financières (cartes de crédits) avec les services de Verisign Payment Services (feu Signio : http://www.verisign.com/payment/).
Ces fonctions sont disponibles dès que PHP a été compilé avec l'option --with-pfpro[=DIR]. Vous aurez alors besoin du Kit de développement ad hoc pour votre plate-forme. Il peut être téléchargé à partir de l'interface de gestion une fois que vous vous êtes enregistés.
Une fois le SDK téléchargé, il faut copier les fichiers qui sont dans le dossier `lib' de la distribution. Copier le fichier d'entête `pfpro.h' dans `/usr/local/include' et la librarie `libpfpro.so' dans le dossier `/usr/local/lib'.
Lorsque vous utilisez ces fonctions, vous pouvez omettre l'appel aux fonctions pfpro_init() et pfpro_cleanup() car l'extension le fera automatiquement. Cependant, ces fonctions sont toujours disponibles, au cas o□ vous traiteriez de nombreuses transactions qui requeriraient un contr™le fin de la librairie. Vous pouvez effectuer autant de transactions que vous voulez avec pfpro_process() entre ces deux fonctions.
Ces fonctions ont été ajoutées dans PHP 4.0.2.
Note : *Ces fonctions ne sont qu'un lien vers Verisign Payment Services. Assurez vous de bien lire le guide Payflow Pro Developers Guide pour connaître toutes les finesses de paramétrage.*

10.49.1 pfpro_init

void pfpro_init
pfpro_init() sert à initialiser la librairie Payflow Pro. Vous pouvez omettre cet appel. Dans ce cas, l'extension initialisera automatiquement la librairie, en appelant cette fonction avant la première transaction.
Voir aussi pfpro_cleanup().

10.49.2 pfpro_cleanup

void pfpro_cleanup
pfpro_cleanup() sert à terminer une session de Payflow Pro proprement. Elle doit être appelée après le traitement de la dernière transaction, et avant la fin du script. Cependant, vous pouvez omettre de l'appeler, car l'extension le fera automatiquement à la fin du script.
Voir aussi pfpro_init().

10.49.3 pfpro_process

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 avec la réponse.
pfpro_process() exécute une transaction avec Payflow Pro. Le premier paramètre est un tableau associatif contenant les clés et valeurs qui seront encodées et passé au serveur de traitement de la transaction.
Le deuxième paramètre est optionnel est spécifie l'h™te à contacter. Par défaut, c'est "test.signio.com" : il vous faudra certainement la changer pour "connect.signio.com", pour faire des transactions en direct.
Le troisième paramètre spécifie le port de connexion. Par défaut, c'est 43, le port standard de SSL.
Le quatrième paramètre spécifie le délai d'expiration, en secondes. Par défaut, c'est 30 secondes. Notez que ce délai n'intervient que lorsqu'un lien a été fait jusqu'au serveur de transactions : il est possbile que votre script tourne un long moment, en cas de congestion du réseau.

Le cinquième paramètre, si nécessaire, spécifie le nom de votre proxy SSL. Le sixième paramètre spéficie le port à utiliser pour le proxy SSL.
Les sept et huit -ième paramètre spécifie l'identitée de connexion et le mot de passe à utiliser sur le proxy.
Retourne un tableau associatif avec la réponse.
Note : *Assurez vous de bien avoir lu le guide Payflow Pro Developers Guide pour avoir tous les détails sur les paramètres.*

```
<?php
pfpro_init();
$transaction = array(USER      => 'mylogin',
PWD       => 'mypassword',
TRXTYPE             => 'S',
TENDER  => 'C',
AMT       => 1.50,
ACCT      => '4111111111111111',
EXPDATE             => '0904'
                        );
$response = pfpro_process($transaction);
if (!$response) {
die("Impossible d'établir un lien avec Verisign.\n");
}
echo "Le code de réponse Verisign est ".$response[RESULT];
echo ", ce qui signifie : ".$response[RESPMSG]."\n";
echo "\nLa transaction : ";
print_r($transaction);
echo "\nLa réponse: ";
print_r($response);
pfpro_cleanup();
?>
```

10.49.4 pfpro_process_raw

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 contenant la réponse.
pfpro_process_raw() exécute une chaîne brute de transaction avec Payflow Pro. Il est vivement recommandé d'utiliser pfpro_process(), car l'encodage de ces transactions n'est pas standard.
Le premier paramètre est une chaîne contenant la transaction. Tous les autres paramètres sont les mêmes que pour pfpro_process(). La valeur de retour est une chaîne contenant la réponse brute.
Note : *Assurez vous de bien avoir lu le guide Payflow Pro Developers Guide pour avoir tous les détails sur les paramètres. Il est vivement recommandé d'utiliser pfpro_process().*

```
<?php
pfpro_init();
$response =
pfpro_process("USER=mylogin&PWD[5]=m&ndy&TRXTYPE=S&TENDER=C&AMT=1.50&ACCT=4111111111
111111&EXPDATE=0904");
if (!$response) {
die("Impossible de se connecter à Verisign.\n");
}
echo "La réponse de Verisign était ".$response;
pfpro_cleanup();
?>
```

10.49.5 pfpro_version

string pfpro_version
pfpro_version() retourne la version de la librairie Payflow Pro utilisée. Au moment de l'écriture de cette documentation, c'était L211.

10.50 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.
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 ".

Postmaster	PHP	Statut
postmaster &	pg_connect("", "", "", "", "dbname");	OK
postmaster -i &	pg_connect("", "", "", "", "dbname");	OK
postmaster &	pg_connect("localhost", "", "", "", "dbname");	Unable to connect to PostgreSQL server: connectDB() failed: Is the postmaster running and accepting TCP/IP (with -i) connection at 'localhost' on port '5432'? in /path/to/file.php3 on line 20. : 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'? dans /path/to/file.php3, ligne 20 ? @tab
postmaster -i &	pg_connect("localhost", "", "", "", "dbname");	OK

Il est possible de se connecter avec la commande suivante : $conn = pg_Connect("host=localhost port=5432 dbname=chris");
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.

```
<?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")
?>
```

10.50.1 pg_Close

bool pg_close (int **connection**)
Retourne FALSE si l'index de connexion n'est pas valable, et TRUE sinon.
Ferme la connexion avec le serveur PostgreSQL associé à l'index de connection.

10.50.2 pg_cmdTuples

int pg_cmdtuples (int **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.

```
<?php
$result = pg_exec($conn, "INSERT INTO verlag VALUES ('Autor')");
```

```
$cmdtuples = pg_cmdtuples($result);
echo $cmdtuples . " <- tuples modifiés.";
?>
```

10.50.3 pg_Connect

int pg_connect (string **host**, string **port**, string **options**, string **tty**, string **dbname**)
Retourne un index de connexion en cas de succès, et FALSE si une connexion n'a pas pu être établie. Ouvre une connexion avec un serveur PostgreSQL. Chaque argument doit être placé entre guillemets, y compris le numéro de port. Les arguments **options** et **tty** sont optionnels et peuvent être ignorés. Cette fonction retourne un index de connexion qui sera utilisé par d'autres fonctions PostgreSQL. Vous pouvez ouvrir plusieurs connexions simultanément.
Une connexion peut être aussi établie avec la commande suivante : $conn = pg_connect("dbname=marliese port=5432"); en dehors des paramètres **dbname** et **port**, **host**, **tty**, **options**, **user** et **password** sont des options.
Voir aussi pg_pConnect().

10.50.4 pg_DBname

string pg_dbname (int **connection**)
Retourne le nom de la base de donnée PostgreSQL associée à l'index de connexion connection, ou FALSE si connection n'est pas valide.

10.50.5 pg_ErrorMessage

string pg_errormessage (int **connection**)
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. Cette fonction retournera une chaîne contenant le message d'erreur généré par le serveur final.

10.50.6 pg_Exec

int pg_exec (int **connection**, string **query**)
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.*

10.50.7 pg_Fetch_Array

array pg_fetch_array (int **result**, int **row**, int **result_type**)
Retourne un tableau qui contient à la ligne demandée, dans le résultat identifiée par result, et FALSE, si 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().

```
<?php
$conn = pg_pconnect("","","","","publisher");
if (!$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";
?>
```

10.50.8 pg_Fetch_Object

object pg_fetch_object (int **result**, int **row**, int **result_type**)
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, cette fonction 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 retourne un objet

```
<?php
$database = "verlag";
$db_conn = pg_connect ("localhost", "5432", "", "", $database);
if (!$db_conn): ?>
   <H1>connexion impossible à la base postgres <? echo $database ?></H1> <?
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);
?>
```

10.50.9 pg_Fetch_Row

array pg_fetch_row (int **result**, int **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(), pg_result().

```
<?php
$conn = pg_pconnect("","","","","publisher");
if (!$conn) {
echo "Erreur.\n";
exit;
}
$result = pg_Exec ($conn, "SELECT * FROM authors");
if (!$result) {
echo "Erreur.\n";
exit;
}
$row = pg_fetch_row ($result, 0);
echo $row[0] . " <- row\n";
$row = pg_fetch_row ($result, 1);
echo $row[0] . " <- row\n";
$row = pg_fetch_row ($result, 2);
echo $row[1] . " <- row\n";
?>
```

10.50.10 pg_FieldIsNull

int pg_fieldisnull (int **result_id**, int **row**, mixed **field**)
Teste si un champs est à NULL. Retourne 0 si le champs n'est pas NULL. Retourne 1 si le champs est à NULL. Le champs peut être identifié avec son nom ou son index numérique (commencant à 0).

10.50.11 pg_FieldName

string pg_fieldname (int **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.

10.50.12 pg_FieldNum

int pg_fieldnum (int **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.

10.50.13 pg_FieldPrtLen

int pg_fieldprtlen (int **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.

10.50.14 pg_FieldSize

int pg_fieldsize (int **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. Retourne FALSE en cas d'erreur.

10.50.15 pg_FieldType

int pg_fieldtype (int **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.

10.50.16 pg_FreeResult

int pg_freeresult (int **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.

10.50.17 pg_GetLastOid

int pg_getlastoid (int **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.

10.50.18 pg_Host

string pg_host (int **connection_id**)
pg_Host() retourne le nom d'hôte associé à l'index de connexion PostgreSQL.

10.50.19 pg_loclose

void pg_loclose (int **fd**)
pg_loclose() ferme un objet de type Inversion Large Object. **fd** est un descripteur de fichier, obtenu avec pg_loopen().

10.50.20 pg_locreate

int pg_locreate (int **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).

10.50.21 pg_loopen

int pg_loopen (int **conn**, int **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".

10.50.22 pg_loread

string pg_loread (int **fd**, 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. **fd** est un identifiant valide d'objet de grande taille, et **len** indique la taille maximale de mémoire alloué à l'objet de grande taille.

10.50.23 pg_loreadall

void pg_loreadall (int **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.

10.50.24 pg_lounlink

void pg_lounlink (int **conn**, int **lobjid**)

pg_lounlink() efface l' objet de grande taille dont l'identifiant est **lobjid**.

10.50.25 pg_lowrite

int pg_lowrite (int **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, ou FALSE en cas d'erreur. **fd** est un descripteut d'objet de grande taille, obtenu avec pg_loopen().

10.50.26 pg_NumFields

int pg_numfields (int **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.

10.50.27 pg_NumRows

int pg_numrows (int **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.

10.50.28 pg_Options

string pg_options (int **connection_id**)
pg_Options() retourne une chaîne contenant les options de la connexion PostgreSQL.

10.50.29 pg_pConnect

int pg_pconnect (string **host**, string **port**, string **options**, string **tty**, string **dbname**)
Retourne un index de connexion en cas de succès, ou FALSE en cas d'erreur. Ouvre une connexion permanente à une base PostgreSQL. Chacun des arguments doit être entre guillemets, y compris le numéro de port. Les arguments **options** et **tty** sont optionnels, et peuvent être ignorés. Cette fonction retourne un index de connexion, nécessaire aux autres fonctions PostgreSQL. Vous pouvez établir plusieurs connexion persistantes en même temps. Voir aussi pg_Connect().
Une connexion peut aussi être établie avec la commande suivante : $conn = pg_pconnect("dbname=marliese port=5432"); les autres paramètres en dehors de **dbname** et **port** sont **host**, **tty**, **options**, **user** et **password** sont optionnels.

10.50.30 pg_Port

int pg_port (int **connection_id**)
pg_Port() retourne le numéro de port de la connexion identifiée **connection_id**.

10.50.31 pg_Result

mixed pg_result (int **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.

10.50.32 pg_tty

string pg_tty (int **connection_id**)
pg_tty()retourne le nom de tty de la connexion associée à **connection_id**.

10.51 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 PHP3. Certains fonctionnalités spécifiques ne sont pas encore disponibles, bien que ce module tâche de remédier à cette situation avec ses fonctions.

10.51.1 posix_kill

bool posix_kill (int **pid**, int **sig**)
Envoie le signal **sig** au processus **pid**. Retourne FALSE, si 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.

10.51.2 posix_getpid

int posix_getpid (void **)**
Retourne l'identifiant du processus courant.

10.51.3 posix_getppid

int posix_getppid (void **)**
Retourne l'identifiant du processus parent du processus courant.

10.51.4 posix_getuid

int posix_getuid (void **)**
Retourne l'ID numérique de l'utilisateur du processus courant. Reportez vous à posix_getpwuid() pour accéder au nom d'utilisateur.

10.51.5 posix_geteuid

int posix_geteuid (void **)**
Retourne l'UID effectif de l'utilisateur du processus courant. Reportez vous à posix_getpwuid() pour obtenir le nom d'utilisateur.

10.51.6 posix_getgid

int posix_getgid (void **)**
Retourne l'UID du groupe du processus courant. Reportez vous à posix_getgrgid() pour accéder au nom du groupe.

10.51.7 posix_getegid

int posix_getegid (void **)**
Retourne l'ID effectif du groupe du processus courant. Reportez vous à posix_getgrgid() pour transformer cette information en nom de groupe.

10.51.8 posix_setuid

bool posix_setuid (int **uid**)
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.
Retourne TRUE en cas de succès, FALSE sinon. Voir aussi posix_setgid().

10.51.9 posix_setgid

bool posix_setgid (int **gid**)
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().
Retourne TRUE en cas de succès, FALSE sinon.

10.51.10 posix_getgroups

array posix_getgroups (void **)**
Retourne un tableau contenant les identifiants du groupe du processus courant. Reportez vous à posix_getgrgid() pour pouvoir utiliser ces id.

10.51.11 posix_getlogin

string posix_getlogin (void **)**

Retourne le nom de login de l'utilisateur qui possède le processus courant. Reportez vous à posix_getpwnam() pour obtenir plus d'information sur cet utilisateur.

10.51.12 posix_getpgrp

int posix_getpgrp (void **)**
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.

10.51.13 posix_setsid

int posix_setsid (void **)**
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. Retourne un identifiant de session.

10.51.14 posix_setpgid

int posix_setpgid (int **pid**, int **pgid**)
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.

10.51.15 posix_getpgid

int posix_getpgid (int **pid**)
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.

10.51.16 posix_getsid

int posix_getsid (int **pid**)
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.

10.51.17 posix_uname

array posix_uname (void **)**
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.

10.51.18 posix_times

array posix_times (void **)**
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.

10.51.19 posix_ctermid

string posix_ctermid (void)
Encore à faire.

10.51.20 posix_ttyname

string posix_ttyname (int **fd**)
Encore à faire.

10.51.21 posix_isatty

bool posix_isatty (int **fd**)
Encore à faire.

10.51.22 posix_getcwd

string posix_getcwd (void)
Encore à faire très rapidement.

10.51.23 posix_mkfifo

bool posix_getcwd (string **pathname**, int **mode**)
Encore à faire très rapidement.

10.51.24 posix_getgrnam

array posix_getgrnam (string **name**)
Encore à faire.

10.51.25 posix_getgrgid

array posix_getgrgid (int **gid**)
Encore à faire.

10.51.26 posix_getpwnam

array posix_getpwnam (string **username**)
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 :

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. @tab
passwd	Contient le mot de passe de l'utilisateur, encrypté. Souvent, dans les systèmes utilisant les mots de passes "fantômes", un astérisque est retourné. @tab
uid	L'UID de l'utilisateur. @tab
gid	L'ID du groupe de l'utilisateur. Utilisez la fonction posix_getgrgid() pour connaitre le nom du groupe, et ses membres. @tab
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. @tab
dir	Cet élément contient le chemin absolu jusqu'au dossier racine de l'utilisateur. @tab
shell	Cet élément contient le chemin absolu jusqu'au dossier d'éxécution du shell de l'utilisateur. @tab

10.51.27 posix_getpwuid

array posix_getpwuid (int **uid**)
Retourne un tableau associatif contenant des informations sur un utilisateur repéré par son UID, passé dans le paramètre **uid**.
Les éléments du tableau sont :

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. @tab
passwd	Contient le mot de passe de l'utilisateur, encrypté. Souvent, dans les systèmes utilisant les mots de passes "fantômes", un astérisque est retourné. @tab
uid	Cette valeur devrait correspondre au paramètre **uid**, et donc, il est redondant. @tab
gid	L'ID du groupe de l'utilisateur. Utilisez la fonction posix_getgrgid() pour connaître le nom du groupe, et ses membres. @tab
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. @tab
dir	Cet élément contient le chemin absolu jusqu'au dossier racine de l'utilisateur. @tab
shell	Cet élément contient le chemin absolu jusqu'au dossier d'éxécution du shell de l'utilisateur. @tab

10.51.28 posix_getrlimit

array posix_getrlimit (void **)**
Encore à faire ASAP.

10.52 Pspell

La librairie @xref{function.pspell , , pspell()} vous permet de vérifier l'orthographe d'un mot, et suggérer des corrections.
Vous devez utiliser les librairies aspell et pspell libraries, disponibles ˆagrave; : http://pspell.sourceforge.net/.

10.52.1 pspell_new

int 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 param□egrave;tre de langage est constitué des deux lettre du codage de langue ISO 639, et du codage optionnel de pays ISO 3166, séparé par un '_'. Ce param□egrave;tre est nécessaire pour les langues qui ont plus d'une orthographe, comme l'anglais. Les valeurs reconnues sont ``americain'', ``britannique'', et ``canadien''. Le param□egrave;tre de jargon contient des informations supplémentaires pour distinguer deux listes de mots qui ont le m□me marquage de langue et d'orthographe. Le param□egrave;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'. Ce param□egrave;tre est livré tel quel, et souffre d'un manque important de tests. Plus d'informations et des exemples sont accessibles sur le site anglais de pspell : http://pspell.sourceforge.net/.

```
$pspell_link = pspell_new ("english");
```

10.52.2 pspell_mode

boolean pspell_mode (int **dictionary_link**, int **mode**)
Pspell_mode() change le mode d'orthographe.
There are three modes available:

- PSPELL_FAST - mode rapide (moins de suggestions)
- PSPELL_NORMAL - mode normal (plus de suggestions)
- PSPELL_BAD_SPELLERS - mode lent (beaucoup de suggestions)

```
$pspell_link = pspell_new ("english");
pspell_mode (PSPELL_FAST);
if (!pspell_check ($pspell_link, "testt")) {
   $suggestions = pspell_suggest ($pspell_link, "testt");
}
```

10.52.3 pspell_runtogether

boolean pspell_runtogether (int **dictionary_link**, int **mode**)
Pspell_runtogether() consid□egrave;re que les mots accolés sont des mots composés. Par exemple, "lechat" devient un composé valide, bien que l'espace entre les deux mots manque. Changer ce paramétrage ne modifie que les résultats retournés par pspell_check(); pspell_suggest() retournera toujours des suggestions qui ne seront pas modifiées par l'appel de @xref{function.php-runtogether , , php_runtogether()}.

```
$pspell_link = pspell_new ("english");
pspell_runtogether (true);
echo pspell_runtogether ($pspell_link, "thecat") ? "correct" : "faux";
```

10.52.4 pspell_check

boolean pspell_check (int **dictionary_link**, string **word**)
Pspell_check() vérifie l'orthographe d'un mot et retourne true si l'orthographe est correcte, false sinon.

```
$pspell_link = pspell_new ("english");
if (pspell_check ($pspell_link, "testt")) {
echo "This is a valid spelling";
} else {
echo "Désolé, mauvaise orthographe";
}
```

10.52.5 pspell_suggest

array pspell_suggest (int **dictionary_link**, string **word**)
Pspell_suggest() retourne un tableau de suggestions pour le mot **word**.

```
$pspell_link = pspell_new ("english");
if (!pspell_check ($pspell_link, "testt")){
   $suggestions = pspell_suggest ($pspell_link, "testt");
for ($i=0; $i < count ($suggestions); $i++) {
echo "Orthographes suggerées : " . $suggestions[$i] . "<br>";
   }
}
```

10.53 GNU Readline

Les fonctions readline() implémente une interface vers la librairie GNU Readline. Ce sont des fonctions qui fournissent des lignes de commandes éditables. C'est à dire qu'il simule le même comportement que Bash, lorsqu'il vous permet d'utiliser les touches de flèches pour insérer des caractères ou de lister l'historique des commandes. A cause de la nature interactive de cette librairie, elle est de peu d'utilité pour écrire des applications web, mais peut être utile pour écrire des scripts qui seront éxécutées depuis un shell. La page du projet de GNU Readline est :
http://cnswww.cns.cwru.edu/~chet/readline/rltop.html. Elle est tenu à jour par Chet Ramey, qui est uassi un des auteurs de Bash.

10.53.1 readline

string readline (string **prompt**)

Cette fonction retourne une chaîne entrée par l'utilisateur. Vous pouvez spécifier une chaîne qui servira de ligne de commande à l'utilisateur. La ligne retournée est débarassée du caractère de nouvelle ligne final. Vous devez l'ajouter vous même à l'historique, si vous souhaitez utiliser readline_add_history().

```
//Lit 3 commandes de l'utilisateur.
for ($i=0; $i < 3; $i++) {
      $line = readline ("Commande: ");
readline_add_history ($line);
}
//affiche l'historique
print_r (readline_list_history());
//affiche les variables
print_r (readline_info());
```

10.53.2 readline_add_history

void readline_add_history (string **line**)
Cette fonction ajoute une ligne à l'historique de commandes.

10.53.3 readline_clear_history

boolean readline_clear_history (void **)**
Cette fonction efface tout l'historique.

10.53.4 readline_completion_function

boolean readline_completion_function (string **line**)
Cette fonction enregistre une fonction de complétion. Vous devez fournir le nom d'une fonction qui accepte une ligne de commande partielle, et qui retourne un tableau contenant les suggestions. C'est le même type de fonctionnalité que lorsque vous utilisez la touche de tabulation sous Bash.

10.53.5 readline_info

mixed readline_info (string **varname** , string **newvalue**)
Appelée sans paramètre, cette fonction retourne un tableau de valeurs, qui sont les paramètres de configuration de la librairie. les éléments sont indexés par les valeurs suivantes : done, end, erase_empty_line, library_version, line_buffer, mark, pending_input, point, prompt, readline_name, et terminal_name.
Appelée avec un paramètre en argument, la valeur de ce paramètre est retournée. Appelée avec deux arguments, la valeur de ce paramètre est remplacé par la valeur fournie.

10.53.6 readline_list_history

array readline_list_history (void **)**
Cette fonction retourne un tableau avec l'historique des commandes complètes. Les éléments sont indexés numériquement, en commencant à zéro.

10.53.7 readline_read_history

boolean readline_read_history (string **filename**)
Cette fonction lit l'historique depuis un fichier.

10.53.8 readline_write_history

boolean readline_write_history (string **filename**)
Cette fonction écrit l'historique dans un fichier.

10.54 Fonction GNU Recode

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.

10.54.1 recode_string

string recode_string (string **request**, string **string**)
Recode la chaîne **string** en fonction de la requête **request**. 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.

10.54.2 recode_file

bool recode_file (int **input**, int **output**)
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.
Cette fonction ne gère pas encore les fichiers distants (URLs). Les deux fichiers doivent faire référence à des fichiers locaux.

10.55 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()

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.

```
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. */
```

10.55.1 ereg

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.
La recherche est sensible à la casse.
Retourne TRUE si une occurence 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 :

```
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().

10.55.2 ereg_replace

string ereg_replace (string **pattern**, string **replacement**, string **string**)
Cette fonction 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 :

```
$string = "Ceci est un test";
echo ereg_replace( " est", " etait", $string );
echo ereg_replace( "( )est ", "\\1etait", $string );
echo ereg_replace( "(( )est)", "\\2etait", $string );
```

Voir aussi ereg(), eregi() et eregi_replace().

10.55.3 eregi

int eregi (string **pattern**, string **string**, array **regs**)
Cette fonction est identique à ereg(), hormis le fait qu'elle ignore la casse des caractères lors de la recherche sur les caractères alphabétiques.
Voir aussi ereg(), ereg_replace() et eregi_replace().

10.55.4 eregi_replace

string eregi_replace (string **pattern**, string **replacement**, string **string**)
Cette fonction 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().

10.55.5 split

array split (string **pattern**, string **string**, int **limit**)
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':

```
$passwd_list = split( ":", $passwd_line, 5 );
```

Pour analyser une date qui est délimitée par des /, des points ou des tirets :

```
$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().

10.55.6 sql_regcase

string sql_regcase (string **string**)
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.

```
echo sql_regcase( "Foo bar" );

affichera @example [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.

10.56 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.

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.

10.56.1 sem_get

int sem_get (int **key**, int **max_acquire** , int **perm**)
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().

10.56.2 sem_acquire

int sem_acquire (int **sem_identifier**)
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().

10.56.3 sem_release

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().

10.56.4 shm_attach

int 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.

10.56.5 shm_detach

int shm_detach (int)
shm_detach() relâche 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.

10.56.6 shm_remove

int shm_remove (int **shm_identifier**)
Supprime un segment de mémoire partagée sous Unix. Toutes les données seront supprimées.

10.56.7 shm_put_var

int shm_put_var (int **shm_identifier**, int **variable_key**, mixed **variable**)
Insère ou modifie la variable **variable** avec la clé **variable_key**. Tous les types de variables (double, int, string, array) sont supportés.

10.56.8 shm_get_var

mixed shm_get_var (int **id**, int **variable_key**)
shm_get_var() retourne la variable repérée par **variable_key**. La variable est toujours présente en mémoire partagée.

10.56.9 shm_remove_var

int shm_remove_var (int **id**, int **variable_key**)
shm_remove_var() efface la variable **variable_key** de la mémoire partagée et libère la mémoire.

10.57 Gestion des 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 des requêtes. Lorsqu'un visiteur accède 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 tel. Lors des accès ultérieurs, elles ne seront définies que si l'utilisateur le fait.
Les options track_vars et gpc_globals modifient la façon avec laquelle les variables sont rechargéees. Si track_vars est activée, alors les variables de session seront accessibles dans le tableau associatif global $HTTP_STATE_VARS. Si gpc_globals est activé, alors les variables de session seront placés dans les variables globales associées. Si les deux options 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 le peu de fiabilité (les clients peuvent les refuser, ou les effacer), 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.
L'exemple suivant montre comment enregistrer une variable, et comment relier correctement des pages avec SID.

```
<?php
session_register("compteur");
$count++;
?>
Salut visiteur, vous avez vu cette page <? echo $compteur; ?> fois.<p>
<php?
# le <?=SID?> est nécessaire pour transmettre l'identifiant de session
# au cas oú les utilisateurs auraient inactivé les cookies
?>
Pour continuer, <A HREF="nextpage.php?<?=SID?>">clique ici</A>
```

Pour enregistrer ces informations dans une base de données, il vous faut utiliser la fonction session_set_save_handler (NDtraducteur : cette fonction semble avoir disparue). Il faudra alors implémenter la fonction suivante pour l'adapter à MySQL ou toute autre base de données :

```
<?php
function open ($save_path, $session_name) {
echo "ouvre ($save_path, $session_name)\n";
return true;
}
function close() {
echo "ferme\n";
return true;
}
function read ($key) {
echo "écriture ($key, $val)\n";
return "foo|i:1;";
}
function write ($key, $val) {
echo "écriture ($key, $val)\n";
return true;
}
function destroy ($key) {
return true;
}
function gc ($maxlifetime) {
return true;
}
session_set_save_handler ("open", "close", "read", "write", "destroy", "gc");
session_start();
```

```
$foo++;
?>
```

Cela va produire le résultat suivant :

```
$ ./php save_handler.php
Content-Type: text/html
Set-cookie: PHPSESSID=f08b925af0ecb52bdd2de97d95cdbe6b
open (/tmp, PHPSESSID)
read (f08b925af0ecb52bdd2de97d95cdbe6b)
write (f08b925af0ecb52bdd2de97d95cdbe6b, foo|i:2;)
close
```

Le <?=SID?> n'est pas nécessaire, si l'option --enable-trans-sid a été utilisé pour compiler PHP.
Le système de gestion des sessions dispose d'un grand nombre d'options, qui sont placées dans le fichier php.ini file. En voici un survol rapide :

- session.save_handler session.save_handler défini les noms des fonctions qui seront utilisées pour enregistrer et retrouver les données associées à une sessions. Par défaut, les sessions sont enregistrées dans des fichiers.
- session.save_path défini 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 oú les fichiers sont créés. Par défaut, le dossier est /tmp.
- 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 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 défini 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 10.66 WDDX functions. Par défaut, c'est le mode php qui est selectionné.
- session.gc_probability précise la probabilité que la routine gc (garbage collection) soit lancée, en pourcentage. Par défaut, 1.
- session.gc_maxlifetime fixe la durée, en secondes, au-dela de laquelle les données considérées comme inutiles seront supprimées.
- session.referer_check détermine si l'identifiant de session ids utilisé par des sites externe seront éliminé. Si les identifiants de sessions sont propagés avec la méthode des URL, des utilisateurs qui n'en connaîtrait pas l'utilité risque de divulguer ces valeurs, et cela ménera 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).

Note : *La gestion des sessions a été ajoutée dans PHP 4.0.*

10.57.1 session_start

bool session_start

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)
Cette fonction retourne toujours true.
Note : *Cette fonction a été ajoutée dans PHP 4.0.*

10.57.2 session_destroy

bool session_destroy
session_destroy() détruit toutes les données associées à la session courante.
Cette fonction retourne toujours true.
Note : *Cette fonction a été ajoutée dans PHP 4.0.*

10.57.3 session_name

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 alphanumériques; 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, enregistrées dans session.name au moment du démarrage. Ainsi, vous devez appeler session_name() à chaque requête (et avant session_start() ou session_register()).

```
<?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 : *Cette fonction a été ajoutée dans PHP 4.0.*

10.57.4 session_module_name

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 : *Cette fonction a été ajoutée dans PHP 4.0.*

10.57.5 session_save_path

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 de grandes quantités de petits fichiers efficacement : par exemple, sous Linux, reiserfs peut être plus efficace que ext2fs.*
Note : *Cette fonction a été ajoutée dans PHP 4.0.*

10.57.6 session_id

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 : *Cette fonction a été ajoutée dans PHP 4.0.*

10.57.7 session_register

bool session_register (mixed **name**, mixed **...**)

session_register() enregistre une variable avec le nom name dans la session courante.
Cette fonction retourne true lorsque la variable est correctement enregistrée. Note : *Cette fonction a été ajoutée dans PHP 4.0.*

10.57.8 session_unregister

bool session_unregister (string **name**)
session_unregister() supprime la variable nommée **name** dans la session courante .
Cette fonction retourne true (vrai) lorsque la variable a été correctement supprimée de la session. Note : *Cette fonction a été ajoutée dans PHP 4.0.*

10.57.9 session_is_registered

bool session_is_registered (string **name**)
session_is_registered() retourne true si il y a une variable du nom de **name** enregistrée dans la session courante. Note : *Cette fonction a été ajoutée dans PHP 4.0.*

10.57.10 session_decode

bool 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 : *Cette fonction a été ajoutée dans PHP 4.0.*

10.57.11 session_encode

bool session_encode
session_encode() retourne les données de session dans une chaîne. Note : *Cette fonction a été ajoutée dans PHP 4.0.*

10.58 SNMP functions

Afin de pouvoir utiliser les fonctions SNMP sous Unix, vous aurez besoin d'installer le package UCD SNMP. 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.

10.58.1 snmpget

string snmpget (string **hostname**, string **community**, string **object_id**, int **timeout**, int **retries**)
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")

10.58.2 snmpset

string snmpget (string **hostname**, string **community**, string **object_id**, string **type**, mixed **value**, int **timeout**, int **retries**)
Fixe 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**.

10.58.3 snmpwalk

array snmpwalk (string **hostname**, string **community**, string **object_id**, int **timeout** , int **retries**)
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 toute 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.
$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.
for ($i=0; $i<count($a); $i++) {
echo $a[$i];
}

10.58.4 snmpwalkoid

array snmpwalkoid (string **hostname**, string **community**, string **object_id**, int **timeout** , int **retries**)
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 dessus 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.
$a = snmpwalkoid("127.0.0.1", "public", "");

La fonction ci dessus 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]
\n";
}

10.58.5 snmp_get_quick_print

boolean snmp_get_quick_print (void **)**
Retourne la valeur courante, stockée dans la librairie UCD, de l'option quick_print. Par défaut, quick_print est inactivée.
$quickprint = snmp_get_quick_print();

L'exemple ci-dessus devrait retourner FALSE, si quick_print est inactivée, et, l'exemple retournera TRUE 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.

10.58.6 snmp_set_quick_print

void snmp_set_quick_print (boolean **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
\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. Cette fonction n'est pas disponible avec la librairie Windows SNMP.

10.59 de chaîne de caractères

Ces fonctions permettent la manipulations de chaînes de caractères. Certaines sections plus spécialisées sont disponibles dès les sections sur les expressions régulières et dans la section URL.

10.59.1 AddCSlashes

string addcslashes (string **str**, string **charlist**)
Retourne une chaîne avec des backslash 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.

```
$escaped = addcslashes ($no_echappe, "\0..\37!@\177..\377");
```

Note : *Ajouté dans PHP4b3-dev.*

Voir aussi stripcslashes(), stripslashes(), htmlspecialchars() et quotemeta().

10.59.2 AddSlashes

string addslashes (string **str**)
Retourne une chaîne avec des backslashes 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 ("), backslash (\) et NULL (la valeur null).
Voir aussi stripslashes(), htmlspecialchars() et quotemeta().

10.59.3 bin2hex

string bin2hex (string **str**)
Retourne une chaîne ASCII contenant la représentation hexadécimale de **str**. La conversion est faite avec le bit de poids fort en premier.

10.59.4 Chop

string chop (string **str**)
Retourne l'argument sans les espaces de fin de chaîne.

```
$trimmed = chop ($line);
```

Voir aussi trim().

10.59.5 Chr

string chr (int **ascii**)
Retourne le caractère de code ASCII **ascii**.

```
$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 escape: %c", 27);
```

Cette fonction est le contraire de ord(). Voir aussi sprintf() avec le format de chaîne %c.

10.59.6 chunk_split

string chunk_split (string **string**, int **chunklen** , string **end**)
Permet de scinder une chaîne en plus petit morceaux, comme dans le cas de la conversion en 10.63.2 base64_encode pour se conformer à la RFC 2045. Cette fonction 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.

```
# formate $data avec la sémantique RFC 2045
$new_string = chunk_split (base64_encode($data));
```

Cette fonction est nettement plus rapide que ereg_replace(). Note : *Cette fonction a été ajoutée en 3.0.6.*

10.59.7 convert_cyr_string

string convert_cyr_string (string **str**, string **from**, string **to**)
Cette fonction 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

10.59.8 count_chars

mixed count_chars (string **string**, **mode**)
Compte le nombre d'occurence de chaque octet (0..255) dans la chaîne **string** et le retourne de différente façon. L'option **Mode** prend, par défaut, la valeur 0. Suivant le **mode**, count_chars() retourne une des réponses suivante :

- 0 - un tableau avec l'octet comme clé, et la fréquence comme valeur.
- 1 - Identique à 0, mais seule les fréquences non nulles sont listées.
- 2 - Identique à 0, mais seule les fréquences nulles sont listées.
- 3 - une chaîne qui contient tous les octets utilisés.
- 4 - une chaîne contenant tous les octets non utilisés.

Note : *Cette fonction a été ajoutée en PHP 4.0.*

10.59.9 crypt

string crypt (string **str**, string **salt**)
crypt() va coder une chaîne en utilisant la méthode d'encryption du DES standard. Les arguments sont : la chaîne à encrypter, et un grain de sel qui servira de base pour l'encryption. 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 d'encryption. En fait, le DES standard est parfois remplacé par une encryption MD5. Le type d'encryption 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 d'encryption. Si le grain de sel n'est pas fourni, PHP générera alors un grain de 2 caractères,

pour le DES standard, à moins que le système ne dispose de MD5 : dans ce cas, PHP générera un grain de sel pour MD, par défaut. PHP affecte la variable d'environnement CRYPT_SALT_LENGTH, à 2 si il utilise le DES standard, et à 12 si il utilise le MD5.
L'encryption standard fournit le grain de sel dans les deux premiers octets du résultat de la fonction crypt().
Sur les systèmes qui supportent plusieurs méthodes d'encryption, 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 commencant à 1
- CRYPT_BLOWFISH - DES étendu avec 16-octets SALT commencant à 2

Il n'y a pas d'algorithme de décryptage, étant donné que crypt() est injective.

10.59.10 echo

echo (string **arg1**, string **argn...**)
Affiche tous les paramètres.
Echo() n'est pas une fonction à proprement parler, ce qui rend l'usage des parenthèses facultatives.

```
echo "Bonjour Monde";
echo "Cet echo se
répartis sur plusieurs lignes. Les nouvelles lignes
seront aussi affichées";
echo "et echo se\nrépartis sur plusieurs lignes. Les nouvelles lignes\nseront aussi affichées.";
```

Note : *En fait, si vous voulez passer plus d'un paramètre à echo(), vous ne DEVEZ pas les placer entre parenthèses.*
Voir aussi : print(), printf() et flush().

10.59.11 explode

array explode (string **separator**, string **string**)
Retourne un tableau qui contient les éléments de la chaîne, séparés par *separator*.

```
$pizza = "piece1 piece2 piece3 piece4 piece5 piece6";
$pieces = explode (" ", $pizza);
```

Voir aussi split() et implode().

10.59.12 flush

void flush
Vide les buffers de sortie de PHP et toud ceux que PHP utilisait (CGI, un serveur web, etc.).

10.59.13 get_html_translation_table

string get_html_translation_table (int **table**)
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.

```
$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 :
$trans = array_flip ($trans);
$original = strtr ($str, $trans);
Le contenu de $original sera : "Hallo & >Frau> & Krämer". Note : *Cette fonction a été ajoutée en PHP 4.0.*

Voir aussi : htmlspecialchars(), htmlentities(), strtr(), et array_flip().

10.59.14 get_meta_tags

array get_meta_tags (string **filename**, int **use_include_path**)
Ouvre le fichier **filename** et l'analyse ligne par ligne, en recherchant les balises <meta>.

<meta name="author" content="name">
<meta name="tags" content="php3 documentation">
</head> <!-- parsing stops here -->

(Faites bien attention aux fins de lignes. PHP utilise une fonction native pour analyser le fichier d'entrée, ce qui fait que les fichiers faits sous Mac 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 la nom de la propriété sont remplacés par des '_', le reste est converti en minuscule.
Mettre **use_include_path** à 1 forcera PHP à ouvrir les fichiers dans le chemin standard d'inclusion.

10.59.15 htmlentities

string htmlentities (string **string**)
Cette fonction est identique à Htmlspecialchars() en tous points, sauf que tous les caractères qui ont une entité équivalente en HTML sont replacés par ces entités.
Actuellement, le jeu de caractères ISO-8859-1 character est utilisé.
Voir aussi htmlspecialchars() et nl2br().

10.59.16 htmlspecialchars

string htmlspecialchars (string **string**)
Certains caractères ont une valeur avec HTML, et doivent être remplacés par des balises HTML pour conserver leur valeur. Cette fonction retourne une chaîne avec tous les caractères sensibles remplacés par leur équivalent.
Cette fonction est utile pour empêcher un utilisateur de fournir un texte avec un sens HTML, comme dans un livre d'or.
Actuellement, PHP remplace les valeurs suivantes :

- '&' (et commercial) devient '&'
- '"' guillement double) devient '"'
- '<' (inférieur à) devient '<'
- '>' (supérieur à) devient '>'

Notez bien que cette fonction ne fait aucun autre remplacment que ceux listés ci-dessus. Pour une traduction complète de toutes les balises, reportez vous à htmlentities().
Voir aussi htmlentities() et nl2br().

10.59.17 implode

string implode (string **glue**, array **pieces**)
Retourne une chaîne constituée de tous les éléments du tableau, pris dans l'ordre, transformés en chaîne, et séparés par **glue**.

$colon_separated = implode (":", $array);

Voir aussi explode(), join(), et split().

10.59.18 join

string join (string **glue**, array **pieces**)
join() join() est un alias de implode(), et lui est identique en tous points.
Voir aussi explode(), implode(), et split().

10.59.19 ltrim

string ltrim (string **str**)
Cette fonction 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().

10.59.20 md5

string md5 (string **str**)
Crypte la chaîne **str** en utilisant la méthode MD5 (voir RSA Data Security. Inc. MD5 Message-Digest Algorithm.).

10.59.21 Metaphone

string metaphone (string **str**)
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.
Metaphone a été développé par Lawrence Philips <lphilips@verity.com>. Elle est décrit dans ["Practical Algorithms for Programmers", Binstock & Rex, Addison Wesley, 1995]. Note : *Cette fonction a été ajoutée en PHP 4.0.*

10.59.22 nl2br

string nl2br (string **string**)
Retourne la chaîne **string** dont toutes les lignes ont été remplacées par '
'.
Voir aussi htmlspecialchars() et htmlentities().

10.59.23 Ord

int ord (string **string**)
Retourne la valeur ASCII du premier caractère de la chaîne **string**. Cette fonction est le contraire de chr().

```
if (ord ($str) == 10) {
echo "Le premier caractère de \$str est un retour chariot.\n";
}
```

Voir aussi chr().

10.59.24 parse_str

void parse_str (string **str**)
Analyse la chaîne **str** comme si c'etait une chaîne passée par URL, et affecte les variables qu'elle y trouve.

```
$str = "first=value&second[]=this+works&second[]=another";
parse_str($str);
echo $first;     /* prints "value" */
echo $second[0]; /* prints "this works" */
echo $second[1]; /* prints "another" */
```

10.59.25 print

print (string **arg**)
Affiche **arg**.
Voir aussi : echo(), printf(), et flush().

10.59.26 printf

int printf (string **format**, mixed **args**...)
Affiche les arguments en fonction du **format**. Ce format est décrit en détails dans la documentation de sprintf().
Voir aussi : print(), sprintf() et flush().

10.59.27 quoted_printable_decode

string quoted_printable_decode (string **str**)
Cette fonction retourne une chaîne 8-bit résultant du décodage de la chaîne **str**. Cette fonction est similaire à imap_qprint(), hormis le fait qu'elle ne requiert pas le module IMAP.

10.59.28 QuoteMeta

string quotemeta (string **str**)
Retourne une version de la chaîne **str**, avec un backslash (\) devant tous les caractères de la liste ci-dessous :
. \\ + * ? [^] ($)

Voir aussi addslashes(), htmlentities(), htmlspecialchars(), nl2br() et stripslashes().

10.59.29 rawurldecode

string rawurldecode (string **str**)
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@example foo%20bar%40baz devient @example foo bar@baz
.
Voir aussi rawurlencode().

10.59.30 rawurlencode

string rawurlencode (string **str**)
Retourne une chaîne dont tous les caractères non-alpha-numériques (hormis @example -_.
.) 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 :

```
echo '<A HREF="ftp://user:', rawurlencode ('foo @+%/'),
    '@ftp.my.com/x.txt">';
```

Ou, si vous transmettez un chemin dans une URL

```
echo '<A HREF="http://x.com/department_list_script/',
rawurlencode ('sales et marketing/Miami'), '">';
```

Voir aussi rawurldecode().

10.59.31 setlocale

string setlocale (string **category**, string **locale**)
Category est une chaîne qui spécifie la catégorie de fonction qui va être affectée par les informations locales :

- LC_ALL Toutes les fonctions ci-dessous
- LC_COLLATE pour les comparaison de chaîne (en cours d'implémentation)
- 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 plateforme. Une catégorie invalide provoque une alerte.

10.59.32 similar_text

int similar_text (string **first**, string **second**, double **percent**)
Cette fonction 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.

10.59.33 soundex

string soundex (string **str**)
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. Cette fonction retourne une chaîne de 4 caractères, commencant par une lettre.
Cette fonction particulière a été décrite par Donald Knuth dans "The Art Of Computer Programming, vol. 3: Sorting And Searching", Addison-Wesley (1973), pp. 391-392.

```
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';
```

10.59.34 sprintf

string sprintf (string **format**, mixed **args**...)
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 quel (hormis %), et des spécifications, chacune d'elle disposant de son propre paramètre. Cela s'applique à sprintf() et printf().
Chaque conversion consiste, dans l'ordre:

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 specifier* 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 specifier* indique le nombre minimum de caractères que la conversion devrait retourner.
4. Argument optionnel, *precision specifier* 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, double. (Une autre fonciton 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.
    - 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 minuscule).
    - X - l'argument est traité comme un entier, et représenté comme un nombre hexadécimal (en majuscule).

Voir aussi : printf() et number_format().

```
$isodate = sprintf ("%04d-%02d-%02d", $year, $month, $day);

$money1 = 68.75;
$money2 = 54.35;
$money = $money1 + $money2;
// echo $money affichera "123.1";
$formatted = sprintf ("%01.2f", $money);
// echo $formatted affichera "123.10"
```

10.59.35 strcasecmp

int strcasecmp (string **str1**, string **str2**)
Retourne < 0 si **str1** est plus petit que **str2**; > 0 si **str1** est plus grand que **str2**, et 0 si ils sont égaux.

```
$var1 = "Bonjour";
$var2 = "bonjour";
if ( !strcasecmp($var1,$var2) ) {
echo '$var1 est égal à $var2, à la casse près.';
}
```

Voir aussi ereg(), strcmp(), substr(), stristr(), et strstr().

10.59.36 strchr

string strchr (string **haystack**, string **needle**)
Cette fonction est un alias de strstr(), et lui est identique en tous points.

10.59.37 strcmp

int strcmp (string **str1**, string **str2**)
Retourne < 0 si **str1** est plus petit que **str2**; > 0 si **str1** est plus grand que **str2**, et 0 si ils sont égaux.
Notez bie nque la comparaison est sensible à la casse.
Voir aussi ereg(), strcasecmp(), substr(), stristr() et strstr().

10.59.38 strcspn

int strcspn (string **str1**, string **str2**)
Retourne la longueur du premier segment de la chaîne **str1** qui *ne* contiennent *pas* aucun des caractères de la chaîne **str2**.
Voir aussi strspn().

10.59.39 strip_tags

string strip_tags (string **str**, string **allowable_tags**)
Cette fonction 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. Cette fonction 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 PHP4B3.*

10.59.40 StripCSlashes

string stripcslashes (string **str**)
Retourne une chaîne dont les backslashes ont été supprimés. Cette fonction reconnait les \n, \r ..., et les représentation octales et hexadécimales utilisées en C. Note : *Ajouté dans PHP4b3-dev.*

Voir aussi addcslashes().

10.59.41 StripSlashes

string stripslashes (string **str**)
Retourne une chaîne dont tous les slashes ont été supprimés. (\' devient ', ... et ainsi de suite). Les doubles backslashes sont remplacés par des simples.
Voir aussi addslashes().

10.59.42 stristr

string stristr (string **haystack**, string **needle**)
Retourne tous les éléments de **haystack** à partir de la première occurence 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, est utilisée comme si elle était passée à chr().
Voir aussi strchr(), strrchr(), substr(), et ereg().

10.59.43 strlen

int strlen (string **str**)
Retourne la longueur de la chaîne **string**.

10.59.44 strpos

int strpos (string **haystack**, string **needle**, int **offset**)
Retourne la position numérique de la première occurence 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 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.
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(), substr(), stristr() et strstr().

10.59.45 strrchr

string strrchr (string **haystack**, string **needle**)
Cette fonction retourne la partie de la chaîne **haystack** qui commence à la dernière occurence de **needle** et va jusqu'à la fin de la chaîne **haystack**.
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.

```
// lis le dernier repertoire de $PATH
$dir = substr (strrchr ($PATH, ":"), 1);
// lis tout après la dernière ligne
$text = "Line 1\nLine 2\nLine 3";
$last = substr (strrchr ($text, 10), 1 );
```

Voir aussi substr(), stristr(), et strstr().

10.59.46 str_repeat

string str_repeat (string **input**, int **multiplier**)
Retourne **input_str** répétées **multiplier** fois. **multiplier** doit être plus grand que 0.

```
echo str_repeat ("-=", 10);
```

Cet exemple affichera "-=-=-=-=-=-=-=-=-=-=".
Note : *Cette fonction a été ajoutée en PHP 4.0.*

10.59.47 strrev

string strrev (string **string**)
Retourne **string**, après avoir changé l'ordre des caractères.

10.59.48 strrpos

int strrpos (string **haystack**, char **needle**)
Retourne la position numérique de la dernière occurence de **needle** dans la chaîne **haystack** string. Cette fonction ne peut accepter qu'un seul caractère.
Si **needle** n'est pas trouvé, retourne FALSE.
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().

10.59.49 strspn

int strspn (string **str1**, string **str2**)
Retourne la longueur du premier segment de **str1** qui est constitué entièrement de caractères dans la chaîne **str2**.
Voir aussi strcspn().

10.59.50 strstr

string strstr (string **haystack**, string **needle**)
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.
Si **needle** n'est pas une chaîne, elle est convertie en entier, et utilisée comme valeur ordinale d'un caractère.

```
$email = 'sterling@designmultimedia.com';
$domain = strstr ($email, '');
print $domain; // affiche designmultimedia.com
```

Voir aussi stristr(), strrchr(), substr(), et ereg().

10.59.51 strtok

string strtok (string **arg1**, string **arg2**)
strtok() est utilisée pour morceller une chaîne. Pour cela, si vous avez une chaîne du type "ceci est une chaîne exemple", vous pouvez la morceller en mots, en utilisant ' ' comme délimiteur.

```
$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 morcellée 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().

10.59.52 strtolower

string strtolower (string **str**)
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.

```
$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().

10.59.53 strtoupper

string strtoupper (string **string**)
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.

```
$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().

10.59.54 str_replace

string str_replace (string **needle**, string **str**, string **haystack**)
Cette fonction remplace toutes les occurences de **needle** dans **haystack** par la chaîne **str**. Si vous n'avez pas besoin de règles de remplacement sophistiquées, vous pouvez toujours utiliser ereg_replace().

```
$bodytag = str_replace ("%body%", "black", "<body text=%body%>");
```

Cette fonction n'altère pas les données binaires.
Note : *Str_replace() a été ajoutée dans PHP 3.0.6, mais était erronée jusqu'à PHP 3.0.8.*
Voir aussi ereg_replace() et strtr().

10.59.55 strtr

string strtr (string **str**, string **from**, string **to**)

Cette fonction travaille sur **str**, remplacant chaque occurence de chaque caractère de la chaîne **from** correspondant à la chaîne **to** et retourne le résultat.
Si **from** et **to** sont de longueur différentes, les caractères en trop sont ignorés.

```
$addr = strtr($addr, "äåö", "aao");
```

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 à déjà remplacé.
Exemples:

```
$trans = array ("hello" => "hi", "hi" => "hello");
echo strtr("hi all, I said hello", $trans) . "\n";
```

Cete exemple affichera : "hello all, I said hi",
Note : *Travailler avec deux arguments a été ajouté dans PHP 4.0.*
Voir aussi ereg_replace().

10.59.56 substr

string substr (string **string**, int **start**, int **length**)
Substr() retourne une portion de **string**, spécifiée avec le début **start** et la longeur **length**.
Si **start**est positif, la chaîne retournée commencera au caractère **start** de la chaîne **string**. Par exemple:
Exemples:

```
$rest = substr ("abcdef", 1);    // retourne "bcdef"
$rest = 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:
Exemples:

```
$rest = substr ("abcdef", -1);    // retourne "f"
$rest = substr ("abcdef", -2);    // retourne "ef"
$rest = substr ("abcdef", -3, 1); // retourne "d"
```

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.
Exemples:

```
$rest = substr ("abcdef", 1, -1); // retourne "bcde"
```

Voir aussi strrchr() et ereg().

10.59.57 substr_replace

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.

```
<?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 dans PHP 4.0.*

10.59.58 trim

string trim (string **str**)
Cette fonction 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().

10.59.59 ucfirst

string ucfirst (string **str**)
Met le premier caractère d'une chaîne **str** en majuscule, 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.

```
$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().

10.59.60 ucwords

string ucwords (string **str**)
Met le premier caractère de chaque mot de la chaîne **str** si ce caractère est une lettre.

```
$text = "marie a un petit agneau, et l'adore.";
$text = ucwords($text); // $text vaut : Marie A Un Petit Agneau, Et l'Adore.
```

Voir aussi strtoupper(), strtolower() et ucfirst().

10.60 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 :

```
<?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, "Voici le mariage de FLASH et 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; (mais traduite en anglais).
Note : *Le support de Flash a été ajouté dans PHP4 RC2.*

10.60.1 swf_openfile

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).

10.60.2 swf_closefile

void swf_closefile
Ferme le fichier courant, qui a été ouvert avec swf_openfile().

10.60.3 swf_labelframe

void swf_labelframe (string **name**)
Donne le nom **name** au frame courant.

10.60.4 swf_showframe

void swf_showframe
swf_showframe() affiche le frame courant.

10.60.5 swf_setframe

void swf_setframe (int **framenumber**)
swf_setframe() selectionne le frame **framenumber** comme frame actif.

10.60.6 swf_getframe

int swf_getframe
swf_getframe() retourne le numéro de frame courant.

10.60.7 swf_mulcolor

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.*

10.60.8 swf_addcolor

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.*

10.60.9 swf_placeobject

void swf_placeobject (int **objid** , int **depth**)
Place l'objet **objid** dans le frame courant, à la profondeur **depth**. **objid** et **depth** doivent être compris entre 1 et 65535.
Cette fonction 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.*

10.60.10 swf_modifyobject

void swf_modifyobject (int **depth** , int **how**)
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.

10.60.11 swf_removeobject

void swf_removeobject (int **depth**)
Enlève l'objet situé à la profondeur **depth** de la scène.

10.60.12 swf_nextid

int swf_nextid
swf_nextid() retourne le prochain identifiant d'objet libre.

10.60.13 swf_startdoaction

void swf_startdoaction
swf_startdoaction() commence la déscription d'une liste d'action pour la frame courante. Cette fonction doit être appelée avant que les actions ne soient définies pour le cadre courant.

10.60.14 swf_actiongotoframe

void swf_actiongotoframe (int **framenumber**)
swf_actionGotoFrame() se déplace jusqu'au frame **framenumber**, le joue, puis s'arrête.

10.60.15 swf_actiongeturl

void swf_actiongeturl (string **url** , string **target**)
swf_actionGetUrl() lit l'URL **url**, avec la destination **target**.

10.60.16 swf_actionnextframe

void swf_actionnextframe
swf_actionnextframe() avance d'un frame le frame courant.

10.60.17 swf_actionprevframe

void swf_actionprevframe
swf_actionnextframe() recule d'un frame le frame courant.

10.60.18 swf_actionplay

void swf_actionplay
swf_actionplay() joue l'animation flash à partir du frame courant.

10.60.19 swf_actionstop

void swf_actionstop
Arrête l'animation flash au frame courant.

10.60.20 swf_actiontogglequality

void swf_actiontogglequality
Choisi le niveau de qualité haut ou bas.

10.60.21 swf_actionwaitforframe

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...".

10.60.22 swf_actionsettarget

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.

10.60.23 swf_actiongotolabel

void swf_actiongotolabel (string **label**)
swf_actionGotoLabel() affiche le frame de nom **label**, puis stoppe.

10.60.24 swf_enddoaction

void swf_enddoaction
swf_startdoaction() termine l'action courante.

10.60.25 swf_defineline

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**.

10.60.26 swf_definerect

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.

10.60.27 swf_definepoly

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.

10.60.28 swf_startshape

void swf_startshape (int **objid**)
swf_startshape() commence une forme complexe, qui sera reperé par l'identifiant d'objet **objid**.

10.60.29 swf_shapelinesolid

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.

10.60.30 swf_shapefilloff

void swf_shapefilloff
swf_shapeFillOff() inactive le remplissage pour la forme courante.

10.60.31 swf_shapefillsolid

void swf_shapefillsolid (float **r** , float **g** , float **b** , float **a**)
swf_shapeFillSolid() fixe la couleur pour le style courant de remplissage à **rgba**.

10.60.32 swf_shapefillbitmaptile

void swf_shapefillbitmapclip (int **bitmapid**)
Choisi le mode de remplissage par texture : les espaces vides seront remplis avec la bitmap **bitmapid**.

10.60.33 swf_shapefillbitmaptile

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.

10.60.34 swf_shapemoveto

void swf_shapemoveto (float **x** , float **y**)
swf_shapeMoveTo() fixe la position courante au point de de coordonnées (**x**, **y**).

10.60.35 swf_shapelineto

void swf_shapelineto (float **x** , float **y**)
swf_shapeLineTo() dessine une ligne entre la position courante et le points de coordonnées (**x**, **y**). La position courante devient alors (**x**, **y**).

10.60.36 swf_shapecurveto

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**, et **y2**).

10.60.37 swf_shapecurveto3

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.

10.60.38 swf_shapearc

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**.

10.60.39 swf_endshape

void swf_endshape
swf_endshape() complète la définition de la forme courante.

10.60.40 swf_definefont

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.

10.60.41 swf_setfont

void swf_setfont (int **fontid**)
swf_setfont() remplace la police courante par la police repérée par l'identifiant **fontid**.

10.60.42 swf_fontsize

void swf_fontsize (float **size**)
swf_fontsize() remplace la taille de la police par la taille **size**.

10.60.43 swf_fontslant

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.

10.60.44 swf_fonttracking

void swf_fonttracking (float **tracking**)
swf_fonttracking() change l'espacement, et lui affecte la valeur de th**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.

10.60.45 swf_getfontinfo

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.

10.60.46 swf_definetext

void swf_definetext (int **objid** , string **str** , int **docenter**)
Définit la chaîne de texte **str**, en utilisant la police courante. **docenter** indique si la chaîne doit être centrée (valeur de 1), ou pas.

10.60.47 swf_textwidth

float swf_textwidth (string **str**)
swf_textwidth() retourne la longueur de la chaîne **str**, en pixels, en utilisant la police courante.

10.60.48 swf_definebitmap

void swf_definebitmap (int **objid** , string **image_name**)
The swf_definebitmap() function defines a bitmap given a GIF, JPEG, RGB or FI image. The image will be converted into a Flash JPEG or Flash color map format.

10.60.49 swf_getbitmapinfo

array swf_getbitmapinfo (int **bitmapid**)
swf_getbitmapinfo() retourne un tableau d'information sur l'image bitmap repérée par **bitmapid**. Le tableau a les éléments suivant :

- "size" - La taille en octets de l'image.
- "width" - La largueur en pixels de l'image.
- "height" - La hauteur en pixels de l'image.

10.60.50 swf_startsymbol

void swf_startsymbol (int **objid**)
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.

10.60.51 swf_endsymbol

void swf_endsymbol
swf_endsymbol() termine la définition de symble, qui a été commencée avec swf_startsymbol().

10.60.52 swf_startbutton

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.

10.60.53 swf_addbuttonrecord

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.

```
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 ();
```

10.60.54 swf_oncondition

void swf_oncondition (int **transition**)
swf_onCondition() décrit une transition qui va déclencher une liste d'actions. Il y a plusieurs types de transitions 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)

10.60.55 swf_endbutton

void swf_endbutton
swf_endButton() termine la définition du bouton courant.

10.60.56 swf_viewport

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.

10.60.57 swf_ortho

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.

10.60.58 swf_ortho2

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().

10.60.59 swf_perspective

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.*

10.60.60 swf_polarview

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.

10.60.61 swf_lookat

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.

10.60.62 swf_pushmatrix

void swf_pushmatrix
swf_pushmatrix() empile la matrice de transformation courante dans la pile.

10.60.63 swf_popmatrix

void swf_popmatrix
swf_popmatrix() dépile la matrice de transformation.

10.60.64 swf_scale

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.

10.60.65 swf_translate

void swf_translate (double **x** , double **y** , double **z**)
swf_translate() translate la transformation courante de **x**, **y**, et **z**, dans les directions x, y et z.

10.60.66 swf_rotate

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).

10.60.67 swf_posround

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).

10.61 d'accès à Sybase

10.61.1 sybase_affected_rows

int sybase_affected_rows (int **link_identifier**)
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 : *Cette fonction est disponible avec l'interface CT vers Sybase, mais pas avec la librairie DB.*

10.61.2 sybase_close

int sybase_close (int **link_identifier**)
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().

10.61.3 sybase_connect

int sybase_connect (string **servername**, string **username**, string **password**)
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().

10.61.4 sybase_data_seek

int sybase_data_seek (int **result_identifier**, int **row_number**)
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().

10.61.5 sybase_fetch_array

int sybase_fetch_array (int **result**)
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().

10.61.6 sybase_fetch_field

object sybase_fetch_field (int **result**, int **field_offset**)
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.

Voir aussi sybase_field_seek().

10.61.7 sybase_fetch_object

int sybase_fetch_object (int **result**)
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().

10.61.8 sybase_fetch_row

array sybase_fetch_row (int **result**)
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().

10.61.9 sybase_field_seek

int sybase_field_seek (int **result**, int **field_offset**)
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().

10.61.10 sybase_free_result

int 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.

10.61.11 sybase_num_fields

int sybase_num_fields (int **result**)
sybase_num_fields() retourne le nombre de champs dans un résultat.
Voir aussi: sybase_query(), sybase_fetch_field() et sybase_num_rows().

10.61.12 sybase_num_rows

int sybase_num_rows (string **result**)
sybase_num_rows() retourne le nombre de lignes dans un résultat.
Voir aussi: sybase_query() et sybase_fetch_row().

10.61.13 sybase_pconnect

int sybase_pconnect (string **servername**, string **username**, string **password**)
Retourne un identifiant de connexion positive 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'.

10.61.14 sybase_query

int sybase_query (string **query**, int **link_identifier**)
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().

10.61.15 sybase_result

int sybase_result (int **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().

10.61.16 sybase_select_db

int sybase_select_db (string **database_name**, int **link_identifier**)
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().

10.62 ODBC

10.62.1 odbc_autocommit

int odbc_autocommit (int **connection_id**, int **OnOff**)
Sans paramètre **OnOff**, cette fonction retourne le statut d'auto-validation de la connexion **connection_id**. TRUE si le mode est activé, FALSE si il ne l'est pas, ou si une erreur survient.
Si **OnOff** vaut TRUE, l'auto-validation est activée. Si il est FALSE, l'auto-validation est desactivée. 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().

10.62.2 odbc_binmode

int odbc_binmode (int **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: Converti 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".

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 modifié par odbc_longreadlen().*

10.62.3 odbc_close

void odbc_close (int **connection_id**)
odbc_close() ferme la connexion avec une source de données, représentée par l'identifiant de connexion. Note : *Cette fonction échouera si il y a des transactions en cours sur cette connexion. Dans ce cas, la connexion restera ouverte.*

10.62.4 odbc_close_all

void odbc_close_all
odbc_close_all() ferme toutes les connexions ODBC à des sources de données. Note : *Cette fonction échouera si il y a des transactions en cours sur cette connexion. Dans ce cas, la connexion restera ouverte.*

10.62.5 odbc_commit

int odbc_commit (int **connection_id**)
Retourne: TRUE en case de succès, FALSE en cas d'erreur. Toutes les connexions en cours sur **connection_id** sont validées. Toutes les connexions en cours sur **connection_id** sont validées.

10.62.6 odbc_connect

int odbc_connect (string **dsn**, string **user**, string **password**, int **cursor_type**)
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 selection (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().

10.62.7 odbc_cursor

string odbc_cursor (int **result_id**)
odbc_cursor() lit le pointeur de fiche courante (cursorname) pour le résultat **result_id**.

10.62.8 odbc_do

string odbc_do (int **conn_id**, string **query**)
odbc_do() excécute la requête **query** avec la connexion **conn_id**.

10.62.9 odbc_exec

int odbc_exec (int **connection_id**, string **query_string**)
Retourne FALSE en cas d'erreur. 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.

10.62.10 odbc_execute

int odbc_execute (int **result_id**, array **parameters_array**)
Exécute une requête SQL préparée par odbc_prepare(). Retourne TRUE en cas d'exécution réussie, et FALSE sinon. Le tableau de paramètres **parameters_array** ne sert que si vous avez besoin de paramètres votre requête.

10.62.11 odbc_fetch_into

int odbc_fetch_into (int **result_id**, int **rownumber**, array **result_array**)
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.

10.62.12 odbc_fetch_row

int odbc_fetch_row (int **result_id**, int **row_number**)
Si odbc_fetch_row() a réussi, TRUE est retourné. Si 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 prochaîne 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.

10.62.13 odbc_field_name

string odbc_field_name (int **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.

10.62.14 odbc_field_type

string odbc_field_type (int **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.

10.62.15 odbc_field_len

int odbc_field_len (int **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.

10.62.16 odbc_free_result

int odbc_free_result (int **result_id**)
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.*

10.62.17 odbc_longreadlen

int odbc_longreadlen (int **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().*

10.62.18 odbc_num_fields

int odbc_num_fields (int **result_id**)
odbc_num_fields() retourne le nombre de colonnes dans un résultat ODBC. Cette fonction retournera -1 en cas d'erreur. L'argument est un identifiant de résultat valide, retourné par odbc_exec().

10.62.19 odbc_pconnect

int odbc_pconnect (string **dsn**, string **user**, string **password**, int **cursor_type**)
Retourne un identifiant de connexion ODBC ou 0 (FALSE) en cas d'erreur. Cette fonction se comporte de manière similaire à odbc_connect(), mais la connexion ouverte n'est pas vraiment terminée lorsque le script est terminé. Les prochaînes 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.

10.62.20 odbc_prepare

int odbc_prepare (int **connection_id**, string **query_string**)
Prépare une commande pour l'exécution.
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().

10.62.21 odbc_num_rows

int odbc_num_rows (int **result_id**)
odbc_num_rows() retourne le nombre de lignes dans un résultat ODBC. Cette fonction 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 certains.
Note: odbc_num_rows() après un SELECT retournera -1 avec de nombreux pilotes.

10.62.22 odbc_result

string odbc_result (int **result_id**, mixed **field**)
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:

```
$item_3 = odbc_result($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().

10.62.23 odbc_result_all

int odbc_result_all (int **result_id**, string **format**)
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.

10.62.24 odbc_rollback

int odbc_rollback (int **connection_id**)
Annule toutes les transactions sur la connexion **connection_id**. Retourne TRUE en cas de succès, et FALSE en cas d'echec.

10.62.25 odbc_setoption

int odbc_setoption (int **id**, int **function**, int **option**, int **param**)
Cette fonction 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 cette fonction 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 cette fonction 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, cette fonction vous permet d'utiliser PHP sans que votre patron vous pousse à utiliser un produit commecial, 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**.

```
// 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);
```

10.63 URL

10.63.1 base64_decode

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.

10.63.2 base64_encode

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.

10.63.3 parse_url

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".

10.63.4 urldecode

string urldecode (string **str**)
Décode toutes les séquences %## et les remplace par leur valeur. La chaîne ainsi décodée est retournée.

```
$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().

10.63.5 urlencode

string urlencode (string **str**)
Retourne une chaîne dont les caractères non-alphanumériques (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 MIMIE 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 (+). Cette fonction est pratique pour transmettre des informations via une URL. C'est aussi un moyen de passer des informations d'une page à l'autre.

```
echo '<A HREF="mycgi?foo=', urlencode ($userinput), '">';
```

Voir aussi urldecode().

10.64 Sur les variables

10.65 Vmailmgr

Ces fonctions requièrent qmail et le vmailmgr package package de Bruce Guenter.
Pour toutes les fonctions, les deux variables suivants sont définies de la façon suivante :
String vdomain : le nom de votre domaine virtuel (vdomain.com)
String basepwd : le mot de passe de l'utilisateur "réel" qui gère les utilisateurs virtuels.
Ces fonctions retournent un statut, qui correspond à ceux définis dans response.h

- O ok
- 1 mauvais
- 2 erreur
- 3 erreur de connexion

Problème connu : vm_deluser() n'efface pas un utilisateur d'un dossier comme il devrait le faire.
vm_addalias() ne fonctionne actuellement pas correctement.

```
<?php
dl("php3_vmailmgr.so"); //charge la librairie
$vdomain="vdomain.com";
$basepwd="password";
?>
```

10.65.1 vm_adduser

int vm_adduser (string **vdomain**, string **basepwd**, string **newusername**, string **newuserpassword**)
Ajoute une nouvel utilisateur virtuel **newusername**, avec le mot de passe **newuserpassword**.

10.65.2 vm_addalias

int vm_addalias (string **vdomain**, string **basepwd**, string **username**, string **alias**)
Ajoute un nouvel alias vers un utilisateur virtuel. **username** est le nom du compte email, et **alias** est un alias pour cet utilisateur.

10.65.3 vm_passwd

int vm_passwd (string **vdomain**, string **username**, string **password**, string **newpassword**)
Change le mot de passe d'utilisateurs virtuels. **username** est le nom de compte email, **password** est l'ancien mot de passe de l'utilisateur, et **newpassword** son nouveau mot de passe.

10.65.4 vm_delalias

int vm_delalias (string **vdomain**, string **basepwd**, string **alias**)
Supprime un alias.

10.65.5 vm_deluser

int vm_deluser (string **vdomain**, string **username**)
Supprime un utilisateur virtuel.

10.66 WDDX functions

Ces fonctions opèrent avec l'aide de 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.

```
<?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>
```

```
<?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>
```

10.66.1 wddx_serialize_value

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.

10.66.2 wddx_serialize_vars

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.

```
<?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>

10.66.3 wddx_packet_start

int 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.

10.66.4 wddx_packet_end

string wddx_packet_end (int **packet_id**)
wddx_packet_end() () clos un paquet WDDX repéré par son identifiant **packet_id**.

10.66.5 wddx_add_vars

wddx_add_vars (int **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().

10.66.6 wddx_deserialize

mixed wddx_deserialize (string **packet**)
wddx_deserialize() prend une 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 tableau associatifs.

10.67 Analyseur syntaxique XML

10.67.1 Introduction

10.67.1.1 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/.

10.67.1.2 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 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 oú 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 !

10.67.1.3 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 10.67.3 xml_parser_create puis de définir des *points d'entrée* pour chaque événement XML. Les analyseurs XML disposent de quelques 10.67.19 xml_parser_set_option.

Les gestionnaires d'évènements XML sont:

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. @tab
xml_set_character_data_handler() @tab "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. @tab	
xml_set_processing_instruction_handler() @tab Les programmeurs PHP sont habitués aux instructions exécutables (processing instructions ou PIs). <?php ?> est une instruction exécutable oú php est appelé programme cible. Ces instructions sont gérées de manière spécifiques, (sauf le programme cible, qui est réservé à XML). @tab	
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. @tab

xml_set_unparsed_entity_decl_handler() @tab Ce gestionnaire est appelé pour gérer les déclaration des entités non analysés. @tab	
xml_set_notation_decl_handler() @tab Ce gestionnaire est appelé pour gérer les notations. @tab	
xml_set_external_entity_ref_handler() @tab 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 10.67.2.3 XML Entitée externe pour un exemple. @tab	

10.67.1.4 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.

10.67.1.5 Error Codes

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

10.67.1.6 Codage des caractères

L'extension XML de PHP supporte les caractères Unicode 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 10.67.12 xml_parse du fichier par XML. Lors de la 10.67.3 xml_parser_create), 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 suceptible de changer ultérieurement).

10.67.2 Quelques exemples

Voici une liste d'exemple de code PHP qui analyse un document XML.

10.67.2.1 Exemple de structure XML

Ce premier exemple affiche la structure de l'élément de début dans un document avec indentation/

```
$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);
```

10.67.2.2 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.

```
$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);
```

10.67.2.3 XML Entitée 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').

```
$file = "xmltest.xml";
function trustedFile($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("Untrusted PHP code: <i>%s</i>",
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);
?>
<?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>
```

This file is included from `xmltest.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>
```

10.67.3 xml_parser_create

int 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

Cette fonction crée un analyseur XML et retourne une référence sur cet analyseur pour qu'il puisse être utilisé ultérieurement par d'autres fonctions XML. Retourne TRUE ou FALSE.

10.67.4 xml_set_object

void xml_set_object (int **parser**, object **&object**)

Cette fonction 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=\"hallo\">PHP</A>");
?>
```

Note : *xml_set_object() a été ajouté dans PHP 4.0.*

10.67.5 xml_set_element_handler

int xml_set_element_handler (int **parser**, string **startElementHandler** , string **endElementHandler**)

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* (int **parser**, string **name**, string **attribs**)

parser

- Le premier paramètre, *parser*, est une référence sur l'analyseur XML qui appelle cette fonction.

name

- Le deuxième paramètre, **name**, contiens le nom de l'élément qui a provoqué l'appel du gestionnaire. Si l'analyseur gère la 10.67.1.4 Problèmes de casse, cet élément sera en majuscule.

attribs

- Le troisième paramètre, **attribs**, contient un tableau associatif avec les attributs de l'éléments (si 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 10.67.1.4 Problèmes de 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* (int **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 10.67.1.4 Problèmes de casse, cet élément sera en majuscule.

Si un gestionnaire recoit une chaîne vide, ou FALSE, c'est qu'il est en train d'être désactivé.
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.

10.67.6 xml_set_character_data_handler

int xml_set_character_data_handler (int **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* (int **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.
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.

10.67.7 xml_set_processing_instruction_handler

int xml_set_processing_instruction_handler (int **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 éxécutable (?>) 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* (int **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.
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.

10.67.8 xml_set_default_handler

int xml_set_default_handler (int **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* (int **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.
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.

10.67.9 xml_set_unparsed_entity_decl_handler

int xml_set_unparsed_entity_decl_handler (int **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 4.2.2 des spécifications XML 1.0 pour connaître les notations des entités externes.
La fonction **handler** doit accepter six paramètres: *handler* (int **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'entitée 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.
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.

10.67.10 xml_set_notation_decl_handler

int xml_set_notation_decl_handler (int **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 4.2.2 des spécifications XML 1.0 pour connaître les notations des entités externes.
La fonction **handler** doit accepter cinq paramètres: *handler* (int **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 cette entitée 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.
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.

10.67.11 xml_set_external_entity_ref_handler

int xml_set_external_entity_ref_handler (int **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* (int **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 i.e. 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.
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.

10.67.12 xml_parse

int xml_parse (int **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)

- Si 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 retourne 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().

10.67.13 xml_get_error_code

int xml_get_error_code (int **parser**)

parser

- Cette fonction 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.

Cette fonction 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.

10.67.14 xml_error_string

string xml_error_string (int **code**)

code

- Un message d'erreur, issu de xml_get_error_code().

Retourne la chaîne avec un message textuel, décrivant l'erreur **code**, ou FALSE si aucune description n'a été trouvée.

10.67.15 xml_get_current_line_number

int xml_get_current_line_number (int **parser**)

parser

- Une référence sur un analyseur XML valide.

Cette fonction retourne FALSE si **parser** n'est pas valide, ou sinon, retourne le numéro de la ligne en cours d'analyse.

10.67.16 xml_get_current_column_number

int xml_get_current_column_number (int **parser**)

parser

- Une référence sur un analyseur XML valide.

Cette fonction 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.

10.67.17 xml_get_current_byte_index

int xml_get_current_byte_index (int **parser**)

parser

- Une référence sur un analyseur XML valide.

Cette fonction retourne FALSE si **parser** n'est pas valide, ou sinon, retourne l'index de l'octet d'analyse courante de l'analyseur XML.

10.67.18 xml_parser_free

string xml_parser_free (int **parser**)

parser

- Une référence sur un analyseur XML.

Cette fonction retourne FALSE si **parser** n'est pas une référence valide, ou sinon, détruit l'analyseur et retourne TRUE.

10.67.19 xml_parser_set_option

int xml_parser_set_option (int **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.

Cette fonction 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 :

Option	Type de données	Description
XML_OPTION_CASE_FOLDING	entier	Contrôle la gestion de la 10.67.1.4 Problèmes de casse des balises de cet analyseur XML. Par défaut, activé. @tab
XML_OPTION_TARGET_ENCODING	string	Modifie le 10.67.1.6 Codage des caractères 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. @tab

10.67.20 xml_parser_get_option

mixed xml_parser_get_option (int **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

Cette fonction 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

10.67.21 utf8_decode

string utf8_decode (string **data**)

Cette fonction 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.

10.67.22 utf8_encode

string utf8_encode (string **data**)

Cette fonction 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 oú 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 :

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.

10.68 YAZ

Les fonctions YAZ donnent accès aux API YAZ. Le site de ce projet est : http://www.indexdata.dk/yaz/. Plus de détails sur le module phpyaz sont disponbiles à http://www.indexdata.dk/phpyaz/.

PHP/YAZ est beaucoup plus simple à utiliser que les API C, mais moins flexible. L'objectif est de rendre aisée la réalisation d'un client basic. Elle supporte les connexions persistantes sans états, comparables aux connexions SQL disponibles sous PHP. Cela signifie que les sessions sont sans état mais partagées entre les utilisateurs, ce qui évite la répétition inutile des étapes de connexion et d' initiation.

Avant de compiler PHP avec le module PHP/YAZ, vous devez vous munir de la librairie YAZ. Compilez YAZ et installez le. Compilez PHP avec vos modules, et ajoutez l'option --with-yaz. Grosso modo, vous pouvez suivre la procédure suivante :

```
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
```

PHP/YAZ repère les connexions aves des "targets" (Z-Associations). Un entier positif représente l'identifiant de connexion d'une association donnée.
Le script ci dessous montre le fonctionnement d'une recherche paralèlle. Lorsqu'elle est appelée, elle affiche un formulaire de requête (si il n'y a pas d'argument), ou bien recherche les "targets" dans le tableau d'h™tes.

```
$num_hosts = count ($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>';
   }
}
```

10.68.1 yaz_addinfo

int yaz_addinfo (int **id**)
Retourne plus de détails après la dernière erreur survenue. Une chaine vide est retournée si la dernière opération a été réussie, ou bien si aucune autre information n'est disponible.

10.68.2 yaz_close

int yaz_close (int **id**)
Ferme une connexion à un h™te YAZ. L'application ne peut plus utiliser l'identifiant de connexoin **id**.

10.68.3 yaz_connect

int yaz_connect (string **zurl**)
Yaz_connect() prépare une connexion à un "target" Z39.50 target. **zurl** est de la forme host[:port][/database]. Si port est omis, 210 est utilisé. Si database est omis, Default est utilisé. Cette fonctoin 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().

10.68.4 yaz_errno

int yaz_errno (int **id**)
Retourne le numéro d'erreur de la dernière requête. Une valeur positive est retournée si le "target" a retournée un diagnostic. La valeur 0 est retournée si aucune erreur n'est survenue. Une valeur négative indique une erreur sans diagnostic.
Yaz_errno() doit être appelée après chaque requête. (après la fin de yaz_wait()), pour savoir si la transaction a réussi ou échoué.

10.68.5 yaz_error

int yaz_error (int **id**)
Retourne un message d'erreur pour la dernière requête. Une cha”icirc;ne 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().

10.68.6 yaz_hits

int yaz_hits (int **id**)
Yaz_hits() retourne le nombre de résultat de la dernière recherche.

10.68.7 yaz_range

int yaz_range (int **id**, int **start**, int **number**)
Cette fonction 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 cette fonction n'est pas utilisée, **start** vaudra 1 et **number** vaudra 10.
Retourne true en cas de succès; false en cas d'erreur.

10.68.8 yaz_record

int yaz_record (int **id**, int **pos**, string **type**)
Retourne un résultatà la position **pos**, ou une cha”icirc;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 cha”icirc;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 cha”icirc;ne, prête à l'affichage. Si **type** vaut "array", la ligne sera retournée sous la forme d'un tableau structuré.

10.68.9 yaz_search

int yaz_search (int **id**, string **type**, string **query**)
Yaz_search() prépare une recherche sur le "target" 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(), cette fonction n'est pas bloquante, et ne fait que préparer la recherche pour exécution ultérieure, avec yaz_wait().

10.68.10 yaz_syntax

int yaz_syntax (int **id**, string **syntax**)
Cette fonction est utilisée conjointement avec yaz_search() pour spécifier la méthode de lecture des lignes.

10.68.11 yaz_wait

int yaz_wait (int **id**, string **syntax**)

Cette fonction exécute les requêtes préparée par les fonctions yaz_connect(), yaz_search(). yaz_wait() fonctionne en mode bloquant. yaz_wait() se termine lorsque toutes les requêtes se sont terminées (succès ou erreur).

10.69 Compression

Ce module utilise les fonctions de la librairie zlib (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 10.22 Système de fichiers. Mais celles-ci fonctionnent non seulement avec des fichiers compressés, mais aussi des fichiers décompressés (hormis les fonctions utilisant les sockets).

10.69.1 Petit exemple

Ouvre un fichier temporaire, écrit un texte et puis affiche deux fois le contenu.

```
<?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";
?>
```

10.69.2 gzclose

int gzclose (int **zp**)

gzclose() ferme le pointeur **zp**.
Retourne TRUE ou FALSE, suivant le succès ou l'echec.
Le pointeur de fichier compressé doit être valide, et doit référencer un fichier qui a été ouvert par gzopen().

10.69.3 gzeof

int gzeof (int **zp**)

Retourne TRUE si le pointeur interne du fichier compressé 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().

10.69.4 gzfile

array gzfile (string **filename**, int **use_include_path**)

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 7.1.1.13 ini.include-path.
Voir aussi readgzfile(), et gzopen().

10.69.5 gzgetc

string gzgetc (int **zp**)
Retourne une chaîne décompressée, qui contient un caractère unique, lu depuis le fichier référencé par le pointeur **zp**. 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().

10.69.6 gzgets

string gzgets (int **zp**, int **length**)
Retourne une chaîne décompressée, de longueur inférieure ou égale à length - 1 octets, lue depuis de fichier référencé par le pointeur de fichier fp. La lecture s'interromp 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().

10.69.7 gzgetss

string gzgetss (int **zp**, int **length**, string **allowable_tags**)
Identique à gzgets(), mais gzgetss essaie de supprimer toutes les balises HTML et PHP du texte lu. Vous pouvez utiliser le troisième argument optionnel pour indiquer les balises qui ne doivent pas être supprimées. Note : ***allowable_tags*** *a été ajouté dans PHP 3.0.13, PHP4B3.*

Voir aussi gzgets(), gzopen(), et strip_tags().

10.69.8 gzopen

int gzopen (string **filename**, string **mode**, int **use_include_path**)
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 opéraitions de compression/décompression seront transparaentes.
Si l'ouverture échoue, la fonction retourne FALSE.
You can use the optional third parameter and set it to "1", if you want to search for the file in the 7.1.1.13 ini.include-path, too.

```
$fp = gzopen("/tmp/file.gz", "r");
```

Voir aussi gzclose().

10.69.9 gzpassthru

int gzpassthru (int **zp**)
Lit toutes les informations d'un fichier compressé jusqu'à EOF 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 rends inutilisable pour les opérations ultérieures.

10.69.10 gzputs

int gzputs (int **zp**, string **str**, int **length**)
gzputs()est un alias de gzwrite(), et lui est identique en tous points.

10.69.11 gzread

string gzread (int **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.

```
// 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().

10.69.12 gzrewind

int gzrewind (int **zp**)
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().

10.69.13 gzseek

int gzseek (int **zp**, int **offset**)
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.
Retourne 0 en cas de succès, sinon retourne -1. Notez positionner le pointeur au delà de la fin du fichier est une erreur.
Voir aussi gztell() et gzrewind().

10.69.14 gztell

int gztell (int **zp**)
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().

10.69.15 gzwrite

int gzwrite (int **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 7.1.1.17 ini.magic-quotes-runtime sera ignorée et les slashes ne seront pas supprimés de la chaîne **string**.
Voir aussi gzread(), gzopen(), et gzputs().

10.69.16 readgzfile

int readgzfile (string **filename**, int **use_include_path**)
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.
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 7.1.1.13 ini.include-path.
Voir aussi gzpassthru(), gzfile(), et gzopen().

A Débuggeur PHP

A.1 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 peut fonctionner avec un **IDE** ou un éditeur programmable (tel que Emacs).
Comment paramétrer le débuggeur :

1. Réservez un port TCP por le débuggeur dans le fichier 7.1 Le fichier de configuration (7.1.4.2 ini.debugger.port) et activez le (7.1.4.3 ini.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*)", oú *host* est l'IP ou le nom de l'hôte qui supporte le client **TCP**.

Desormais, toutes les alertes, notes, ... seront envoyées sur la socket client, , *même si vous avez inactivé le rapport d'erreur avec* *error_reporting()*.

A.2 Debugger Protocol

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 inclus 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 :

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. @tab
message	Le mesage d'erreur PHP.
location	Nom du fichier, et numéro de ligne, oú l'erreur est survenue. La première occurence 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. @tab
frames	Nombre de frames dans le dump de lap il. Si il y a 4 frames, attendez vous à des information sur 4 niveaux de fonctions. Si la ligne "frame" n'existe pas, la profondeur doit être 0 (une erreur est survenue au niveau général). @tab
function	Nom de la fonction qui a généré l'erreur. Elle sera répétée à chaque niveau de la pile d'appel. @tab
end	Indique au client que le message se termine ici. @tab

- *data*

- Ligne de données.

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)	

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

B Migration de PHP/FI 2.0 à PHP 3.0

B.1 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.

B.2 Start/end tags

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 :

```
<? echo "Ceci est du code PHP/FI 2.0.\n"; >
```

Comme en version 2.0, PHP/FI accepte aussi cette variante :

```
<? 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 à " >. Cependant, si vous souhaitez utiliser XML sur votre serveur, vous aurez surement 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 :

```
<?php echo "This is PHP 3.0 code!\n"; ?>
```

Some people have had problems with editors that don't understand the processing instruction tags at all. Microsoft FrontPage is one such editor, and as a workaround for these, the following variation was introduced as well:

```
<script language="php">
echo "Ceci est du code PHP 3.0!\n";
</script>
```

B.3 if..endif syntax

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 :

```
if ($foo);
echo "oui\n";
elseif ($bar);
echo "presque\n";
else;
echo "non\n";
endif;
```

```
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 point dans toutes les commandes, sauf pour la dernière expression (endif).

B.4 while syntax

Tout comme pour if..endif, la syntaxe des boucles while..endwhile a changée as well:

```
while ($more_to_come);
   ...
endwhile;
```

```
while ($more_to_come):
   ...
endwhile;
```

Attention Si vous utilisez la vieille syntaxe while..endwhile en PHP 3.0, vous obtiendrez une boucle sans fin !

B.5 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 cotés de l'expression, et cela peut produire des résultats inattendus avec les scripts 2.0.

Considérez les lignes suivantes:

```
$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 effectué, 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 variablelist 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:

```
while ((string)$key != "") {
```

B.6 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.

B.7 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.

B.8 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'echec, la plus part des scripts devront probablement être vérifié manuellement, après avoir utilisé le convertisseur 2.0 à 3.0.

```
$fp = fopen($file, "r");
if ($fp == -1);
echo("Impossible d'ouvrir le fichier $file en lecture <br>\n");
endif;
```

```
$fp = @fopen($file, "r") or print("Impossible d'ouvrir le fichier $file en lecture<br>\n");
```

B.9 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 formattage. 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 converti les arguments en nombres, et effectue une addition numérique. Utilisez "." à la place.

```
echo "1" + "1";
```

En PHP 2.0 cela retournerait 11, en PHP 3.0 cela va retourner 2. A la place, faites :

```
echo "1"."1";
```

```
$a = 1;
$b = 1;
echo $a + $b;
```

Cela va afficher 2, tant en PHP 2.0 qu'en 3.0.

```
$a = 1;
$b = 1;
echo $a.$b;
```

Cela va afficher 11 en PHP 3.0.

C Développement PHP

C.1 Adding functions to PHP3

C.1.1 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.

C.1.2 Arguments de fonctions

Les arguments sont toujours de type val. Ce type contient un membre de type union, qui indique le type réél d'argument. De cette façon, si votre fonction prend deux arguments, elle ressemble à ceci :

```
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()'ed 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.

C.1.3 Fonctions à nombre d'arguments variable

Une function peut prendre un nombre variable d'arguments. Si votre fonction peut prendre deux ou trois arguments, utiliser la syntaxe suivante :

```
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;
}
```

C.1.4 Utiliser les arguments d'une fonction

De type de chaque argument est stocké dans le champs pval. Ce champs peut prendre les valeurs suivantes :

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, te 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 suivante :

```
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);  /* Converti une chaîne en LONG ou DOUBLE suivant la chaîne */
```

Ces fonctions convertissent sur place : elles ne retourne aucune valeur.
La valeur de l'argument est enregistrées dans une union. Les membres sont :

- IS_STRING: arg1->value.str.val
- IS_LONG: arg1->value.lval
- IS_DOUBLE: arg1->value.dval

C.1.5 Gestion de la mémoire dans une fonction

Toute la mémoire nécessaire à une fonctoin doit être allouée avec emalloc() ou estrdup(). Ces fonctions ont le gout et l'odeur des classiques malloc() et strdup(). La mémoire doit êtrel 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 assignes 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 leur 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 leur homologues" est que si vous libérez une variable qui n'a pas été créée avec emalloc() ou estrdup(), vous courez droit à la "segmentation fault". Soyez alors extrêmement prudent, et libérez toute votre mémoire inutilisée.
Si vous compilez avec "-DDEBUG", PHP3 affichera la liste de tous les appels à emalloc() et estrdup() mais jamais à efree() lorsque ce intervient dans un script spécifié.

C.1.6 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)

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 3.0 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 etes 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.

```
if (hash_exists(active_symbol_table,"foo",sizeof("foo"))) { existe... }
else { n'existe pas }
```

```
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, utiliser le code suivant.

D'abord, vous devez vérifier qu'il n'existe pas, avec hash_exists() ou hash_find().
Puis, initialisez le tableau :

```
pval arr;
if (array_init(&arr) == FAILURE) { failed... };
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 :

```
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 sur, 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 recoit un pointeur sur un pointeur sur pval, et pas un pointeur sur pval.
Toutes les fonctions d'accès aux hash retourne SUCCES (SUCCES) ou ECHEC (FAILURE), excepté hash_exists(), qui retourne un booléen.

C.1.7 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.

C.1.8 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 réél peut être fait comme suit :
4. 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. Remplisez 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) specify the string length
- 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) - specify the string length

C.1.9 Using the resource list

PHP 3.0 dispose de standard 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 :

```
RESOURCE *resource;
/* ...alloue de la mémoire pour la ressource, et l'acquiert ... */
/* Ajoute la nouvelel ressource dans la liste */
return_value->value.lval = php3_list_insert((void *) resource, LE_RESOURCE_TYPE);
return_value->type = IS_LONG;
```

```
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... */
```

```
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 fonctoin 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.

C.1.10 Utiliesr 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 fonctoin 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 travailel avec les ressources standard, comme mentionné dans la section (9).
2. Ajoutez un autre fonctoin 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 ressource, ce qui fait que le code va fonctionner, et que le prochain appel renvoiera une ressource éxistante. 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 persistante 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 ressource persistantes, il faut ajouter les destructeurs dans les deux listes de ressources, persistante 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 si 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.

C.1.11 Ajouter des directives de configuration à l'exécution

De nombreuses caractéristiques de PHP3 peuvent être configurée à l'éxécution. Ces directives peuvent apparaitre 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 conf Apache. Toutes les directives OR_OPTIONS peuvent être placées n'importe ou, 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.

C.2 Appeler des fonctions utilisateurs

Pour appelrr 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 responsble 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 appelent 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 :

C.2.1 HashTable *function_table

La table de hash dans laquelle le fonction doit être recherchée.

C.2.2 pval *object

Un pointeur sur un objet sur lequel la fonctoin 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'ojbet à la place.

C.2.3 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.

C.2.4 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.

C.2.5 int param_count

Le nombre de paramètre passé à la fonction.

C.2.6 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, ceq ui signifie que si la fonction modifie les arguments, les valeurs originales seront modifiées. Si vous voulez l'éviter, passez une copie à la place.

C.3 Rapport d'erreurs

Pour signaler les erreurs d'une fonction interne, vous devez appelez la fonctoin 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 :

C.3.1 E_NOTICE

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.

C.3.2 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.

C.3.3 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.

C.3.4 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.

C.3.5 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.

C.3.6 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.

Index des fonctions

A

- aspell_check
- aspell_check-raw
- aspell_new
- aspell_suggest
- Atan
- Atan2

B

- base64_decode
- base64_encode
- base_convert
- basename
- bcadd
- bccomp
- bcdiv
- bcmod
- bcmul
- bcpow
- bcscale
- bcsqrt
- bcsub
- bin2hex
- BinDec
- bindtextdomain

C

- Ceil
- chdir
- checkdate
- checkdnsrr
- chgrp
- chmod
- Chop
- chown
- Chr
- chunk_split
- clearstatcache
- closedir
- closelog
- com_get
- com_invoke
- com_load
- com_propget
- com_propput
- com_propset
- com_set
- compact
- connection_aborted
- connection_status

- connection_timeout
- convert_cyr_string
- copy
- Cos
- count
- count_chars
- cpdf_add_annotation
- cpdf_add_outline
- cpdf_arc
- cpdf_begin_text
- cpdf_circle
- cpdf_clip
- cpdf_close
- cpdf_closepath
- cpdf_closepath_fill_stroke
- cpdf_closepath_stroke
- cpdf_continue_text
- cpdf_curveto
- cpdf_end_text
- cpdf_fill
- cpdf_fill_stroke
- cpdf_finalize
- cpdf_finalize_page
- cpdf_global_set_document_limits
- cpdf_import_jpeg
- cpdf_lineto
- cpdf_moveto
- cpdf_open
- cpdf_output_buffer
- cpdf_page_init
- cpdf_place_inline_image
- cpdf_rect
- cpdf_restore
- cpdf_rlineto
- cpdf_rmoveto
- cpdf_rotate
- cpdf_save
- cpdf_save_to_file
- cpdf_scale
- cpdf_set_char_spacing
- cpdf_set_creator
- cpdf_set_current_page
- cpdf_set_font
- cpdf_set_horiz_scaling
- cpdf_set_keywords
- cpdf_set_leading
- cpdf_set_page_animation
- cpdf_set_subject
- cpdf_set_text_matrix

- cpdf_set_text_pos
- cpdf_set_text_rendering
- cpdf_set_text_rise
- cpdf_set_title
- cpdf_set_word_spacing
- cpdf_setdash
- cpdf_setflat
- cpdf_setgray
- cpdf_setgray_fill
- cpdf_setgray_stroke
- cpdf_setlinecap
- cpdf_setlinejoin
- cpdf_setlinewidth
- cpdf_setmiterlimit
- cpdf_setrgbcolor
- cpdf_setrgbcolor_fill
- cpdf_setrgbcolor_stroke
- cpdf_show
- cpdf_show_xy
- cpdf_stringwidth
- cpdf_stroke
- cpdf_text
- cpdf_translate
- crypt
- curl_close
- curl_exec
- curl_init
- curl_setopt
- curl_version
- current
- cybercash_base64_decode
- cybercash_base64_encode
- cybercash_decr
- cybercash_encr

D

- date
- dba_close
- dba_delete
- dba_exists
- dba_fetch
- dba_firstkey
- dba_insert
- dba_nextkey
- dba_open
- dba_optimize
- dba_popen
- dba_replace
- dba_sync

- dbase_add_record
- dbase_close
- dbase_create
- dbase_delete_record
- dbase_get_record
- dbase_get_record_with_names
- dbase_numfields
- dbase_numrecords
- dbase_open
- dbase_pack
- dbase_replace_record
- dblist
- dbmclose
- dbmdelete
- dbmexists
- dbmfetch
- dbmfirstkey
- dbminsert
- dbmnextkey
- dbmopen
- dbmreplace
- dcgettext
- debugger_off
- debugger_on
- DecBin
- DecHex
- DecOct
- define
- defined
- delete
- dgettext
- die
- dir
- dirname
- diskfreespace
- dl
- doubleval

E

- each
- easter_date
- easter_days
- echo
- empty
- end
- ereg
- ereg_replace
- eregi
- eregi_replace

- error_log
- error_reporting
- escapeshellcmd
- eval
- exec
- exit
- Exp
- explode
- extension_loaded
- extract

F

- fclose
- fdf_close
- fdf_create
- fdf_get_file
- fdf_get_status
- fdf_get_value
- fdf_next_field_name
- fdf_open
- fdf_save
- fdf_set_ap
- fdf_set_file
- fdf_set_status
- fdf_set_value
- feof
- fgetc
- fgetcsv
- fgets
- fgetss
- file
- file_exists
- fileatime
- filectime
- filegroup
- fileinode
- filemtime
- fileowner
- fileperms
- filepro
- filepro_fieldcount
- filepro_fieldname
- filepro_fieldtype
- filepro_fieldwidth
- filepro_retrieve
- filepro_rowcount
- filesize
- filetype
- flock

- Floor
- flush
- fopen
- fpassthru
- fputs
- fread
- FrenchToJD
- fseek
- fsockopen
- ftell
- ftp_cdup
- ftp_chdir
- ftp_connect
- ftp_delete
- ftp_fget
- ftp_fput
- ftp_get
- ftp_login
- ftp_mdtm
- ftp_mkdir
- ftp_nlist
- ftp_pasv
- ftp_put
- ftp_pwd
- ftp_quit
- ftp_rawlist
- ftp_rename
- ftp_rmdir
- ftp_site
- ftp_size
- ftp_systype
- func_get_arg
- func_get_args
- func_num_args
- function_exists
- fwrite

G

- get_browser
- get_cfg_var
- get_class_methods
- get_class_vars
- get_current_user
- get_html_translation_table
- get_magic_quotes_gpc
- get_magic_quotes_runtime
- get_meta_tags
- get_object_vars
- getallheaders

- getdate
- getenv
- gethostbyaddr
- gethostbyname
- gethostbynamel
- GetImageSize
- getlastmod
- getmxrr
- getmyinode
- getmypid
- getmyuid
- getprotobyname
- getprotobynumber
- getrandmax
- getrusage
- getservbyname
- getservbyport
- gettext
- gettimeofday
- gettype
- gmdate
- gmmktime
- gmstrftime
- GregorianToJD
- gzclose
- gzeof
- gzfile
- gzgetc
- gzgets
- gzgetss
- gzopen
- gzpassthru
- gzputs
- gzread
- gzrewind
- gzseek
- gztell
- gzwrite

H

- header
- HexDec
- htmlentities
- htmlspecialchars
- hw_Array2Objrec
- hw_Children
- hw_ChildrenObj
- hw_Close
- hw_Connect

- hw_Cp
- hw_Deleteobject
- hw_DocByAnchor
- hw_DocByAnchorObj
- hw_DocumentAttributes
- hw_DocumentBodyTag
- hw_DocumentContent
- hw_DocumentSetContent
- hw_DocumentSize
- hw_EditText
- hw_Error
- hw_ErrorMsg
- hw_Free_Document
- hw_GetAnchors
- hw_GetAnchorsObj
- hw_GetAndLock
- hw_GetChildColl
- hw_GetChildCollObj
- hw_GetChildDocColl
- hw_GetChildDocCollObj
- hw_GetObject
- hw_GetObjectByQuery
- hw_GetObjectByQueryColl
- hw_GetObjectByQueryCollObj
- hw_GetObjectByQueryObj
- hw_GetParents
- hw_GetParentsObj
- hw_GetRemote
- hw_GetRemoteChildren
- hw_GetSrcByDestObj
- hw_GetText
- hw_Identify
- hw_InCollections
- hw_Info
- hw_InsColl
- hw_InsDoc
- hw_InsertDocument
- hw_InsertObject
- hw_mapid
- hw_Modifyobject
- hw_Mv
- hw_New_Document
- hw_Objrec2Array
- hw_OutputDocument
- hw_pConnect
- hw_PipeDocument
- hw_Root
- hw_Unlock
- hw_Username

- ifx_pconnect
- ifx_prepare
- ifx_query
- ifx_textasvarchar
- ifx_update_blob
- ifx_update_char
- ifxus_close_slob
- ifxus_create_slob
- ifxus_open_slob
- ifxus_read_slob
- ifxus_seek_slob
- ifxus_tell_slob
- ifxus_write_slob
- ignore_user_abort
- ImageArc
- ImageChar
- ImageCharUp
- ImageColorAllocate
- ImageColorAt
- ImageColorClosest
- ImageColorExact
- ImageColorResolve
- ImageColorSet
- ImageColorsForIndex
- ImageColorsTotal
- ImageColorTransparent
- ImageCopyResized
- ImageCreate
- ImageCreateFromGif
- ImageDashedLine
- ImageDestroy
- ImageFill
- ImageFilledPolygon
- ImageFilledRectangle
- ImageFillToBorder
- ImageFontHeight
- ImageFontWidth
- ImageGif
- ImageInterlace
- ImageLine
- ImageLoadFont
- ImagePolygon
- ImagePSBBox
- ImagePSEncodeFont
- ImagePSFreeFont
- ImagePSLoadFont
- ImagePSText
- ImageRectangle
- ImageSetPixel

- ImageString
- ImageStringUp
- ImageSX
- ImageSY
- ImageTTFBBox
- ImageTTFText
- imap_8bit
- imap_alerts
- imap_append
- imap_base64
- imap_binary
- imap_body
- imap_check
- imap_clearflag_full
- imap_close
- imap_createmailbox
- imap_delete
- imap_deletemailbox
- imap_errors
- imap_expunge
- imap_fetchbody
- imap_fetchheader
- imap_fetchstructure
- imap_getmailboxes
- imap_getsubscribed
- imap_header
- imap_headers
- imap_last_error
- imap_listmailbox
- imap_listsubscribed
- imap_mail_copy
- imap_mail_move
- imap_mailboxmsginfo
- imap_msgno
- imap_num_msg
- imap_num_recent
- imap_open
- imap_ping
- imap_qprint
- imap_renamemailbox
- imap_reopen
- imap_rfc822_parse_adrlist
- imap_rfc822_write_address
- imap_scanmailbox
- imap_search
- imap_setflag_full
- imap_sort
- imap_status
- imap_subscribe

- imap_uid
- imap_undelete
- imap_unsubscribe
- imap_utf7_decode
- imap_utf7_encode
- implode
- in_array
- intval
- iptcparse
- is_array
- is_dir
- is_double
- is_executable
- is_file
- is_float
- is_int
- is_integer
- is_link
- is_long
- is_object
- is_readable
- is_real
- is_string
- is_writeable
- isset

J

- JDDayOfWeek
- JDMonthName
- JDToFrench
- JDToGregorian
- JDToJewish
- JDToJulian
- JewishToJD
- join
- JulianToJD

K

- key
- krsort
- ksort

L

- ldap_add
- ldap_bind
- ldap_close
- ldap_connect
- ldap_count_entries
- ldap_delete

- ldap_dn2ufn
- ldap_err2str
- ldap_errno
- ldap_error
- ldap_explode_dn
- ldap_first_attribute
- ldap_first_entry
- ldap_free_result
- ldap_get_attributes
- ldap_get_dn
- ldap_get_entries
- ldap_get_values
- ldap_get_values_len
- ldap_list
- ldap_mod_add
- ldap_mod_del
- ldap_mod_replace
- ldap_modify
- ldap_next_attribute
- ldap_next_entry
- ldap_read
- ldap_search
- ldap_unbind
- leak
- link
- linkinfo
- list
- Log
- Log10
- lstat
- ltrim

M

- mail
- max
- mcal_append_event
- mcal_close
- mcal_date_compare
- mcal_date_valid
- mcal_day_of_week
- mcal_day_of_year
- mcal_days_in_month
- mcal_delete_event
- mcal_event_init
- mcal_event_set_alarm
- mcal_event_set_category
- mcal_event_set_class
- mcal_event_set_description
- mcal_event_set_end

- mcal_event_set_recur_daily
- mcal_event_set_recur_monthly_mday
- mcal_event_set_recur_monthly_wday
- mcal_event_set_recur_none
- mcal_event_set_recur_weekly
- mcal_event_set_recur_yearly
- mcal_event_set_start
- mcal_event_set_title
- mcal_fetch_current_stream_event
- mcal_fetch_event
- mcal_is_leap_year
- mcal_list_alarms
- mcal_list_events
- mcal_next_recurrence
- mcal_open
- mcal_snooze
- mcal_store_event
- mcal_time_valid
- mcrypt_cbc
- mcrypt_cfb
- mcrypt_create_iv
- mcrypt_ecb
- mcrypt_get_block_size
- mcrypt_get_cipher_name
- mcrypt_get_key_size
- mcrypt_ofb
- md5
- Metaphone
- method_exists
- mhash
- mhash_count
- mhash_get_block_size
- mhash_get_hash_name
- microtime
- min
- mkdir
- mktime
- msql
- msql_affected_rows
- msql_close
- msql_connect
- msql_create_db
- msql_createdb
- msql_data_seek
- msql_dbname
- msql_drop_db
- msql_dropdb
- msql_error
- msql_fetch_array

- msql_fetch_field
- msql_fetch_object
- msql_fetch_row
- msql_field_seek
- msql_fieldflags
- msql_fieldlen
- msql_fieldname
- msql_fieldtable
- msql_fieldtype
- msql_free_result
- msql_freeresult
- msql_list_dbs
- msql_list_fields
- msql_list_tables
- msql_listdbs
- msql_listfields
- msql_listtables
- msql_num_fields
- msql_num_rows
- msql_numfields
- msql_numrows
- msql_pconnect
- msql_query
- msql_regcase
- msql_result
- msql_select_db
- msql_selectdb
- msql_tablename
- mssql_close
- mssql_connect
- 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_free_result
- mssql_get_last_message
- mssql_min_error_severity
- mssql_min_message_severity
- mssql_num_fields
- mssql_num_rows
- mssql_pconnect
- mssql_query
- mssql_result
- mssql_select_db

N

O

- OCIDefineByName
- OCIError
- OCIExecute
- OCIFetch
- OCIFetchInto
- OCIFetchStatement
- OCIFreeCursor
- OCIFreeStatement
- OCIInternalDebug
- OCILogOff
- OCILogon
- OCINewCursor
- OCINewDescriptor
- OCINLogon
- OCINumCols
- OCIParse
- OCIPLogon
- OCIResult
- OCIRollback
- OCIRowCount
- OCIServerVersion
- OCIStatementType
- OctDec
- odbc_autocommit
- odbc_binmode
- odbc_close
- odbc_close_all
- odbc_commit
- odbc_connect
- odbc_cursor
- odbc_do
- odbc_exec
- odbc_execute
- odbc_fetch_into
- odbc_fetch_row
- odbc_field_len
- odbc_field_name
- odbc_field_type
- odbc_free_result
- odbc_longreadlen
- odbc_num_fields
- odbc_num_rows
- odbc_pconnect
- odbc_prepare
- odbc_result
- odbc_result_all
- odbc_rollback
- odbc_setoption
- opendir

- openlog
- Ora_Bind
- Ora_Close
- Ora_ColumnName
- Ora_ColumnType
- Ora_Commit
- Ora_CommitOff
- Ora_CommitOn
- Ora_Error
- Ora_ErrorCode
- Ora_Exec
- Ora_Fetch
- Ora_GetColumn
- Ora_Logoff
- Ora_Logon
- Ora_Open
- Ora_Parse
- Ora_Rollback
- Ord

P

- pack
- parse_str
- parse_url
- passthru
- pclose
- pdf_add_annotation
- PDF_add_outline
- PDF_arc
- PDF_begin_page
- PDF_circle
- PDF_clip
- PDF_close
- PDF_close_image
- PDF_closepath
- PDF_closepath_fill_stroke
- PDF_closepath_stroke
- PDF_continue_text
- PDF_curveto
- PDF_end_page
- PDF_endpath
- PDF_execute_image
- PDF_fill
- PDF_fill_stroke
- PDF_get_info
- PDF_lineto
- PDF_moveto
- PDF_open
- PDF_open_gif

- PDF_open_jpeg
- PDF_open_memory_image
- PDF_place_image
- PDF_put_image
- PDF_rect
- PDF_restore
- PDF_rotate
- PDF_save
- PDF_scale
- PDF_set_char_spacing
- PDF_set_duration
- PDF_set_font
- PDF_set_horiz_scaling
- PDF_set_info_author
- PDF_set_info_creator
- PDF_set_info_keywords
- PDF_set_info_subject
- PDF_set_info_title
- PDF_set_leading
- PDF_set_parameter
- PDF_set_text_matrix
- PDF_set_text_pos
- PDF_set_text_rendering
- PDF_set_text_rise
- PDF_set_transition
- PDF_set_word_spacing
- PDF_setdash
- PDF_setflat
- PDF_setgray
- PDF_setgray_fill
- PDF_setgray_stroke
- PDF_setlinecap
- PDF_setlinejoin
- PDF_setlinewidth
- PDF_setmiterlimit
- PDF_setrgbcolor
- PDF_setrgbcolor_fill
- PDF_setrgbcolor_stroke
- PDF_show
- PDF_show_boxed
- PDF_show_xy
- PDF_skew
- PDF_stringwidth
- PDF_stroke
- PDF_translate
- pfpro_cleanup
- pfpro_init
- pfpro_process
- pfpro_process_raw

- pfpro_version
- pfsockopen
- pg_Close
- pg_cmdTuples
- pg_Connect
- pg_DBname
- pg_ErrorMessage
- pg_Exec
- pg_Fetch_Array
- pg_Fetch_Object
- pg_Fetch_Row
- pg_FieldIsNull
- pg_FieldName
- pg_FieldNum
- pg_FieldPrtLen
- pg_FieldSize
- pg_FieldType
- pg_FreeResult
- pg_GetLastOid
- pg_Host
- pg_loclose
- pg_locreate
- pg_loopen
- pg_loread
- pg_loreadall
- pg_lounlink
- pg_lowrite
- pg_NumFields
- pg_NumRows
- pg_Options
- pg_pConnect
- pg_Port
- pg_Result
- pg_tty
- php_logo_guid
- phpinfo
- phpversion
- pi
- popen
- pos
- posix_ctermid
- posix_getcwd
- posix_getegid
- posix_geteuid
- posix_getgid
- posix_getgrgid
- posix_getgrnam
- posix_getgroups
- posix_getlogin

- posix_getpgid
- posix_getpgrp
- posix_getpid
- posix_getppid
- posix_getpwnam
- posix_getpwuid
- posix_getrlimit
- posix_getsid
- posix_getuid
- posix_isatty
- posix_kill
- posix_mkfifo
- posix_setgid
- posix_setpgid
- posix_setsid
- posix_setuid
- posix_times
- posix_ttyname
- posix_uname
- pow
- preg_grep
- preg_match
- preg_match_all
- preg_quote
- preg_replace
- preg_split
- prev
- print
- print_r
- printf
- pspell_check
- pspell_mode
- pspell_new
- pspell_runtogether
- pspell_suggest
- putenv

Q

- quoted_printable_decode
- QuoteMeta

R

- rand
- range
- rawurldecode
- rawurlencode
- readdir
- readfile
- readgzfile

- readline
- readline_add_history
- readline_clear_history
- readline_completion_function
- readline_info
- readline_list_history
- readline_read_history
- readline_write_history
- readlink
- recode_file
- recode_string
- register_shutdown_function
- rename
- reset
- rewind
- rewinddir
- rmdir
- round
- rsort

S

- sem_acquire
- sem_get
- sem_release
- serialize
- session_decode
- session_destroy
- session_encode
- session_id
- session_is_registered
- session_module_name
- session_name
- session_register
- session_save_path
- session_start
- session_unregister
- set_file_buffer
- set_magic_quotes_runtime
- set_socket_blocking
- set_time_limit
- setcookie
- setlocale
- settype
- shm_attach
- shm_detach
- shm_get_var
- shm_put_var
- shm_remove
- shm_remove_var

- shuffle
- similar_text
- Sin
- sizeof
- sleep
- snmp_get_quick_print
- snmp_set_quick_print
- snmpget
- snmpset
- snmpwalk
- snmpwalkoid
- sort
- soundex
- split
- sprintf
- sql_regcase
- Sqrt
- srand
- stat
- str_repeat
- str_replace
- strcasecmp
- strchr
- strcmp
- strcspn
- strftime
- strip_tags
- StripCSlashes
- StripSlashes
- stristr
- strlen
- strpos
- strrchr
- strrev
- strrpos
- strspn
- strstr
- strtok
- strtolower
- strtoupper
- strtr
- strval
- substr
- substr_replace
- swf_actiongeturl
- swf_actiongotoframe
- swf_actiongotolabel
- swf_actionnextframe
- swf_actionplay

- swf_actionprevframe
- swf_actionsettarget
- swf_actionstop
- swf_actiontogglequality
- swf_actionwaitforframe
- swf_addbuttonrecord
- swf_addcolor
- swf_closefile
- swf_definebitmap
- swf_definefont
- swf_defineline
- swf_definepoly
- swf_definerect
- swf_definetext
- swf_endbutton
- swf_enddoaction
- swf_endshape
- swf_endsymbol
- swf_fontsize
- swf_fontslant
- swf_fonttracking
- swf_getbitmapinfo
- swf_getfontinfo
- swf_getframe
- swf_labelframe
- swf_lookat
- swf_modifyobject
- swf_mulcolor
- swf_nextid
- swf_oncondition
- swf_openfile
- swf_ortho
- swf_ortho2
- swf_perspective
- swf_placeobject
- swf_polarview
- swf_popmatrix
- swf_posround
- swf_pushmatrix
- swf_removeobject
- swf_rotate
- swf_scale
- swf_setfont
- swf_setframe
- swf_shapearc
- swf_shapecurveto
- swf_shapecurveto3
- swf_shapefillbitmaptile, swf_shapefillbitmaptile
- swf_shapefilloff

- swf_shapefillsolid
- swf_shapelinesolid
- swf_shapelineto
- swf_shapemoveto
- swf_showframe
- swf_startbutton
- swf_startdoaction
- swf_startshape
- swf_startsymbol
- swf_textwidth
- swf_translate
- swf_viewport
- sybase_affected_rows
- sybase_close
- sybase_connect
- sybase_data_seek
- sybase_fetch_array
- sybase_fetch_field
- sybase_fetch_object
- sybase_fetch_row
- sybase_field_seek
- sybase_free_result
- sybase_num_fields
- sybase_num_rows
- sybase_pconnect
- sybase_query
- sybase_result
- sybase_select_db
- symlink
- syslog
- system

T

- Tan
- tempnam
- textdomain
- time
- touch
- trim

U

- uasort
- ucfirst
- ucwords
- uksort
- umask
- uniqid
- unlink
- unpack

- unserialize
- unset
- urldecode
- urlencode
- usleep
- usort
- utf8_decode
- utf8_encode

V

- var_dump
- virtual
- vm_addalias
- vm_adduser
- vm_delalias
- vm_deluser
- vm_passwd

W

- wddx_add_vars
- wddx_deserialize
- wddx_packet_end
- wddx_packet_start
- wddx_serialize_value
- wddx_serialize_vars

X

- xml_error_string
- xml_get_current_byte_index
- xml_get_current_column_number
- xml_get_current_line_number
- xml_get_error_code
- xml_parse
- xml_parser_create
- xml_parser_free
- xml_parser_get_option
- xml_parser_set_option
- xml_set_character_data_handler
- xml_set_default_handler
- xml_set_element_handler
- xml_set_external_entity_ref_handler
- xml_set_notation_decl_handler
- xml_set_object
- xml_set_processing_instruction_handler
- xml_set_unparsed_entity_decl_handler
- xmldoc
- xmldocfile
- xmltree

Y

Z

Index des concepts

A

- Affecte les gestionnaires d'entité non déclaré. .
- Affecte les gestionnaires d'instructions exécutables. .
- Affecte les gestionnaires de caractère bruts.
- Affecte les gestionnaires de début et de fin.
- Affecte les gestionnaires de notation.
- Affecte les options d'un analyseur XML.
- Affecte un type à une variable.
- Affecte une nouvelle date de modification à un fichier.
- Affiche de nombreuses informations sur le PHP.
- Affiche des informations lisibles pour une variable.
- Affiche hw_document.
- Affiche la partie du fichier située après le pointeur du fichier.
- Affiche le frame courant.
- Affiche le frame nommé.
- Affiche le résultat sous la forme d'une table HTML.
- Affiche ou affecte le paramètre "apache request notes".
- Affiche un fichier compressé
- Affiche un fichier.
- Affiche un message et termine le script courant
- Affiche un texte à la position courante.
- Affiche un texte à une position donnée.
- Affiche un texte à une position.
- Affiche un texte dans un rectangle.
- Affiche un texte sur une nouvelle ligne.
- Affiche une chaîne formatée.
- Affiche une chaîne.
- Affiche une ou plusieurs chaînes.
- Ajoute des slashes dans une chaîne, comme en langage C.
- Ajoute un backslash devant tous les caractères méta
- Ajoute un enregistrement dans une base dBase.
- Ajoute un nouvel utilisateur virtuel, avec mot de passe, Ajoute un nouvel utilisateur virtuel, avec mot de passe
- Ajoute un signet à la page courante.
- Ajoute un signet sur la page courante.
- Ajoute un slash devant tous les caractères spéciaux.
- Ajoute une annotation., Ajoute une annotation.
- Ajoute une chaîne dans une boîte au lettre.
- Ajoute une entrée à un dossier LDAP.
- Ajoute une ligne à l'historique
- Ajoute une valeur aux attributs courants.
- Ajouter des variables à un paquet WDDX.
- Aligne les dessins sur le chemin courant.
- Aligne sur le chemin courant.
- Alloue une couleur pour une image.
- Analyse un bloc binaire IPTC http://www.xe.net/iptc/ et recherche les balises simples.
- Analyse une chaîne d'adresse.
- Analyse une chaîne, et en déduit des variables et leur valeur.
- Analyse une requête SQL.
- Analyse une requête.

- Analyse une URL et retourne ses composants.
- Annule les transactions en cours
- Annule une transaction
- Annule une transaction.
- arc cosinus
- arc sinus
- arc tangent
- arc tangent de deux variables
- Arrête l'animation flash.
- Arrondi à l'entier inférieur
- arrondi au nombre supérieur
- Arrondi.
- Attributs d'un objet.
- Attributs de l'objet dans l'ancrage.
- Attributs des ancrages d'un document.
- Attributs des documents fils d'un groupe.
- Attributs des parents.
- Authentification d'une connexion FTP
- Avance d'un frame.
- Avance le pointeur interne d'un tableau

B

- Balise de corps d'un document.

C

- Calcule la clé metaphone d'une chaîne.
- Calcule la similarité de deux chaînes.
- Calcule la valeur soundex d’une chaîne.
- Calcule le nombre de couleur d'une palette.
- Calcule un hash.
- Calcule un md5 avec la chaîne.
- Change de dossier
- Change de dossier, et passe au dossier parent.
- Change l'espacement des caractères.
- Change l'inclinaison de la police courante.
- Change la couleur dans une palette à l'index donné.
- Change la police courante.
- Change la position courante.
- Change la taille de la police.
- Change le "umask" courant.
- Change le codage vectoriel d'un caractère dans une police.
- Change le dossier courant.
- Change le groupe possesseur du fichier.
- Change le groupe propriétaire du fichier.
- Change le mode d'orthographe
- Change le mode du fichier.
- Change le mot de passe d'utilisateurs virtuels.
- Change le nom de session de l'utilisateur actif.
- Change les informations locales.

- Charge un fichier ouvert sur un serveur FTP.
- Charge un fichier sur un serveur FTP.
- Charge un nouveau dictionnaire, Charge un nouveau dictionnaire
- charge une extension PHP à la volée
- Charge une nouvelle police.
- Charge une police PostScript Type 1 depuis un fichier.
- Chemin du dossier courant.
- Choisi l'échelle.
- Choisi l'élévation du texte.
- Choisi l'origine du système de coordonnées.
- Choisi la "miter limit".
- Choisi la couleur grise comme couleur de remplissage et de dessin.
- Choisi la couleur grise comme couleur de remplissage.
- Choisi la couleur rgb comme couleur de dessin et de remplissage.
- Choisi la couleur rgb comme couleur de remplissage., Choisi la couleur rgb comme couleur de remplissage.
- Choisi la distance entre les lignes du textes.
- Choisi la durée de transition entre deux pages.
- Choisi la largeur de ligne.
- Choisi la rotation.
- Choisi le mode de remplissage par texture repétée.
- Choisi le mode de remplissage par texture.
- Choisi le mode par défaut de lecture des valeurs.
- Choisi le mode par défaut des objets BLOB pour toutes les requêtes SELECT.
- Choisi le mode par défaut des objets BYTE.
- Choisi le mode par défaut des objets text.
- Choisi le niveau de qualité haut ou bas.
- Choisi le paramètre linecap.
- Choisi le paramètre linejoin.
- Choisi le point courant.
- Choisi les caractères de remplissage.
- Choisit un niveau de gris comme couleur de dessin et de remplissage.
- Choisit un niveau de gris comme couleur de dessin.
- Choisit un niveau de gris comme couleur de remplissage.
- Choisit une couleur rgb comme couleur de dessin et de remplissage.
- Choisit une couleur rgb comme couleur de dessin.
- Choisit une couleur rgb comme couleur de remplissage.
- Classe dossier
- Clos un paquet WDDX.
- Commence l'analyse d'un fichier XML.
- Commence la définition d'un bouton.
- Commence la déscription d'une liste d'action pour la frame courante.
- Commence une forme complexe.
- Commence une nouvelle page., Commence une nouvelle page.
- Commencer un nouveau paquet WDDX avec une structure
- Compacte des données dans une chaîne binaire.
- Compacte une base dBase.
- Comparaison binaire de chaînes, insensible à la casse.

- Comparaison binaire de chaînes.
- Compare deux dates.
- Compare deux nombres de taille arbitraire.
- Complète la définition de la forme courante.
- Complète un tableau jusqu'à la longueur spécifiée, avec une valeur.
- Compte le nombre d'élément d'un tableau
- Compte le nombre d'entrées d'une recherche.
- Compte le nombre de champs d'une base dBase.
- Compte le nombre de ligne déjà lues dans un résultat.
- Compte le nombre de valeurs dans un tableau
- Compter le nombre d'enregistrements dans une base dBase.
- Connection persistante à un serveur Oracle.
- Connexion à une source
- Consid&rer deux mots accolés comme un composé
- Contenu d'un document.
- Contrôle la situation, l'aparance et la zone active du bouton courant.
- Converti d'octal en décimal.
- Converti de binaire en décimal
- Converti de décimal en binaire
- Converti de décimal en hexadécimal
- converti de décimal en octal
- Converti de hexadécimal en décimal
- Converti la chaîne d’un alphabet cyrillique vers un autre.
- Converti le nombre de jour du calendrier Julien en date du calendrier Julien.
- Converti le nombre de jours du calendrier julien en date du calendrier français républicain
- Converti le nombre de jours du calendrier julien en date du calendrier juif.
- Converti le nombre de jours du calendrier Julien en date du calendrier Julien.
- Converti le nombre de jours du calendrier Julien en date grégorienne.
- Converti les attributs d'un objet en tableau.
- Converti les nouvelles lignes en HTML (<BR≶).
- Converti tous les caractères spéciaux en équivalent HTML., Converti tous les caractères spéciaux en équivalent HTML.
- Converti un ND dans un format plus accessible.
- Converti un nombre en des bases arbitraires.
- Converti un numéro d'erreur LDAP en message d'erreur.
- converti un tableau en un objet.
- Converti un une chaîne UTF-8 en ISO-8859 .
- Converti une chaîne à 8 bits en une chaîne à base64.
- Converti une date du calendrier français républicain en nombre de jours du calendrier julien.
- Converti une date du calendrier juif en nombre de jours du calendrier julien.
- Converti une date grégorienne en nombre de jours du calendrier julien.
- Converti une valeur binaire en hexadécimal
- Convertie des données 8bit en texte UTF-7.

- Convertit un une chaîne ISO-8859-1 en UTF-8.
- Convertit une chaîne à 8 bits en une chaîne à guillemets.
- Convertit une chaîne à guillemets en une chaîne à 8 bits.
- Copie des objets.
- Copie et redimensionne une partie d'une image.
- Copie les messages spécifiés dans une boîte aux lettres.
- Copie un fichier.
- cosinus
- Création d'un analyseur XML.
- Crée ou ouvre un segment de mémoire partagée.
- Crée un arbre d'objet PHP, à partir d'un document XML.
- Crée un dossier., Crée un dossier.
- Crée un fichier avec un nom unique.
- Crée un fichier fifo (first in, first out) (un pipe nommé) .
- Crée un lien symbolique.
- Crée un lien.
- Crée un nouveau document FDF.
- Crée un nouveau document.
- Crée un objet BLOB.
- Crée un objet char.
- Crée un objet de grande taille.
- Crée un objet DOM à partir d'un fichier XML
- Crée un objet DOM pour un document XML.
- Crée un objet SLOB et l'ouvre.
- Crée un processus de pointeur de fichier.
- Crée un tableau
- Crée un tableau contenant les variables et leur valeur
- Crée un tableau contenant un intervalle d'entiers
- Crée un vecteur d'initialisation à partir d'une source aléatoire.
- Crée une base de données dBase.
- Crée une base de données mSQL., Crée une base de données mSQL.
- Crée une base de données MySQL.
- Crée une connexion persistante.
- Crée une nouvelle boîte aux lettres.
- Crée une nouvelle image à partir d'un fichier ou d'une URL.
- Crée une nouvelle image.
- Crypte/décrypte en mode OFB.

D

- Décode les données de session à partir d'une chaîne
- Décode un texte encodé en BASE64.
- Décode une chaîne
- Décode une chaîne en MIME base64
- Décode une chaîne encodée URL.
- Décode une chaîne modifiée UTF-7.
- Décode une chaîne URL.
- Déconditionne des données depuis une chaîne binaire.
- Déconnecte d'un serveur LDAP.
- Déconnection d'un serveur Oracle

- Décrit la librairie dbm utilisée.
- Décrit une transition utilisée pour déclencher une liste d'actions.
- Défini le point de vue de l'utilisateur en coordonnées polaire.
- Définit la couleur transparente.
- Définit un polygone.
- Définit un rectangle.
- Définit un symbole.
- Définit une chaîne de texte.
- Définit une constante.
- Définit une image bitmap.
- Définit une ligne.
- Définit une police.
- Définit une projection orthogonale à 2 dimensions entre les coordonnées utilisateur et le port courant.
- définit une projection orthogonale à 3 dimensions entre les coordonnées utilisateur et le port courant
- Définit une projection orthogonale entre les coordonnées utilisateur et le port courant.
- Définit une transformation de vue.
- Démarre une section de texte.
- Dépile la matrice de transformation.
- Dépile un élément au début d'un tableau
- Dépile un élément de la fin d'un tableau
- Déplace le pointeur courant dans un fichier compressé
- Déplace le pointeur interne de ligne.
- Déplace le pointeur interne de lignes.
- Déplace le pointeur interne de résultat.
- Déplace le pointeur interne.
- Déplace les messages spécifiés dans une boîte aux lettres.
- Déplace un objet.
- Déquote une chaîne quotée avec addcslashes
- Désaffecte une variable.
- Détermine le nombre de décimales par défaut pour les fonctions de précision mathématiques.
- détermine si une extension est chargée ou non.
- Détermine si une variable est affectée., Détermine si une variable est affectée.
- Détermine si une variable est de type double.
- Détermine si une variable est de type float.
- Détermine si une variable est de type int.
- Détermine si une variable est de type integer., Détermine si une variable est de type integer.
- Détermine si une variable est de type object.
- Détermine si une variable est de type real.
- Détermine si une variable est de type string.
- Détermine si une variable est un tableau.
- Détermines le rendu du texte.
- Détruit toutes les données enregistrées d'une session
- Détruit un analyseur XML.

- Détruit un document.
- détruit une image.
- Déverrouille un objet.
- Defface un enregistrement dans une base dBase.
- Demande d'informations d'arbre sur une entité du réseau.
- Dessine le long du chemin.
- Dessine un arc de cercle.
- Dessine un arc.
- Dessine un caractère horizontalement.
- Dessine un caractère verticalement.
- Dessine un cercle., Dessine un cercle.
- Dessine un pixel.
- Dessine un polygone rempli.
- Dessine un polygone.
- Dessine un rectangle rempli.
- Dessine un rectangle., Dessine un rectangle., Dessine un rectangle.
- Dessine un texte avec une police TrueType.
- Dessine un texte sur une image avec une police PostScript Type1.
- Dessine une arc de cercle.
- Dessine une chaîne horizontale.
- Dessine une chaîne verticale.
- Dessine une courbe Bézier cubique.
- Dessine une courbe de Bézier quadratique entre deux points.
- Dessine une courbe., Dessine une courbe.
- Dessine une ellipse partielle.
- Dessine une ligne le long du chemin.
- Dessine une ligne pointillée.
- Dessine une ligne, relativement.
- Dessine une ligne., Dessine une ligne., Dessine une ligne.
- Determine le rendu du texte.
- Determine si un pointeur de fichier est un terminal interactif .
- Divise deux nombres de taille arbitraire.
- Draw a line.
- Dumpe les informations d'une variable.
- Duplique un objet BLOB.

E

- Echappe les méta-caractères Shell.
- Echappement des caractères spéciaux des expressions régulières.
- Eclatement d'une chaîne par expression régulière.
- Ecrit dans un fichier compressé
- Ecrit dans un fichier.
- Ecrit l'historique
- Ecrit la valeur courante de l'option quick_print de la librairie UCD.
- Ecrit un document PDF dans un fichier.
- Ecrit un fichier compressé en mode binaire
- Ecrit un objet de grande taille
- Ecrit une chaîne dans un objet SLOB.
- Ecriture du fichier en mode binaire.

- Efface des objets.
- Efface et remplace une portion de tableau
- Efface l'historique
- Efface le cache de la fonction "stat".
- Efface le résultat de la mémoire.
- Efface tous les messages marqués pour l'effacement.
- Efface un événement dans un calendrier MCAL.
- Efface un événement dans un agenda ICAP.
- Efface un dossier., Efface un dossier.
- Efface un fichier sur un serveur FTP.
- Efface un fichier.
- Efface un objet de grande taille
- Efface une base de données mSQL., Efface une base de données mSQL.
- Efface une base de données MySQL.
- Efface une boîte aux lettres.
- Efface une entrée dans un dossier.
- Efface une entrée.
- Efface une valeur des attributs courants.
- Efface une valeur.
- Efface une variable de la mémoire partagée.
- Effacer
- Effectue une requête partielle pour l'URI spécifiée et renvoie toutes les informations.
- Effectue une rotation.
- Effectue une sous-requête Apache
- Elève un nombre à la puissance n-ième.
- Empile la matrice de transformation courante dans la pile.
- Empile un ou plusieurs éléments à la fin d'un tableau
- Empile un ou plusieurs éléments au début d'un tableau
- Encode les données de session dans une chaîne
- Encode une chaîne en MIME base64.
- Encode une chaîne en URL, selon la RFC1738.
- Encode une chaîne en URL.
- Encrypte une chaîne avec un DES.
- Encrypte/décrypte des données en mode CBC.
- Encrypte/décrypte des données en mode CFB.
- Encrypte/décrypte des données en mode ECB.
- Enlève la marque d'effacement d'un message.
- Enlève les balises HTML et PHP.
- Enlève les espaces de début de chaîne.
- Enlève les espaces de fin de chaîne.
- Enlève les espaces de fin et de fin de chaîne.
- Enlève les slash ajoutés par la fonction addslashes
- Enlève un objet.
- Enregistre l'environnement courant.
- Enregistre un événement dans un agenda ICAP.
- Enregistre un nouvel événement dans un calendrier MCAL.
- Enregistre une fonction de complétion
- Enregistre une fonction pour exécution à l'extinction.

- Enregistre une image dans un fichier PDF pour utilisation ultérieure.
- Enregistre une variable dans la session courante
- Enregistrer plusieurs valeurs dans un paquet WDDX
- Enregistrer une valeur dans un paquet WDDX
- Envoie la commande SITE au serveur.
- Envoie le document PDF dans un buffer mémoire.
- Envoie un cookie
- Envoie un entête HTTP.
- Envoie un message d'erreur.
- Envoie un objet SNMP.
- Envoie un signal à un process.
- Envoie une image vers un navigateur ou un fichier.
- Envoie une requête à une base Sybase.
- Envoie une requête Informix.
- Envoie une requête mSQL
- Envoie une requête MySQL à un serveur MySQL.
- Envoie une requête SQL à un serveur MySQL.
- Envoie une requête SQL.
- Etablit une connexion à un serveur Oracle.
- Etablit une connexion persistante.
- Eteind l'alarme d'un événement., Eteind l'alarme d'un événement.
- Evalue une chaîne comme un script PHP.
- Exécute un programme externe et affiche le résultat brut.
- Exécute un programme externe et affiche le résultat.
- Exécute un programme externe.
- Exécute une commande
- Exécute une commande analysée sur un pointeur Oracle.
- Exécute une fonction sur chacun des membres d'un tableau.
- Exécute une requête
- Exécute une requête mSQL.
- Exécute une requête préparée.
- Exécute une requête SQL déjà préparée.
- Exécute une requête SQL préparée.
- Exécute une requête sur une base Interbase
- Exécute une requête.
- Excécute une session CURL
- Excécute une transaction avec Payflow Pro
- exponentielle
- Expression régulière globale.
- Expression régulière standard., Expression régulière standard.
- Extrait toutes les balises meta d’un fichier, et les retourne sous forme d’un tableau.
- Extrait une portion de tableau

F

- Fait du processus courant un chef de session.
- Fait une liste détaillée de fichier dans un dossier.
- Ferme et clos le chemin.
- Ferme la connexion à l'historique système.

- Ferme la connexion Hyperwave.
- Ferme la connexion MySQL.
- Ferme le chemin courant.
- Ferme le chemin et dessine le long du chemin.
- Ferme le chemin.
- Ferme le fichier courant Shockwave Flash.
- Ferme le fichier et dessine une ligne le long du chemin.
- Ferme le flot ICAP.
- Ferme le pointeur sur le dossier.
- Ferme toutes les connexions ODBC
- Ferme un document PDF.
- Ferme un fichier PDF.
- Ferme un fichier.
- Ferme un objet de grande taille.
- Ferme un objet SLOB.
- Ferme un pointeur Oracle.
- Ferme un pointeur sur un fichier compressé.
- Ferme un processus de pointeur de fichier.
- Ferme une base dBase.
- Ferme une base de données dbm.
- Ferme une base.
- Ferme une connexion à un serveur Informix.
- Ferme une connexion à une base de données Interbase.
- Ferme une connexion FTP.
- Ferme une connexion MCAL.
- Ferme une connexion MS SQL Server.
- Ferme une connexion mSQL.
- Ferme une connexion ODBC.
- Ferme une connexion Oracle.
- Ferme une connexion Sybase.
- Ferme une connexion YAZ
- Ferme une document FDF.
- Ferme une image.
- Ferme une session CURL
- Fills the author field of the info structure.
- Fixe certains paramètres.
- Fixe d'offset d'un champs.
- Fixe l'échelle horizontale du texte.
- Fixe l'élévation du texte.
- Fixe l'alarme de la structure globale.
- Fixe l'animation de la transition entre les pages.
- Fixe l'apparence d'un champs.
- Fixe l'echelle horizontale du texte.
- Fixe l'espacement des caractères., Fixe l'espacement des caractères.
- Fixe l'espacement des mots., Fixe l'espacement des mots.
- Fixe l'identifiant de group de processus.
- Fixe l'offset du pointeur de champs.
- Fixe l'UID effective du processus courant.
- Fixe la catégorie de la structure globale.

- Fixe la classe de la structure globale.
- Fixe la couleur de dessin à un niveau de gris.
- Fixe la couleur globale d'addition (? : the global add color).
- Fixe la couleur globale de multiplication (? : the global multiply color).
- Fixe la couleur pour le style courant de remplissage.
- Fixe la déscription de la structure globale.
- Fixe la date de fin de la structure globale.
- Fixe la distance entre deux lignes.
- Fixe la largeur de ligne.
- Fixe la matrice de texte.
- Fixe la matrice du texte.
- Fixe la page courante.
- Fixe la platitude (flatness)., Fixe la platitude (flatness).
- Fixe la position du texte., Fixe la position du texte.
- Fixe la prochaîne ligne dans le pointeur interne de résultat.
- fixe la récurence annuelle.
- Fixe la récurence hedbomadaire.
- fixe la récurence mensuelle.
- Fixe la récurence quotidienne.
- Fixe la récurence.
- Fixe la valeur d'un champs.
- Fixe la valeur d'une variable d'environnement.
- Fixe la valeur de la clé /F.
- Fixe la valeur de la clé /STATUS.
- Fixe le chemin d'un domaine.
- Fixe le contexte des actions.
- Fixe le créateur d'un document PDF.
- Fixe le domaine par défaut.
- Fixe le fichier courant, ou la position courante.
- Fixe le format de date pour les prochaînes requêtes.
- Fixe le frame courant.
- Fixe le gestionnaire de référence externes.
- Fixe le GID effective du processus courant.
- Fixe le mode de transition entre les pages.
- Fixe le motif de pointillé.
- Fixe le niveau de sévérité des erreurs.
- Fixe le niveau de sévérité des messages d'erreurs.
- Fixe le paramètre linecap.
- Fixe le paramètre linejoin.
- Fixe le paramètre miter limit.
- Fixe le point courant relativement.
- Fixe le point courant.
- Fixe le style courant de ligne.
- Fixe le sujet d'un document PDF.
- Fixe le temps maximum d'exécution d'un script.
- Fixe le titre d'un document PDF.
- fixe le titre de la structure globale.
- Fixe les dates de début et de fin de la structure globale.
- Fixe les limites d'un document PDF.

- Fixe les mot clés d'un document PDF.
- Fixe une option de transfert CURL
- Fonctionnement des expressions régulières.
- Force le premier caractère de chaque mot d’une chaîne en majuscule
- Force le premier caractère d’une chaîne en majuscule.
- Formate un nombre par groupe de milliers.
- Formate une date/heure GMT/CUT en fonction des paramétrages locaux.
- Formate une date/heure GMT/CUT.
- Formate une date/heure locale
- Formate une date/heure locale avec les options locales.
- Fuite de mémoire.

G

- Génére un identifiant unique.
- Génére un message dans l'historique système.
- Génère une meilleure valeur aléatoire.
- Génère une représentation enregistrable d'une valeur.
- Génère une valeur aléatoire.
- Gestion des colonnes de données binaires.
- Gestion des colonnes de type LONG.
- Get result data.

H

- Homothétie.

I

- Identifiant d'objet de l'objet dans l'ancrage.
- Identifiant d'objet des groupes fils.
- Identifiant d'objet des parents.
- Identifiants des ancrages d'un document.
- Identifie un utilisateur.
- ids des documents fils d'un groupe.
- Ignore les actions si le frame n'est pas chargé.
- Import variables into the symbol table from an array
- Imprime le texte à la ligne suivante.
- Imprime un texte à la position courante.
- Imprime un texte avec des options.
- Inactive la validation automatique.
- Inactive le debugger interne de PHP.
- Inactive le remplissage.
- Indique de quoi est capable le navigateur client.
- Indique si le fichier est exécutable.
- Indique si le fichier est un lien symbolique.
- Indique si le fichier est un véritable fichier.
- Indique si le nom de fichier est un dossier.
- Indique si une valeur existe.
- Indique si une variable a été enregistrée dans la session ou pas
- indique un fichier est autorisé en lecture
- Indique un fichier est autorisé en lecture.

- Informations à propos d'une connexion.
- Initialise la structure globale d'un flot.
- Initialise le générateur de nombres aléatoires
- Initialise les données de session
- Initialise un nouveau pointeur vide de LOB/FILE
- Initialise une meilleure valeur aléatoire
- Initialise une session CURL
- Initialise une session Payflow Pro
- Insère ou modifie une variable de la mémoire partagée.
- Insère un document dans un groupe.
- Insère un document.
- Insère un groupe.
- Insère un object record.
- Insère une entrée.
- Insère une valeur.
- Inverse l'ordre des caractères d'une chaîne.

J

- Joue l'animation flash à partir du frame courant.
- Joue un frame puis stoppe.

L

- La plus grand valeur aléatoire possible.
- La plus grande valeur.
- La plus petite valeur.
- Le jour de l'année.
- Le jour de la semaine.
- Lecture du fichier en mode binaire.
- Lecture du pointeur de fiche courante (cursorname).
- Libère la mémoire
- Libère la mémoire occupée par une police PostScript Type 1.
- Libère la mémoire prise par un résultat.
- Libère la mémoire reservée par une requête préparée.
- Libère la mémoire.
- Libère le résultat de la mémoire., Libère le résultat de la mémoire.
- Libère les ressources associées à un résultat
- Libère les ressources prises par un résultat.
- Libère toutes les ressources occupées par un pointeur.
- Libère toutes les ressources occupées par une commande.
- Libère un résultat de la mémoire.
- Libère un résultat.
- Libère un sémaphore.
- Lie les paramètres avec une requête précédemment préparée.
- Lie une variable PHP à un paramètre Oracle.
- Lire le nom du groupe
- Lire un paquet WDDX.
- Lis la liste des boîtes aux lettres, et y recherche une chaîne.
- Lis le corps d'un message.
- Lis les informations à propos de la boîte aux lettres courante.

- Liste des object ids des objets fils.
- Liste des object records des objets fils.
- Liste des utilisateurs actuellement identifiés.
- Liste l'historique
- Liste les bases de données disponibles sur le serveur MySQL.
- Liste les bases de données mSQL sur un serveur., Liste les bases de données mSQL sur un serveur.
- Liste les boîtes aux lettres souscrites.
- Liste les boîtes aux lettres, et retourne le détail pour chacune.
- Liste les boîtes aux lettres.
- Liste les champs d'une table., Liste les champs d'une table.
- Liste les champs du résultat MySQL.
- Liste les champs Informix SQL.
- Liste les propriétés des champs SQL.
- Liste les tables d'une base de données.
- Liste les tables mSQL sur une base de données
- Liste les tables mSQL sur une base de données.
- Liste toutes les boîtes aux lettres souscrites.
- Lit et vérifie un fichier.
- Lit l'attribut suivant., Lit l'attribut suivant.
- Lit l'entête d'un message.
- Lit l'historique
- Lit la clé suivante., Lit la clé suivante.
- Lit la longueur d'un champs., Lit la longueur d'un champs.
- Lit la première clé., Lit la première clé.
- Lit la structure d'un message.
- Lit la totalité d'un fichier compressé dans un tableau.
- Lit la valeur courante de l'option quick_print de la librairie UCD.
- Lit la valeur d'un champs.
- Lit la valeur de la clé /F.
- Lit la valeur de la clé /STATUS.
- Lit le fichier et renvoie le résultat dans un tableau.
- Lit le message d'erreur de l'analyseur XML.
- Lit le nom d'un champs., Lit le nom d'un champs., Lit le nom d'un champs.
- Lit le nom de la base de données courante.
- Lit le nom de la colonne.
- Lit le nom du champs suivant.
- Lit le nom du chiffrement utilisé.
- Lit les données de résultat.
- Lit les données liées à une clé.
- Lit les inforamtions sur le champs.
- Lit les informations d'un champs.
- Lit les informations sur une image.
- Lit les options d'un analyseur XML.
- Lit n bytes d'un objet SLOB.
- Lit toutes les informations restantes d'un fichier compressé
- Lit toutes les lignes d'un tableau, et la met sous la forme d'un tableau HTML.
- Lit un caractère d'un fichier compressé.

- Lit un enregistrement dans une base dBase.
- Lit un enregistrement dans une base, sous la forme d'un tableau associatif.
- Lit un fichier compressé en mode binaire
- Lit un objet de grande taille en totalité.
- Lit un objet de grande taille.
- Lit une entrée du dossier.
- Lit une entrée.
- Lit une ligne
- Lit une ligne comme un tableau.
- Lit une ligne d'un fichier compressé
- Lit une ligne d'un fichier compressé et supprime les balises HTML
- Lit une ligne dans un objet.
- Lit une ligne dans un tableau., Lit une ligne dans un tableau., Lit une ligne dans un tableau.
- Lit une ligne dans une base Interbase
- Lit une ligne dans une base Interbase dans un objet.
- Lit une ligne de résultat, et la place dans un tableau.
- Lit une ligne de résultat.
- Lit une ligne sous la forme d'un objet.
- Lit une ligne sous la forme d'un tableau.
- Lit une représentation enregistrée d'une valeur.
- Lit une valeur dans un résultat.
- Lit une valeur.
- Lit une variable dans la mémoire partagée.
- Lit/écrit diverses variables internes à la librairie
- logarithme en base 10.
- Logarithme naturel

M

- Mélange les éléments d'un tableau
- Marque le fichier pour l'effacement, dans la boîte aux lettres courante.
- Message d'erreur.
- Met en place un buffer sur le pointeur de fichier courant.
- Met tous les caractères en majuscule.
- Met tous les caractères en minuscule.
- Mode auto-validation
- Modifie l'echelle.
- Modifie l'index d'un champs.
- Modifie l'origine du système de coordonées.
- Modifie le contenu d'un objet BLOB.
- Modifie le contenu d'un objet char.
- Modifie le pointeur de fichier.
- Modifie le système de coordonnées.
- Modifie les attributs d'objet record.
- Modifie les paramètres ODBC.
- Modifie un événement dans un calendrier MCAL.
- Modifie un objet.
- Modifie une entrée LDAP.

- Modifie/remplace le contenu d'un document.
- Morcelle une chaîne
- Mot la valeur d'un champs.
- Multiplie deux nombres de taille arbitraire.

N

- Nom de l'utilisateur actuellement identifié.
- Nom de la base de données.
- Nombre de colonnes dans un résultat
- Nombre de ligne dans un résultat.
- Nomme le frame courant.

O

- Object id de la racine.
- Object record de hw_document.
- object records d'un groupe d'enfants.
- Oouvre un nouveau document PDF.
- Optimise une base.
- Options disponibles pour les expressions régulières.
- Ouverture d'un fichier ou d'une URL.
- Ouverture d'une base dBase.
- Ouvre la connexion à l'historique système.
- Ouvre un document FDF.
- Ouvre un dossier, et récupère un pointeur dessus.
- Ouvre un fichier compressé
- Ouvre un flot IMAP vers une boîte aux lettres.
- Ouvre un flot IMAP vers une nouvelle boîte aux lettres.
- Ouvre un nouveau document PDF.
- Ouvre un nouveau fichier Shockwave Flash
- Ouvre un objet de grande taille.
- Ouvre un objet SLOB.
- Ouvre un pointeur Oracle.
- Ouvre une base de données dbm
- Ouvre une base de données.
- Ouvre une connexion à un serveur Informix.
- Ouvre une connexion à un serveur MS SQL server.
- Ouvre une connexion à un serveur MySQL.
- Ouvre une connexion à un serveur Sybase.
- Ouvre une connexion à une base de données Interbase.
- Ouvre une connexion FTP
- Ouvre une connexion Hyperwave.
- Ouvre une connexion ICAP.
- Ouvre une connexion MCAL.
- Ouvre une connexion mSQL.
- Ouvre une connexion Oracle.
- Ouvre une connexion persistante à un serveur Informix.
- Ouvre une connexion persistante à un serveur MS SQL.
- Ouvre une connexion persistante à un serveur mSQL.
- Ouvre une connexion persistante à un serveur MySQL.

- Ouvre une connexion persistante à un serveur Sybase.
- Ouvre une connexion persistante à une base de données Interbase.
- Ouvre une connexion persistante à une base de données.
- Ouvre une connexion persistante à une source de données.
- Ouvre une connexion.
- Ouvre une image créée par les fonctions images PHP.
- Ouvre une image GIF.
- Ouvre une image JPEG., Ouvre une image JPEG.
- Ouvre une socket de connexion Internet ou Unix persistante.
- Ouvre une socket de connexion Internet ou Unix.

P

- Place le pointeur de résultat à un offset donné
- Place un objet sur la scène.
- Place une image dans la page.
- Place une image enregistrée dans la page.
- Places une image dans la page.
- Plus grande valeur aléatoire possible.
- Positionne le pointeur de tableau en fin de tableau
- Positionne un flag sur un message.
- Prépare et exécute une requête SQL.
- Prépare une chaîne pour une recherche par expression régulière insensible à la casse.
- Prépare une commande pour l'exécution
- Prépare une expression régulière pour effectuer une recherche insensible à la casse.
- Prépare une requête pour lier les paramètres et l'éxécuter ultérieurement.

- Prépare une requête SQL pour l'exécution.
- Prépares une recherche
- Process a raw transaction with Payflow Pro
- Puissance

R

- Récupère tous les entêtes des requêtes HTTP.
- Répéte une chaîne.
- Réserve un sémaphore.
- Résolution DNS d'une adresse IP.
- Racine carrée.
- Rassemble plusieurs tableaux
- Reçoit tous les objets SNMP d'un agent.
- Reçoit un objet SNMP.
- Recherche dans tout l'arbre LDAP.
- Recherche dans un seul niveau.
- Recherche la dernière occurrence d’un caractère dans une chaîne
- Recherche la dernière occurrence d’un caractère dans une chaîne., Recherche la dernière occurrence d’un caractère dans une chaîne.
- Recherche la longueur du premier segment de chaîne qui ne corresponde pas au masque donné.

- Recherche la première occurrence d’un caractère.
- Recherche par expression régulière insensible à la casse.
- Recherche un événement dans le calendrier., Recherche un événement dans le calendrier.
- Recherche un message dans le domaine courant.
- Recherche un objet dans un groupe., Recherche un objet dans un groupe.
- Recherche un objet., Recherche un objet.
- Rechercher et remplacer par expression régulière standard.
- Recode de fichier à fichier, en fonction de la requête.
- Recode une chaîne en fonction de la requête.
- Recule d'un frame.
- Recule le pointeur courant de tableau
- Regroupe tous les éléments d’un tableau dans une chaîne, avec une chaîne de jointure., Regroupe tous les éléments d’un tableau dans une chaîne, avec une chaîne de jointure.
- Relâche un segment de mémoire partagée.
- Remet le pointeur interne de tableau au début
- Remplace dans une sous partie de chaîne
- Remplace le domaine courant.
- Remplace le domaine lors d'une recherche.
- Remplace les clés par les valeurs, et les valeurs par les clés
- Remplace ou insère une entrée.
- Remplace toutes les occurrences d’un caractère par un autre.
- Remplace toutes les occurrences d’une chaîne par une autre.
- Remplace un enregistrement dans une base dBase.
- Remplace une valeur des attributs courants par une autre.
- Remplace une valeur.
- Remplacement par expression régulière insensible à la casse.
- Remplacement par expression régulière.
- remplir avec une région avec une couleur spécifique.
- Remplis et dessine le chemin courant.
- Remplis le chemin courant., Remplis le chemin courant.
- Remplis le chemin, dessine le bord et ferme le chemin.
- Remplis le chemin, et dessine le bord.
- Remplis, dessine et ferme le chemin courant.
- Remplit.
- Renomme un fichier sur un serveur FTP.
- Renomme un fichier.
- Renomme une boîte aux lettres.
- Renvoie l'espace disque disponible dans le répertoire.
- Renvoie l'heure à laquelle l'inode a été accédé pour la dernière fois.
- Renvoie la date à laquelle le fichier a été accédé pour la dernière fois.
- Renvoie la date de dernière modification du fichier.
- Renvoie la ligne courant sur laquelle se trouve le pointeur du fichier et élimine les balises HTML
- Renvoie la ligne courante sur laquelle se trouve le pointeur du fichier et cherche dans le résultat les champs CSV
- Renvoie la ligne courante sur laquelle se trouve le pointeur du fichier.
- Renvoie la position du pointeur du fichier.

- Renvoie la racine carrée d'un nombre de taille arbitraire.
- Renvoie la taille du fichier.
- Renvoie le caractère que pointe le pointeur du fichier.
- Renvoie le nom du dossier.
- Renvoie le nom du fichier vers lequel pointe un lien symbolique.
- Renvoie le nom du possesseur du fichier.
- Renvoie le numéro d'inode du fichier.
- Renvoie les informations à propos d'un fichier ou d'un lien symbolique.
- Renvoie les informations à propos d'un fichier.
- Renvoie les informations à propos d'un lien.
- Renvoie les permissions affectées au fichier.
- Replace le pointeur courant au début du fichier
- Replace le pointeur de fichier au début.
- Représente un id globale en un id virtuel local.
- Restaure un environnement sauvé.
- Restaures un environnement.
- Retarde l'exécution en micro-secondes
- Retarde l'exécution.
- Retourne à la première entrée du dossier.
- Retourne chaque paire clé/valeur d'un tableau
- Retourne des informations sur les caractères utilisés dans une chaîne.
- Retourne des informations sur un groupe., Retourne des informations sur un groupe.
- Retourne des informations sur un utilisateur., Retourne des informations sur un utilisateur.
- Retourne l' UID d'un message.
- Retourne l'élément courant d'un tableau
- Retourne l'adresse IP correspondant à un hôte.
- Retourne l'entête d'un message.
- Retourne l'heure actuelle
- Retourne l'ID de l'utilisateur du processus courant.
- Retourne l'id du groupe de processus.
- Retourne l'ID effectif du groupe du processus courant.
- Retourne l'identifiant du groupe de processus.
- Retourne l'identifiant du premier attribut.
- Retourne l'identifiant du processus courant.
- Retourne l'identifiant du processus parent.
- Retourne l'identifiant généré par la dernière requête INSERT.
- retourne l'identifiant maximal de hash.
- Retourne l'index de l'octet courant d'un analyseur XML.
- Retourne l'index de la couleur d'un pixel donné.
- Retourne l'index de la couleur donnée, ou la plus proche possible. .
- Retourne l'index de la couleur donnée.
- Retourne l'index de la couleur la plus proche d'une couleur donnée.
- retourne l'inode du script.
- Retourne l'UID du groupe du processus courant.
- retourne l'UID du propriétaire du script actuel.
- Retourne l'UID effectif de l'utilisateur du processus courant.
- Retourne l'URL d'une animation Shockwave Flash.

- Retourne la configuration actuelle de l'option magic_quotes_runtime.
- Retourne la couleur associée à un index.
- Retourne la date de dernière modification d'un fichier sur un serveur FTP.
- retourne la date de dernière modification de la page.
- Retourne la date/heure
- Retourne la dernière erreur (si elle existe) qui est survenu lors de la dernière requête.
- Retourne la dernière erreur de stmt|conn|global.
- Retourne la hauteur de l'image.
- Retourne la hauteur de la police.
- Retourne la hauteur du A majuscule, et du x minuscule.
- Retourne la largeur d'une image.
- Retourne la largeur de la police.
- Retourne la largeur du texte avec la police courante.
- Retourne la ligne associée.
- Retourne la ligne suivante dans un tableau.
- Retourne la liste d'IP correspondants à un hôte.
- Retourne la liste des arguments d'une fonction
- Retourne la liste des fichiers dans un dossier.
- Retourne la longueur d'un champs.
- Retourne la longueur d'une chaîne.
- Retourne la longueur de la chaîne.
- Retourne la longueur du champs spécifié.
- Retourne la longueur du premier segment qui vérifie le masque.
- Retourne la position courante du pointeur interne
- Retourne la prochaine occurence d'une événement.
- Retourne la structure d'info par défaut d'un document PDF.
- Retourne la table de traduction utilisée par @ref{function.htmlspecialchars , , htmlspecialchars()} et @ref{function.htmlentities , , htmlentities()}.
- Retourne la taille d'un champs.
- Retourne la taille d'un fichier.
- Retourne la taille d'une image GIF, JPG ou PNG.
- Retourne la taille de bloc d'un chiffrement.
- Retourne la taille de bloc du hash.
- Retourne la taille de chaque colonne d’une ligne de résultat.
- Retourne la taille de la chaîne.
- Retourne la taille de la clé d'un chiffrement.
- Retourne la taille de la colonne.
- Retourne la taille imprimée.
- Retourne la taille interne de stockage d'un champs donné.
- Retourne la valeur ASCII du caractère.
- Retourne la valeur d'un champs.
- Retourne la valeur d'une colonne dans une ligne lue
- Retourne la valeur d'une option de PHP
- retourne la valeur de la variable d'environnement.
- Retourne la valeur de la variable, au format chaîne.
- Retourne la valeur de pi
- Retourne la valeur numérique (double) de la variable.

- Retourne la valeur numérique (integer) de la variable.
- Retourne la version courante de CURL
- Retourne la version du Payflow Pro
- Retourne le chemin du terminal.
- Retourne le code d'erreur de la dernière requête Informix.
- Retourne le code d'erreur Oracle.
- Retourne le code d'erreur.
- retourne le configuration actuelle de l'option magic_quotes_gpc.
- Retourne le contenu d'un objet BLOB.
- Retourne le contenu d'un objet char.
- Retourne le contenu de la variable sqlca.sqlerrd[0..5] après une requête.
- Retourne le couple (clé ; valeur) suivant d'une carte donnée.
- Retourne le dernier identifiant d'objet.
- Retourne le dernier message d'erreur du serveur (min_message_severity?).
- Retourne le domaine NIS par défaut.
- Retourne le fichier courant, ou la position courante.
- Retourne le flag d'un champs.
- Retourne le logo
- Retourne le logo de Zend
- Retourne le message d'erreur
- Retourne le message d'erreur de la dernière requête Informix.
- Retourne le message d'erreur Oracle.
- Retourne le message LDAP de la dernière commande LDAP.
- retourne le niveau d'utilisation des ressources.
- Retourne le nom d'hôte correspondant à une IP.
- Retourne le nom d'hôte.
- Retourne le nom d'un champs., Retourne le nom d'un champs.
- Retourne le nom d'une colonne.
- Retourne le nom d'une table à partir d'un nom de champs., Retourne le nom d'une table à partir d'un nom de champs.
- Retourne le nom de device du terminal.
- Retourne le nom de la colonne de résultat.
- Retourne le nom de la machine maître pour une carte.
- Retourne le nom de la table oú se trouve une colonne
- Retourne le nom de la table qui contient le champs spécifié.
- Retourne le nom de login.
- Retourne le nom de protocole associé au numéro de protocole
- Retourne le nom de tty.
- Retourne le nom du dossier courant.
- Retourne le nom du hash.
- Retourne le nom du mois.
- Retourne le nom du possesseur du script courant.
- Retourne le nom du système.
- Retourne le nom d’une colonne
- Retourne le nombre courant de colonne d'un analyseur XML.
- Retourne le nombre courant de colonne d'un analyseur XML. .
- Retourne le nombre d'élément d'un tableau
- Retourne le nombre d'arguments passés à une fonction.

- Retourne le nombre de champs
- Retourne le nombre de champs d'un résultat.
- Retourne le nombre de champs dans un résultat., Retourne le nombre de champs dans un résultat., Retourne le nombre de champs dans un résultat., Retourne le nombre de champs dans un résultat.
- Retourne le nombre de champs dans une base filePro., Retourne le nombre de champs dans une base filePro.
- Retourne le nombre de colonnes dans un résultat
- Retourne le nombre de colonnes dans une requête.
- Retourne le nombre de jour d'un mois.
- Retourne le nombre de jour entre le 21 Mars et Pâques, pour une année donnée.
- Retourne le nombre de ligne affectées.
- Retourne le nombre de ligne d'un résultat.
- Retourne le nombre de lignes affectées lors de la dernière opération SQL.
- Retourne le nombre de lignes affectées par la dernière requête.
- Retourne le nombre de lignes affectées par une requête.
- Retourne le nombre de lignes affectées.
- Retourne le nombre de lignes dans un résultat., Retourne le nombre de lignes dans un résultat., Retourne le nombre de lignes dans un résultat., Retourne le nombre de lignes dans un résultat., Retourne le nombre de lignes dans un résultat.
- Retourne le nombre de lignes.
- Retourne le nombre de message dans la boîte aux lettres courante.
- Retourne le nombre de messages récents dans la boîte aux lettres courante.
- Retourne le nombre de résultat de la dernière recherche
- Retourne le nombre de tuples affectés.
- Retourne le numéro d'erreur
- Retourne le numéro d'erreur LDAP de la dernière commande exécutée.
- Retourne le numéro d'ordre d'une carte.
- Retourne le numéro d'une colonne.
- Retourne le numéro de frame courant.
- retourne le numéro de la version courante de PHP.
- Retourne le numéro de ligne courant d'un analyseur XML.
- Retourne le numéro de message d'erreur de la dernière opération MySQL.

- Retourne le numéro de port associé à un service Internet, et un protocole.
- Retourne le numéro de port.
- retourne le numéro de processus courant.
- Retourne le numéro de protocole associé au nom de protocole
- Retourne le numéro de séquence de message pour un UID donné.
- Retourne le numéro du jour de la semaine.
- Retourne le premier attribut.
- Retourne le premier couple (clé ; valeur) d'une carte donnée.
- Retourne le prochain identifiant d'objet libre.
- Retourne le rectangle entourant un texte et dessiné avec une police PostScript Type1.

- retourne le rectangle entourant un texte et dessiné avec une police TrueType.
- Retourne le reste d'une division entre nombre de taille arbitraire.
- Retourne le sémaphore associé à la colonne spécifiée dans le résultat courant.
- Retourne le service Internet qui correspond au port et protocole.
- Retourne le sid du processus.
- Retourne le texte associée avec l'erreur générée lors de la dernière requête.
- Retourne le timestamp UNIX actuel avec microsecondes.
- Retourne le timestamp UNIX actuel.
- Retourne le timestamp UNIX d'une date GMT.
- Retourne le timestamp UNIX d'une date.
- Retourne le type d'un champs donné par index.
- Retourne le type d'un champs.
- Retourne le type de champs.
- Retourne le type de commande OCI.
- Retourne le type de données d'une colonne.
- Retourne le type de fichier
- Retourne le type de la colonne de résultat.
- Retourne le type de la colonne spécifiée dans le résultat courant.
- Retourne le type de la variable.
- Retourne les ancrages qui pointent sur un objet.
- Retourne les attributs d'une entrée d'un résultat.
- Retourne les attributs, et verrouille l'objet.
- Retourne les bits de status de la connexion.
- Retourne les données de résultat.
- Retourne les données enregistrées dans une colonne, à partir d'un résultat, et retourne un objet.
- Retourne les enregistrements MX d'un hôte.
- Retourne les entêtes de tous les messages d'une boîte aux lettres.
- Retourne les fils d'un document distant.
- Retourne les identifiants du groupe du processus courant.
- Retourne les informations de statut sur une boîte aux lettres autre que la boîte courante.
- Retourne les lignes résultats sous la forme d’un objet.
- Retourne les limites système.
- Retourne les noms des méthodes d'une classe.
- Retourne les options.
- Retourne les valeurs d'un identifiant de résultat.
- Retourne les valeurs d'un tableau
- Retourne les valeurs des attributs d'une classe.
- Retourne les valeurs par défaut des attributs d'une classe.
- Retourne plus de détails après une erreur
- Retourne toutes les alertes (si elles existent) qui sont survenues lors de la dernière requête, ou depuis que la pile d'alerte a été réinitialisée.
- Retourne toutes les clés d'un tableau
- Retourne toutes les entrées d'un résultat.
- Retourne toutes les entrées.

- Retourne toutes les erreurs (si elles existent) qui sont survenus lors de la dernière requête, ou depuis que la pile d'erreur a été réinitialisée.
- Retourne toutes les lignes d'un résultat.
- Retourne toutes les valeurs binaires à partir d'un identifiant de résultat.
- Retourne TRUE si la fonction a été définie.
- Retourne TRUE si le client a abandonné la connexion.
- Retourne TRUE si le script a expiré.
- Retourne un caractère.
- Retourne un champs d'un résultat.
- Retourne un document distant.
- Retourne un document texte., Retourne un document texte.
- Retourne un document.
- Retourne un identifiant de sémaphore.
- Retourne un identifiant de type de serveur FTP.
- Retourne un identifiant positif d'association.
- Retourne un message d'erreur.
- Retourne un ND d'une entrée d'un résultat.
- Retourne un nouveau pointeur à utiliser pour lier les pointeurs de références (ref-cursors)
- Retourne un objet contenant la structure de date pour le flot courant.
- Retourne un résultat
- Retourne un tableau avec les résultat de la recherche.
- Retourne un tableau de message après recherche.
- Retourne un tableau dont les éléments sont classés en sens inverse
- Retourne un timestamp UNIX pour Pâques, à minuit, pour une année donnée.
- Retourne une adresse email proprement formatée, à partir du nom de la boîte aux lettre, de l'hôte, et des informations personnelles.
- Retourne une chaîne contenant les informations de version du serveur.
- Retourne une chaîne formatée.
- Retourne une clé d'un tableau associatif
- Retourne une description de l'erreur
- Retourne une donnée d'une ligne lue.
- Retourne une ligne de résultat sous la forme d'un tableau associatif.
- Retourne une ligne de résultat sous la forme d’un tableau.
- Retourne une ligne de résultat.
- Retourne une ligne sous la forme d'un objet., Retourne une ligne sous la forme d'un objet., Retourne une ligne sous la forme d'un objet.
- Retourne une ligne sous la forme d'un tableau énuméré., Retourne une ligne sous la forme d'un tableau énuméré.
- Retourne une ligne sous la forme d'un tableau.
- Retourne une liste d'événement entre deux dates., Retourne une liste d'événement entre deux dates.
- Retourne une liste d'événements qui ont une alarme prévue à une date., Retourne une liste d'événements qui ont une alarme prévue à une date.
- Retourne une partie de la chaîne.
- Retourne une section extraite du corps d'un message.
- Retourne une tableau contenant la liste des arguments d'une fonction.
- Retourne vrai si une valeur appartient à un tableau

- Rotation de la transformation courante.

S

- Sélectionne la police et sa taille.
- Sélectionne une base de données mSQL., Sélectionne une base de données mSQL.
- Sélectionne une base de données MySQL.
- Sélectionne une base de données Sybase.
- Sélectionne une nouvelle zone pour un dessin ultérieur.
- Sépare le nom du fichier et le nom du dossier.
- Sauve l'environnement courant.
- Sauver un document FDF.
- Scinde un ND en plusieurs composants.
- Scinde une chaîne en morceau, grâce à un délimiteur.
- Scinde une chaîne en plus petits morceaux.
- Scinde une chaîne en un tableau, grâce à une expression régulière.
- Se connecte à un serveur LDAP.
- Se connecte à un serveur Oracle avec une nouvelle connexion. Retourne une nouvelle session.
- Se lie à un serveur LDAP.
- Selectionne la base de données MS SQL.
- Selectionne la police courante, et sa taille.
- send mail
- Sinus
- Souscrit à une boîte aux lettres.
- Soustrait un nombre de taille arbitraire à un autre.
- Spécifie la syntaxe de lecture des lignes
- Spécifie le nombre maximal de résultat à lire
- suggère l'orthographe d'un mot
- Sugg□egrave;re une orthographe
- Supprime la récurence de la structure globale.
- Supprime un alias.
- Supprime un flag sur un message.
- Supprime un objet BLOB.
- Supprime un objet char.
- Supprime un objet SLOB.
- Supprime un segment de mémoire partagée sous Unix.
- Supprime un utilisateur virtuel.
- Supprime une variable dans la session courante
- Synchronise une base de données.
- Synonyme de @ref{function.odbc-exec , , odbc_exec()}

T

- Télécharge un fichier depuis un serveur FTP et le sauve dans un fichier déjà ouvert.
- Télécharge un fichier depuis un serveur FTP.
- Taille d'un document.
- Tangente
- Termine l'action courante.

- Termine la définition de symbole.
- Termine la définition du bouton courant.
- Termine la liaison avec un serveur LDAP.
- Termine la souscription à une boîte aux lettres.
- Termine le script courant.
- Termine un document.
- Termine un flot IMAP.
- Termine une connexion PostgreSQL.
- Termine une page., Termine une page.
- Termine une section de texte.
- Termine une session de Payflow Pro
- Teste la fin d'un fichier compressé.
- Teste la fin du fichier.
- Teste si la valeur d'une colonne est NULL
- Teste si un champs est à NULL.
- Transforme une liste de variables en tableau
- Transforme une variable en tableau
- Translate la transformation courante.
- Trie des messages.
- Trie en ordre inverse
- Trie le tableau
- Trie les clés d'un tableau en utilisant une fonction de comparaison définie par l'utilisateur
- Trie les valeurs d'un tableau en utilisant une fonction de comparaison définie par l'utilisateur
- Trie un tableau en ordre
- Trie un tableau en ordre inverse
- Trie un tableau en sens inverse et suivant les clés
- Trie un tableau en utilisant une fonction de comparaison définie par l'utilisateur.
- Trie un tableau suivant les clés
- Trouve la première occurence d'une chaîne.
- Type de données d'un champs.

U

- Utilisation des ressources.
- Utilise un analyseur XML à l'intérieur d'un objet.
- Utilise une variable PHP pour la phase de définition, dans un SELECT.
- Utilise une variable PHP pour la phase de définition, dans une commande SELECT.

V

- Vérifie le courrier de la boîte aux lettres courante.
- Vérifie qu'un identifiant d'objet est dans un groupe.
- Vérifie qu'une clé existe.
- Vérifie qu'une constante existe.
- Vérifie que l'année est bissextile.
- Vérifie que la méthode existe pour une classe.
- Vérifie que le flot IMAP est toujours actif.

- Vérifie si un fichier existe.
- Vérifie un mot, Vérifie un mot
- Vérifie un mot sans en changer la casse et sans essayer de supprimer les espaces aux extrémités.
- Valeur absolue
- Valide les transactions en cours.
- Valide une date.
- Valide une date/heure.
- Valide une heure.
- Valide une transaction ODBC
- Valide une transaction Oracle.
- Verrouille le fichier.
- Vide les buffers de sorties.