Manuale PHP

Stig Sæther Bakken

Alexander Aulbach

Egon Schmid

Jim Winstead

Lars Torben Wilson

Rasmus Lerdorf

Andrei Zmievski

Jouni Ahto

A cura di

Luca Perugini

Simone Cortesi

Tradotto con la collaborazione di:
Fabio Gandola
Massimo Colombo
Marco De Nittis
Darvin Andrioli
Marco Cucinato
Sergio Marchesini
02-10-2001

Manuale PHP

Stig Sæther Bakken, Alexander Aulbach, Egon Schmid, Jim Winstead, Lars Torben Wilson, Rasmus Lerdorf, Andrei Zmievski, e Jouni Ahto

A cura di Luca Perugini

A cura di Simone Cortesi

Tradotto con la collaborazione di:
Fabio Gandola
Massimo Colombo
Marco De Nittis
Darvin Andrioli
Marco Cucinato
Sergio Marchesini

Pubblicato 02-10-2001

Copyright

Sommario

Prefazione

PHP, che sta per "PHP: Hypertext Preprocessor", è un linguaggio di script immerso nel HTML. Molta della sua sintassi è presa in prestito dai linguaggi C, Java e Perl, a cui sono state aggiunte alcune specifiche caratteristiche del PHP. L'obiettivo del linguaggio è di semplificare il lavoro dei webmaster nella realizzazione di pagine dinamiche.

Occorre notare che, il PHP non è più esclusivamente un prelaboratore di ipertesti. Ora è possibile generare immagini, file PDF o animazioni Flash al volo tramite semplici script PHP. Nel sito PHP-GTK (http://gtk.php.net/) si può vedere come il PHP può essere utilizzato per scrivere applicazioni con interfaccia grafica (GUI Graphic User Interface) lato-client.

Informazioni sul Manuale

Questo manuale è scritto in XML facendo uso di DocBook XML DTD (http://www.nwalsh.com/docbook/xml/) e di DSSSL (http://www.jclark.com/dsssl/) (Document Style and Semantics Specification Language) per la formattazione. Gli strumenti utilizzati per generare la versione del manuale in HTML e TeX sono Jade (http://www.jclark.com/jade/), scritto da James Clark (http://www.jclark.com/bio.htm) e The Modular DocBook Stylesheets (http://nwalsh.com/docbook/dsssl/) scritto da Norman Walsh (http://nwalsh.com/). Per generare il formato WinHelp del manuale, viene utilizzato il Microsoft HTML Help Workshop (http://msdn.microsoft.com/library/en-us/htmlhelp/html/vsconhh1start.asp)

Il manuale è prelevabile nei vari formati e nelle varie lingue, inclusi semplice testo, semplice HTML, PDF, PalmPilot DOC, PalmPilot iSilo e WinHelp, dal sito http://www.php.net/docs.php. I manuali sono aggiornati giornalmente.

Ulteriori informazioni su come prelevare il codice XML di questa documentazione sono reperibili presso http://cvs.php.net/. La documentazione è archiviata nel modulo `phpdoc`.

I. Guida Rapida

Capitolo 1. Introduction

Che cosa è il PHP?

PHP (ufficialmente "PHP: Hypertext Preprocessor") è un linguaggio di scripting lato server immerso nel HTML.

Risposta banale, ma che cosa significa? Un esempio:

Esempio 1-1. Un esempio introduttivo

```
<html>
    <head>
        <title>Esempio</title>
    </head>
    <body>
        <?php echo "Ciao, sono uno script PHP!"; ?>
    </body>
</html>
```

Notate come questo esempio è differente da uno script CGI scritto in altri linguaggi tipo Perl o C -- invece di scrivere un programma con parecchi comandi per produrre HTML, si scrive in HTML con qualche comando immerso per ottenere dei risultati (in questo semplice esempio, la visualizzazione di una frase). Il codice PHP è delimitato da speciali tagche ne indicano l'inizio e la fine e che consentono di passare dal modo HTML al modo PHP.

What distinguishes PHP from something like client-side Javascript is that the code is executed on the server. If you were to have a script similar to the above on your server, the client would receive the results of running that script, with no way of determining what the underlying code may be. You can even configure your web server to process all your HTML files with PHP, and then there's really no way that users can tell what you have up your sleeve.

What can PHP do?

At the most basic level, PHP can do anything any other CGI program can do, such as collect form data, generate dynamic page content, or send and receive cookies.

Perhaps the strongest and most significant feature in PHP is its support for a wide range of databases. Writing a database-enabled web page is incredibly simple. The following databases are currently supported:

Adabas D	InterBase	PostgreSQL
dBase	FrontBase	Solid

Empress	mSQL	Sybase
FilePro (read-only)	Direct MS-SQL	Velocis
IBM DB2	MySQL	Unix dbm
Informix	ODBC	
Ingres	Oracle (OCI7 and OCI8)	

PHP also has support for talking to other services using protocols such as IMAP, SNMP, NNTP, POP3, HTTP and countless others. You can also open raw network sockets and interact using other protocols.

A brief history of PHP

PHP was conceived sometime in the fall of 1994 by Rasmus Lerdorf (mailto:rasmus@php.net). Early non-released versions were used on his home page to keep track of who was looking at his online resume. The first version used by others was available sometime in early 1995 and was known as the Personal Home Page Tools. It consisted of a very simplistic parser engine that only understood a few special macros and a number of utilities that were in common use on home pages back then. A guestbook, a counter and some other stuff. The parser was rewritten in mid-1995 and named PHP/FI Version 2. The FI came from another package Rasmus had written which interpreted html form data. He combined the Personal Home Page tools scripts with the Form Interpreter and added mSQL support and PHP/FI was born. PHP/FI grew at an amazing pace and people started contributing code to it.

It is difficult to give any hard statistics, but it is estimated that by late 1996 PHP/FI was in use on at least 15,000 web sites around the world. By mid-1997 this number had grown to over 50,000. Mid-1997 also saw a change in the development of PHP. It changed from being Rasmus' own pet project that a handful of people had contributed to, to being a much more organized team effort. The parser was rewritten from scratch by Zeev Suraski and Andi Gutmans and this new parser formed the basis for PHP Version 3. A lot of the utility code from PHP/FI was ported over to PHP 3 and a lot of it was completely rewritten.

Today (end-1999) either PHP/FI or PHP 3 ships with a number of commercial products such as C2's StrongHold web server and RedHat Linux. A conservative estimate based on an extrapolation from numbers provided by NetCraft (http://www.netcraft.com/) (see also Netcraft Web Server Survey (http://www.netcraft.com/survey/)) would be that PHP is in use on over 1,000,000 sites around the world. To put that in perspective, that is more sites than run Netscape's flagship Enterprise server on the Internet.

Also as of this writing, work is underway on the next generation of PHP, which will utilize the powerful Zend (http://www.zend.com/) scripting engine to deliver higher performance, and will also support running under webservers other than Apache as a native server module.

Capitolo 2. Installation

Downloading the latest version

The source code, and binary distributions for some platforms (including Windows), can be found at `http://www.php.net/`. We recommend you to choose mirror (http://www.php.net/mirrors.php) nearest to you for downloading the distributions.

Installation on UNIX systems

This section will guide you through the general configuration and installation of PHP on unix systems. Be sure to investigate any sections specific to your platform or web server before you begin the process.

Prerequisite knowledge and software:

- Basic UNIX skills (being able to operate "make" and a C compiler, if compiling)
- An ANSI C compiler (if compiling)
- flex (for compiling)
- bison (for compiling)
- A web server
- Any module specific components (such as gd, pdf libs, etc.)

There are several ways to install PHP for the Unix platform, either with a compile and configure process, or through various pre-packaged methods. This documentation is mainly focused around the process of compiling and configuring PHP.

The initial PHP setup and configuration process is controlled by the use of the commandline options of the `configure` script. This page outlines the usage of the most common options, but there are many others to play with. Check out the Complete list of configure options for an exhaustive rundown. There are several ways to install PHP:

- As an Apache module
- As an fhttpd module
- For use with AOLServer, NSAPI, phttpd, Pi3Web, Roxen, thttpd, or Zeus.
- As a CGI executable

Apache Module Quick Reference

PHP can be compiled in a number of different ways, but one of the most popular is as an Apache module. The following is a quick installation overview.

Esempio 2-1. Quick Installation Instructions for PHP 4 (Apache Module Version)

```
1.  gunzip apache_1.3.x.tar.gz
2.  tar xvf apache_1.3.x.tar
3.  gunzip php-x.x.x.tar.gz
4.  tar xvf php-x.x.x.tar
5.  cd apache_1.3.x
6.  ./configure --prefix=/www
7.  cd ../php-x.x.x
8.  ./configure --with-mysql --with-apache=../apache_1.3.x --enable-track-
vars
9.  make
10. make install
11. cd ../apache_1.3.x
12. ./configure --activate-module=src/modules/php4/libphp4.a
13. make
14. make install
15. cd ../php-x.x.x
16. cp php.ini-dist /usr/local/lib/php.ini
17. Edit your httpd.conf or srm.conf file and add:
      AddType application/x-httpd-php .php

18. Use your normal procedure for restarting the Apache server. (You must
    stop and restart the server, not just cause the server to reload by
    use a HUP or USR1 signal.)
```

Building

When PHP is configured, you are ready to build the CGI executable. The command **make** should take care of this. If it fails and you can’t figure out why, see the Problems section.

Unix/Linux installs

This section contains notes and hints specific to installing PHP on Linux distributions.

Using Packages

Many Linux distributions have some sort of package installation, such as RPM. This can assist in setting up a standard configuration, but if you need to have a different set of features (such as a secure server, or a different database driver), you may need to build php and/or your webserver. If

you are unfamiliar with building and compiling your own software, it is worth checking to see whether somebody has already built a packaged version of PHP with the features you need.

Unix/HP-UX installs

This section contains notes and hints specific to installing PHP on HP-UX systems.

Esempio 2-2. Installation Instructions for HP-UX 10

```
From: paul_mckay@clearwater-it.co.uk
04-Jan-2001 09:49
(These tips are for PHP 4.0.4 and Apache v1.3.9)

So you want to install PHP and Apache on a HP-UX 10.20 box?

1. You need gzip, download a binary distribution from
http://hpux.connect.org.uk/ftp/hpux/Gnu/gzip-1.2.4a/gzip-1.2.4a-sd-
10.20.depot.Z
uncompress the file and install using swinstall

2. You need gcc, download a binary distribution from
http://gatekeep.cs.utah.edu/ftp/hpux/Gnu/gcc-2.95.2/gcc-2.95.2-sd-
10.20.depot.gz
gunzip this file and install gcc using swinstall.

3. You need the GNU binutils, you can download a binary distribution from
http://hpux.connect.org.uk/ftp/hpux/Gnu/binutils-2.9.1/binutils-2.9.1-sd-
10.20.depot.gz
gunzip and install using swinstall.

4. You now need bison, you can download a binary distribution from
http://hpux.connect.org.uk/ftp/hpux/Gnu/bison-1.28/bison-1.28-sd-
10.20.depot.gz
install as above.

5. You now need flex, you need to download the source from one of the
http://www.gnu.org mirrors. It is in the non-gnu directory of the ftp site.
Download the file, gunzip, then tar -
xvf it. Go into the newly created flex
directory and do a ./configure, then a make, and then a make install

If you have errors here, it's probably because gcc etc. are not in your
PATH so add them to your PATH.

Right, now into the hard stuff.
```

```
6. Download the PHP and apache sources.

7. gunzip and tar -xvf them.

We need to hack a couple of files so that they can compile ok.

8. Firstly the configure file needs to be hacked because it seems to lose
track of the fact that you are a hpux machine, there will be a
better way of doing this but a cheap and cheerful hack is to put
    lt_target=hpux10.20
on line 47286 of the configure script.

9. Next, the Apache GuessOS file needs to be hacked. Under
apache_1.3.9/src/helpers change line 89 from
    "echo "hp${HPUXMACH}-hpux${HPUXVER}"; exit 0"
to:
    "echo "hp${HPUXMACH}-hp-hpux${HPUXVER}"; exit 0"

10. You cannot install PHP as a shared object under HP-
UX so you must compile
it as a static, just follow the instructions at the Apache page.

11. PHP and apache should have com-
piled OK, but Apache won't start. you need
to cre-
ate a new user for Apache, eg www, or apache. You then change lines 252
and 253 of the conf/httpd.conf in Apache so that instead of
    User nobody
    Group nogroup
you have something like
    User www
    Group sys

This is because you can't run Apache as nobody under hp-ux.
Apache and PHP should then work.

Hope this helps somebody,
Paul Mckay.
```

Unix/Solaris installs

This section contains notes and hints specific to installing PHP on Solaris systems.

Required software

Solaris installs often lack C compilers and their related tools. The required software is as follows:

- gcc (recommended, other C compilers may work)
- make
- flex
- bison
- m4
- autoconf
- automake
- perl
- gzip
- tar

In addition, you will need to install (and possibly compile) any additional software specific to your configuration (such as Oracle or MySQL.

Using Packages

You can simplify the Solaris install process by using pkgadd to install most of your needed components.

Unix/OpenBSD installs

This section contains notes and hints specific to installing PHP on OpenBSD (http://www.openbsd.org/).

Using Ports

This is the recommended method of installing PHP on OpenBSD, as it will have the latest patches and security fixes applied to it by the maintainers. To use this method, ensure that you have a recent ports tree (http://www.openbsd.org/ports.html). Then simply find out which flavors you wish to install, and issue the **make install** command. Below is an example of how to do this.

Esempio 2-3. OpenBSD Ports Install Example

```
$ cd /usr/ports/www/php4
$ make show VARNAME=FLAVORS
 (choose which flavors you want from the list)
$ env FLAVOR="imap gettext ldap mysql gd" make install
```

```
$ /usr/local/sbin/php4-enable
```

Using Packages

There are pre-compiled packages available for your release of OpenBSD (http://www.openbsd.org/). These integrate automatically with the version of Apache installed with the OS. However, since there are a large number of options (called *flavors*) available for this port, you may find it easier to compile it from source using the ports tree. Read the packages(7) (http://www.openbsd.org/cgi-bin/man.cgi?query=packages) manual page for more information in what packages are available.

Unix/Mac OS X installs

This section contains notes and hints specific to installing PHP on Mac OS X Server.

Using Packages

There are a few pre-packaged and pre-compiled versions of PHP for Mac OS X. This can help in setting up a standard configuration, but if you need to have a different set of features (such as a secure server, or a different database driver), you may need to build PHP and/or your web server yourself. If you are unfamiliar with building and compiling your own software, it's worth checking whether somebody has already built a packaged version of PHP with the features you need. Lightyear Design (http://homepage.mac.com/LightyearDesign/MacOSX/Packages/) offers a pre-built version of PHP for OS X, as does Tenon Intersystems (http://www.tenon.com/products/webten/).

Compiling for OS X server

There are two slightly different versions of Mac OS X, client and server. The following is for OS X Server.

Esempio 2-4. Mac OS X server install

```
1. Get the latest distributions of Apache and PHP
2. Untar them, and run the configure program on Apache like so.
    ./configure --exec-prefix=/usr \
    --localstatedir=/var \
    --mandir=/usr/share/man \
    --libexecdir=/System/Library/Apache/Modules \
    --iconsdir=/System/Library/Apache/Icons \
    --
includedir=/System/Library/Frameworks/Apache.framework/Versions/1.3/Headers \
```

```
    --enable-shared=max \
    --enable-module=most \
    --target=apache

4. You may also want to add this line:
    setenv OPTIM=-O2
    If you want the compiler to do some optimization.

5. Next, go to the PHP 4 source directory and configure it.
    ./configure --prefix=/usr \
    --sysconfdir=/etc \
    --localstatedir=/var \
    --mandir=/usr/share/man \
    --with-xml \
    --with-apache=/src/apache_1.3.12

    If you have any other addiitons (MySQL, GD, etc.), be sure to add
    them here. For the --with-apache string, put in the path to your
    apache source directory, for example "/src/apache_1.3.12".
6. make
7. make install
    This will add a directory to your Apache source directory under
    src/modules/php4.

8. Now, reconfigure Apache to build in PHP 4.
    ./configure --exec-prefix=/usr \
    --localstatedir=/var \
    --mandir=/usr/share/man \
    --libexecdir=/System/Library/Apache/Modules \
    --iconsdir=/System/Library/Apache/Icons \
    --
includedir=/System/Library/Frameworks/Apache.framework/Versions/1.3/Headers \
    --enable-shared=max \
    --enable-module=most \
    --target=apache \
    --activate-module=src/modules/php4/libphp4.a

    You may get a message telling you that libmodphp4.a is out of date.
    If so, go to the src/modules/php4 directory inside your apache
    source directory and run this command:

    ranlib libmodphp4.a

    Then go back to the root of the apache source directory and run the
    above configure command again. That'll bring the link table up to
    date.

9. make
```

```
10. make install

11. copy and rename the php.ini-
dist file to your "bin" directory from your
    PHP 4 source directory:
    cp php.ini-dist /usr/local/bin/php.ini

    or (if your don't have a local directory)

    cp php.ini-dist /usr/bin/php.ini
```

Other examples for Mac OS X client (http://www.stepwise.com/Articles/Workbench/Apache-1.3.14-MacOSX.html) and Mac OS X server (http://www.stepwise.com/Articles/Workbench/Apache-1.3.14-MacOSX.html) are available at Stepwise (http://www.stepwise.com/).

Compiling for MacOS X client

Those tips are graciously provided by Marc Liyanage (http://www.entropy.ch/software/macosx).

The PHP module for the Apache web server included in Mac OS X. This version includes support for the MySQL and PostgreSQL databases.

NOTE: Be careful when you do this, you could screw up your Apache web server!

Do this to install:

- 1. Open a terminal window
- 2. Type "wget http://www.diax.ch/users/liyanage/software/macosx/libphp4.so.gz", wait for download to finish
- 3. Type "gunzip libphp4.so.gz"
- 4. Type "sudo apxs -i -a -n php4 libphp4.so"

Now type "`sudo open -a TextEdit /etc/httpd/httpd.conf`" TextEdit will open with the web server configuration file. Locate these two lines towards the end of the file: (Use the Find command)

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

Remove the two hash marks (#), then save the file and quit TextEdit.

Finally, type "`sudo apachectl graceful`" to restart the web server.

PHP should now be up and running. You can test it by dropping a file into your "Sites" folder which is called "test.php". Into that file, write this line: "`<?php phpinfo() ?>`".

Now open up `127.0.0.1/~your_username/test.php` in your web browser. You should see a status table with information about the PHP module.

Complete list of configure options

> **Nota:** These are only used at compile time. If you want to alter PHP's runtime configuration, please see the chapter on Configuration.

The following is a complete list of options supported by the PHP 3 and PHP 4 `configure` scripts, used when compiling in Unix-like environments. Some are available in PHP 3, some in PHP 4, and some in both, as noted. There are many options the names of which have changed between PHP 3 and PHP 4, but which accomplish the same things. These entries are cross-referenced to each other, so if you have a problem getting your PHP 3-era configuration options to work, check here to see whether the names have changed.

- Database
- Ecommerce
- Graphics
- Miscellaneous
- Networking
- PHP Behaviour
- Server
- Text and language
- XML

Database

`--with-adabas[=DIR]`

PHP 3, PHP 4: Include Adabas D support. DIR is the Adabas base install directory, defaults to /usr/local.

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

`--enable-dba=shared`

PHP 3: Option not available in PHP 3

PHP 4: Build DBA as a shared module

--enable-dbx

PHP 3: Option not available in PHP 3

PHP 4: Include DBX support

--enable-dbase

PHP 3: Option not available; use --with-dbase instead.

PHP 4: Enable the bundled dbase library. No external libraries are required.

--with-dbase

PHP 3: Include the bundled dbase library. No external libraries are required.

PHP 4: Option not available; use --enable-dbase instead.

--with-db2[=DIR]

PHP 3, PHP 4: Include Berkeley DB2 support

--with-db3[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Include Berkeley DB3 support

--with-dbm[=DIR]

PHP 3, PHP 4: Include DBM support

--with-dbmaker[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Include DBMaker support. DIR is the DBMaker base install directory, defaults to where the latest version of DBMaker is installed (such as /home/dbmaker/3.6).

--with-empress[=DIR]

PHP 3, PHP 4: Include Empress support. DIR is the Empress base install directory, defaults to $EMPRESSPATH

--enable-filepro

PHP 3: Option not available; use --with-filepro instead.

PHP 4: Enable the bundled read-only filePro support. No external libraries are required.

--with-filepro

PHP 3: Include the bundled read-only filePro support. No external libraries are required.

PHP 4: Option not available; use --enable-filepro instead.

--with-fbsql[=DIR]

PHP 3: Option not available.

PHP 4: Include FrontBase SQL support. DIR is the FrontBase base install directory, defaults to usual Frontbase install dir. Usual install dirs depends on your OS : Solaris: `/opt/FrontBase`, WinNT: `\usr\FrontBase`, Linux: `/usr/frontbase`, Mac OSX: `/Library/FrontBase`.

--with-gdbm[=DIR]

PHP 3, PHP 4: Include GDBM support

--with-hyperwave

PHP 3, PHP 4: Include Hyperwave support

--with-ibm-db2[=DIR]

PHP 3, PHP 4: Include IBM DB2 support. DIR is the DB2 base install directory, defaults to `/home/db2inst1/sqllib`.

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

--with-informix[=DIR]

PHP 3, PHP 4: Include Informix support. DIR is the Informix base install directory, defaults to nothing.

--with-ingres[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Include Ingres II support. DIR is the Ingres base directory (default /II/ingres)

--with-interbase[=DIR]

PHP 3, PHP 4: Include InterBase support. DIR is the InterBase base install directory, which defaults to `/usr/interbase`.

Interbase functions

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

--with-ldap[=DIR]

PHP 3: Include LDAP support. DIR is the LDAP base install directory. Defaults to `/usr` and `/usr/local`

PHP 4: Include LDAP support. DIR is the LDAP base install directory.

This provides LDAP (Lightweight Directory Access Protocol support). The parameter is the LDAP base install directory, defaults to `/usr/local/ldap`.

More information about LDAP can be found in RFC1777 (http://www.faqs.org/rfcs/rfc1777.html) and RFC1778 (http://www.faqs.org/rfcs/rfc1778.html).

`--with-msql[=DIR]`

PHP 3, PHP 4: Enables mSQL support. The parameter to this option is the mSQL install directory and defaults to `/usr/local/Hughes`. This is the default directory of the mSQL 2.0 distribution. **configure** automatically detects which mSQL version you are running and PHP supports both 1.0 and 2.0, but if you compile PHP with mSQL 1.0, you can only access mSQL 1.0 databases, and vice-versa.

See also mSQL Configuration Directives in the configuration file.

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

`--with-mysql[=DIR]`

PHP 3: Include MySQL support. DIR is the MySQL base install directory, defaults to searching through a number of common places for the MySQL files.

PHP 4: Include MySQL support. DIR is the MySQL base directory. If unspecified, the bundled MySQL library will be used. This option is turned on by default.

See also MySQL Configuration Directives in the configuration file.

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

`--with-ndbm[=DIR]`

PHP 3, PHP 4: Include NDBM support

`--with-ovrimos`

PHP 3, PHP 4: Include Ovrimos support.

`--with-oci8[=DIR]`

PHP 3: Option not available in PHP 3

PHP 4: Include Oracle-oci8 support. Default DIR is ORACLE_HOME.

`--with-oracle[=DIR]`

PHP 3: Include Oracle database support. DIR is Oracle’s home directory, defaults to $ORACLE_HOME.

PHP 4: Include Oracle-oci7 support. Default DIR is ORACLE_HOME.

Includes Oracle support. Has been tested and should be working at least with Oracle versions 7.0 through 7.3. The parameter is the ORACLE_HOME directory. You do not have to specify this parameter if your Oracle environment has been set up.

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

--with-pgsql[=DIR]

PHP 3: Include PostgresSQL support. DIR is the PostgresSQL base install directory, which defaults to `/usr/local/pgsql`.

PHP 4: Include PostgreSQL support. DIR is the PostgreSQL base install directory, which defaults to `/usr/local/pgsql`. Set DIR to shared to build as a dl, or shared,DIR to build as a dl and still specify DIR.

See also Postgres Configuration Directives in the configuration file.

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

--with-solid[=DIR]

PHP 3, PHP 4: Include Solid support. DIR is the Solid base install directory, defaults to /usr/local/solid

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

--with-sybase-ct[=DIR]

PHP 3, PHP 4: Include Sybase-CT support. DIR is the Sybase home directory, defaults to /home/sybase.

See also Sybase-CT Configuration Directives in the configuration file.

--with-sybase[=DIR]

PHP 3, PHP 4: Include Sybase-DB support. DIR is the Sybase home directory, which defaults to `/home/sybase`.

See also Sybase Configuration Directives in the configuration file.

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

--with-openlink[=DIR]

PHP 3, PHP 4: Include OpenLink ODBC support. DIR is the OpenLink base install directory, defaults to /usr/local/openlink. As of PHP 4.0.6 this configure option is no longer valid. Please use the *--with-iodbc* if you wish to use OpenLink Software's ODBC system.

OpenLink Software's home page (http://www.openlinksw.com/)

--with-iodbc[=DIR]

PHP 3, PHP 4: Include iODBC support. DIR is the iODBC base install directory, defaults to `/usr/local`.

This feature was first developed for iODBC Driver Manager, a freely redistributable ODBC driver manager which runs under many flavors of UNIX.

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

--with-custom-odbc[=DIR]

PHP 3, PHP 4: Includes support for an arbitrary custom ODBC library. The parameter is the base directory and defaults to `/usr/local`.

This option implies that you have defined CUSTOM_ODBC_LIBS when you run the configure script. You also must have a valid odbc.h header somewhere in your include path. If you don't have one, create it and include your specific header from there. Your header may also require some extra definitions, particularly when it is multiplatform. Define them in CFLAGS.

For example, you can use Sybase SQL Anywhere on QNX as following:

```
CFLAGS=-DODBC_QNX LDFLAGS=-lunix CUSTOM_ODBC_LIBS="-ldblib -lodbc"
./configure --with-custom-odbc=/usr/lib/sqlany50
```

--disable-unified-odbc

PHP 3: Disable unified ODBC support. Only applicable if iODBC, Adabas, Solid, Velocis or a custom ODBC interface is enabled.

PHP 4: Option not available in PHP 4

The Unified ODBC module, which is a common interface to all the databases with ODBC-based interfaces, such as Solid, IBM DB2 and Adabas D. It also works for normal ODBC libraries. Has been tested with iODBC, Solid, Adabas D, IBM DB2 and Sybase SQL Anywhere. Requires that one (and only one) of these extensions or the Velocis extension is enabled, or a custom ODBC library specified. This option is only applicable if one of the following options is used: --with-iodbc, --with-solid, --with-ibm-db2, --with-adabas, --with-velocis, or --with-custom-odbc.

See also Unified ODBC Configuration Directives in the configuration file.

--with-unixODBC[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Include unixODBC support. DIR is the unixODBC base install directory, defaults to /usr/local.

--with-velocis[=DIR]

PHP 3, PHP 4: Include Velocis support. DIR is the Velocis base install directory, defaults to /usr/local/velocis.

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

Ecommerce

--with-ccvs[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Compile CCVS support into PHP 4. Please specify your CCVS base install directory as DIR.

--with-cybermut[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Include Cybermut support into PHP 4. DIR is the Cybermut SDK directory, which contains both `libcm-mac.a` and `cm-mac.h`.

--with-mck[=DIR]

PHP 3: Include Cybercash MCK support. DIR is the cybercash mck build directory, which defaults to `/usr/src/mck-3.2.0.3-linux`. For help, look in `extra/cyberlib`.

PHP 4: Option not available; use --with-cybercash instead.

--with-cybercash[=DIR]

PHP 3: Option not available; use --with-mck instead.

PHP 4: Include CyberCash support. DIR is the CyberCash MCK install directory.

--with-pfpro[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Include Verisign Payflow Pro support

Graphics

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

PHP 3: Option not available in PHP 3

PHP 4: Include support for FreeType2 (experimental).

--with-gd[=DIR]

PHP 3: Include GD support (DIR is GD's install dir).

PHP 4: Include GD support (DIR is GD's install dir). Set DIR to shared to build as a dl, or shared,DIR to build as a dl and still specify DIR.

--without-gd

PHP 3, PHP 4: Disable GD support.

--with-imagick[=DIR]

PHP 3: Include ImageMagick support. DIR is the install directory, and if left out, PHP will try to find it on its own. [experimental]

PHP 4: Option not available in PHP 4

--with-jpeg-dir[=DIR]

PHP 3: jpeg dir for pdflib 2.0

PHP 4: jpeg dir for pdflib 3.x and 4.x

--with-png-dir[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: png dir for pdflib 3.x and 4.x

--enable-t1lib

PHP 3: Enable t1lib support.

PHP 4: Option not available; use --with-t1lib instead.

--with-t1lib[=DIR]

PHP 3: Option not available; use --enable-t1lib instead.

PHP 4: Include T1lib support.

--with-tiff-dir[=DIR]

PHP 3: tiff dir for pdflib 2.0

PHP 4: tiff dir for pdflib 3.x and 4.x

--with-ttf[=DIR]

PHP 3, PHP 4: Include FreeType support

--with-xpm-dir[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: xpm dir for gd-1.8+

Miscellaneous

These are being classified over time, where appropriate.

--with-gmp

PHP 3, PHP 4 : Include GMP support.

--disable-bcmath

PHP 3: Compile without BC arbitrary precision math functions. These functions allow you to operate with numbers outside of the ranges allowed by regular integers and floats; see BCMath Arbitrary Precision Mathematics Functions for more information.

PHP 4: Option not available; bcmath is not compiled in by default. Use --enable-bcmath to compile it in.

--disable-display-source

PHP 3: Compile without displaying source support

PHP 4: Option not available in PHP 4

--disable-libtool-lock

PHP 3: Option not available in PHP 3

PHP 4: avoid locking (might break parallel builds)

--disable-pear

PHP 3: Option not available in PHP 3

PHP 4: Do not install PEAR

--disable-pic

PHP 3: Option not available in PHP 3

PHP 4: Disable PIC for shared objects

--disable-posix

PHP 3: Option not available in PHP 3; use --without-posix instead.

PHP 4: Disable POSIX-like functions

`--enable-pcntl`

PHP 3: Option not available in PHP 3

PHP 4: Enable process control functions. (fork, waitpid, signal, et. al.)

`--disable-rpath`

PHP 3: Option not available in PHP 3

PHP 4: Disable passing additional runtime library search paths

`--disable-session`

PHP 3: Option not available in PHP 3

PHP 4: Disable session support

`--enable-bcmath`

PHP 3: Option not available in PHP 3; bcmath is compiled in by default. Use --disable-bcmath to disable it.

PHP 4: Compile with bc style precision math functions. Read README-BCMATH for instructions on how to get this module installed. These functions allow you to operate with numbers outside of the ranges allowed by regular integers and floats; see BCMath Arbitrary Precision Mathematics Functions for more information.

`--enable-c9x-inline`

PHP 3: Option not available in PHP 3

PHP 4: Enable C9x-inline semantics

`--enable-calendar`

PHP 3: Option not available in PHP 3

PHP 4: Enable support for calendar conversion

`--enable-debug`

PHP 3, PHP 4: Compile with debugging symbols.

`--enable-debugger`

PHP 3: Compile with remote debugging functions

PHP 4: Option not available in PHP 4

`--enable-discard-path`

PHP 3, PHP 4: If this is enabled, the PHP CGI binary can safely be placed outside of the web tree and people will not be able to circumvent .htaccess security.

`--enable-dmalloc`

PHP 3, PHP 4: Enable dmalloc

`--enable-exif`

PHP 3: Option not available in PHP 3

PHP 4: Enable exif support

`--enable-experimental-zts`

PHP 3: Option not available in PHP 3

PHP 4: This will most likely break your build

`--enable-fast-install[=PKGS]`

PHP 3: Option not available in PHP 3

PHP 4: optimize for fast installation [default=yes]

`--enable-force-cgi-redirect`

PHP 3, PHP 4: Enable the security check for internal server redirects. You should use this if you are running the CGI version with Apache.

`--enable-inline-optimization`

PHP 3: Option not available in PHP 3

PHP 4: If you have much memory and are using gcc, you might try this.

`--enable-libgcc`

PHP 3: Option not available in PHP 3

PHP 4: Enable explicitly linking against libgcc

`--enable-maintainer-mode`

PHP 3, PHP 4: enable make rules and dependencies not useful (and sometimes confusing) to the casual installer

`--enable-mbstr-enc-trans`

PHP 4: enable http input character automatic detection and translation for multi-byte character encodings.

Attenzione

This switch is only available for PHP 4.0.6 or higher.

--enable-mbstring

PHP 4: enable multi-byte character encoding related functions.

Attenzione

This switch is only available for PHP 4.0.6 or higher.

--enable-memory-limit

PHP 3, PHP 4: Compile with memory limit support. [default=no]

--enable-safe-mode

PHP 3, PHP 4: Enable safe mode by default.

--enable-satellite

PHP 3: Option not available in PHP 3

PHP 4: Enable CORBA support via Satellite (Requires ORBit)

--enable-shared[=PKGS]

PHP 3: Option not available in PHP 3

PHP 4: build shared libraries [default=yes]

--enable-sigchild

PHP 3, PHP 4: Enable PHP's own SIGCHLD handler.

--enable-static[=PKGS]

PHP 3: Option not available in PHP 3

PHP 4: build static libraries [default=yes]

--enable-sysvsem

PHP 3, PHP 4: Enable System V semaphore support.

--enable-sysvshm

PHP 3, PHP 4: Enable the System V shared memory support

--enable-trans-sid

PHP 3: Option not available in PHP 3

PHP 4: Enable transparent session id propagation

--with-cdb[=DIR]

PHP 3, PHP 4: Include CDB support

--with-config-file-path=PATH

PHP 3: Sets the path in which to look for php3.ini. Defaults to `/usr/local/lib`

PHP 4: Sets the path in which to look for `php.ini`. Defaults to `/usr/local/lib`

--with-cpdflib[=DIR]

PHP 3: Include ClibPDF support. DIR is the ClibPDF install directory, defaults to /usr/local.

PHP 4: Include cpdflib support (requires cpdflib >= 2). DIR is the cpdfllib install directory, defaults to /usr.

--with-esoob[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Include Easysoft OOB support. DIR is the OOB base install directory, defaults to /usr/local/easysoft/oob/client.

--with-exec-dir[=DIR]

PHP 3, PHP 4: Only allow executables in DIR when in safe mode defaults to /usr/local/php/bin

--with-fdftk[=DIR]

PHP 3, PHP 4: Include fdftk support. DIR is the fdftk install directory, defaults to /usr/local.

--with-gnu-ld

PHP 3: Option not available in PHP 3

PHP 4: assume the C compiler uses GNU ld [default=no]

--with-icap[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Include ICAP support.

--with-imap[=DIR]

PHP 3, PHP 4: Include IMAP support. DIR is the IMAP include and c-client.a directory.

--with-imsp[=DIR]

PHP 3: Include IMSP support (DIR is IMSP's include dir and libimsp.a dir).

PHP 4: Option not available in PHP 4

--with-java[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Include Java support. DIR is the base install directory for the JDK. This extension can only be built as a shared dl.

--with-kerberos[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Include Kerberos support in IMAP.

--with-mcal[=DIR]

PHP 3, PHP 4: Include MCAL support.

--with-mcrypt[=DIR]

PHP 3, PHP 4: Include mcrypt support. DIR is the mcrypt install directory.

--with-mhash[=DIR]

PHP 3, PHP 4: Include mhash support. DIR is the mhash install directory.

--with-mm[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Include mm support for session storage

--with-mod_charset

PHP 3, PHP 4: Enable transfer tables for mod_charset (Rus Apache).

--with-pdflib[=DIR]

PHP 3: Include pdflib support (tested with 0.6 and 2.0). DIR is the pdflib install directory, which defaults to `/usr/local`.

PHP 4: Include pdflib 3.x/4.x support. DIR is the pdflib install location, which defaults to `/usr/local`.

PHP 4 and PDFlib 3.x/4.x require that you have the JPEG and TIFF libraries available. When compiling PDFlib support, use the --with-jpeg-dir and --with-tiff-dir configure options. You may wish to additionally specify the --with-png-dir and --with-zlib-dir configure options to compile PNG and Zlib support into the PDFlib extension.

`--enable-shared-pdflib`

PHP 3, PHP 4: Activate pdflib as shared library.

`--with-readline[=DIR]`

PHP 3: Option not available in PHP 3

PHP 4: Include readline support. DIR is the readline install directory.

`--with-regex=TYPE`

PHP 3: Option not available in PHP 3

PHP 4: regex library type: system, apache, php

`--with-servlet[=DIR]`

PHP 3: Option not available in PHP 3

PHP 4: Include servlet support. DIR is the base install directory for the JSDK. This SAPI requires that the Java extension be built as a shared dl.

`--with-ming`

PHP 3: Option not available in PHP 3

PHP 4: Include Flash 4 support, with Ming

`--with-swf[=DIR]`

PHP 3: Option not available in PHP 3

PHP 4: Include swf support

`--with-system-regex`

PHP 3: Do not use the bundled regex library

PHP 4: (deprecated) Use system regex library

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

PHP 3: Option not available in PHP 3

PHP 4: Use GNU Pth.

`--with-tsrm-pthreads`

PHP 3: Option not available in PHP 3

PHP 4: Use POSIX threads (default)

--with-x

PHP 3: use the X Window System

PHP 4: Option not available in PHP 4

--with-bz2[=DIR]

PHP 4: Include support bzip2. DIR is the bzip2 install dir.

--with-zlib-dir[=DIR]

PHP 3: zlib dir for pdflib 2.0 or include zlib support

PHP 4: zlib dir for pdflib 3.x/4.x or include zlib support

--with-zlib[=DIR]

PHP 3, PHP 4: Include zlib support (requires zlib >= 1.0.9). DIR is the zlib install directory, defaults to /usr.

--with-zip[=DIR]

PHP 4: Include zip support (requires zziplib >= 0.10.6). DIR is the zziplib install directory, defaults to /usr/local.

The latest version of zziplib can be found at http://zziplib.sourceforge.net/.

--without-pcre-regex

PHP 3: Don’t include Perl Compatible Regular Expressions support

PHP 4: Do not include Perl Compatible Regular Expressions support. Use --with-pcre-regex=DIR to specify DIR where PCRE’s include and library files are located, if not using bundled library.

--without-posix

PHP 3: Don’t include POSIX functions

PHP 4: Option not available in PHP 4; use --disable-posix instead.

Networking

--with-curl[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Include CURL support

`--enable-ftp`

PHP 3: Option not available; use --with-ftp instead.

PHP 4: Enable FTP support

`--with-ftp`

PHP 3: Include FTP support.

PHP 4: Option not available; use --enable-ftp instead

`--disable-url-fopen-wrapper`

PHP 3, PHP 4: Disable the URL-aware fopen wrapper that allows accessing files via http or ftp.

Attenzione

This switch is only available for PHP versions up to 4.0.3, newer versions provide an INI parameter called `allow_url_fopen` instead of forcing you to decide upon this feature at compile time.

`--with-mod-dav=DIR`

PHP 3, PHP 4: Include DAV support through Apache's mod_dav, DIR is mod_dav's installation directory (Apache module version only!)

`--with-openssl[=DIR]`

PHP 3, PHP 4: Include OpenSSL support in SNMP.

`--with-snmp[=DIR]`

PHP 3, PHP 4: Include SNMP support. DIR is the SNMP base install directory, defaults to searching through a number of common locations for the snmp install. Set DIR to shared to build as a dl, or shared,DIR to build as a dl and still specify DIR.

`--enable-ucd-snmp-hack`

PHP 3, PHP 4: Enable UCD SNMP hack

`--enable-sockets`

PHP 3: Option not available in PHP 3

PHP 4: Enable sockets support

`--with-yaz[=DIR]`

PHP 3: Option not available in PHP 3

PHP 4: Include YAZ support (ANSI/NISO Z39.50). DIR is the YAZ bin install directory

--enable-yp

PHP 3: Option not available; use --with-yp instead.

PHP 4: Include YP support

--with-yp

PHP 3: Include YP support

PHP 4: Option not available; use --enable-yp instead.

--with-mnogosearch

PHP 3, PHP 4: Include mnoGoSearch support.

PHP Behaviour

--enable-magic-quotes

PHP 3, PHP 4: Enable magic quotes by default.

--disable-short-tags

PHP 3, PHP 4: Disable the short-form <? start tag by default.

--enable-track-vars

PHP 3: Enable GET/POST/Cookie track variables by default.

PHP 4: Option not available in PHP 4; as of PHP 4.0.2, track_vars is always on.

Server

--with-aolserver-src=DIR

PHP 3: Option not available in PHP 3

PHP 4: Specify path to the source distribution of AOLserver

--with-aolserver=DIR

PHP 3: Option not available in PHP 3

PHP 4: Specify path to the installed AOLserver

--with-apache[=DIR]

PHP 3, PHP 4: Build Apache module. DIR is the top-level Apache build directory, defaults to /usr/local/etc/httpd.

--with-apxs[=FILE]

PHP 3, PHP 4: Build shared Apache module. FILE is the optional pathname to the Apache apxs tool; defaults to apxs.

--enable-versioning

PHP 3: Take advantage of versioning and scoping Provided by Solaris 2.x and Linux

PHP 4: Export only required symbols. See INSTALL for more information

--with-caudium[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Build PHP as a Pike module for use with the Caudium webserver. DIR is the Caudium base directory. If no directory is specified $prefix/caudium/server is used. The prefix is controlled by the --prefix option and is /usr/local per default.

--with-fhttpd[=DIR]

PHP 3, PHP 4: Build fhttpd module. DIR is the fhttpd sources directory, defaults to /usr/local/src/fhttpd.

--with-nsapi=DIR

PHP 3: Option not available in PHP 3

PHP 4: Specify path to the installed Netscape

--with-phttpd=DIR

PHP 3: Option not available in PHP 3

PHP 4:

--with-pi3web=DIR

PHP 3: Option not available in PHP 3

PHP 4: Build PHP as a module for use with Pi3Web.

--with-roxen=DIR

PHP 3: Option not available in PHP 3

PHP 4: Build PHP as a Pike module. DIR is the base Roxen directory, normally /usr/local/roxen/server.

--enable-roxen-zts

PHP 3: Option not available in PHP 3

PHP 4: Build the Roxen module using Zend Thread Safety.

--with-thttpd=SRCDIR

PHP 3: Option not available in PHP 3

PHP 4:

--with-zeus=DIR

PHP 3: Option not available in PHP 3

PHP 4: Build PHP as an ISAPI module for use with Zeus.

Text and language

--with-aspell[=DIR]

PHP 3, PHP 4: Include ASPELL support.

--with-gettext[=DIR]

PHP 3, PHP 4: Include GNU gettext support. DIR is the gettext install directory, defaults to /usr/local

--with-iconv[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Include iconv support.

--with-pspell[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Include PSPELL support.

--with-recode[=DIR]

PHP 3: Include GNU recode support.

PHP 4: Include recode support. DIR is the recode install directory.

--enable-shmop

PHP 3, PHP 4 : Activate shmop support.

XML

--with-dom[=DIR]

PHP 3: Option not available in PHP 3

PHP 4: Include DOM support (requires libxml >= 2.0). DIR is the libxml install directory, defaults to `/usr`

`--enable-sablot-errors-descriptive`

PHP 3: Option not available in PHP 3

PHP 4: Enable Descriptive errors

`--with-sablot[=DIR]`

PHP 3: Option not available in PHP 3

PHP 4: Include Sablotron support

`--enable-wddx`

PHP 3: Option not available in PHP 3

PHP 4: Enable WDDX support

`--disable-xml`

PHP 3: Option not available in PHP 3; XML functionality is not built in by default. Use --with-xml to turn it on.

PHP 4: Disable XML support using bundled expat lib

`--with-xml`

PHP 3: Include XML support

PHP 4: Option not available; XML support is built in by default. Use --disable-xml to turn it off.

Installation on Windows 9x/Me/NT/2000 systems

There are two main ways to install PHP for Windows: either manually or by using the InstallShield installer.

If you have Microsoft Visual Studio, you can also build PHP from the original source code.

Once you have PHP installed on your Windows system, you may also want to load various extensions for added functionality.

Windows InstallShield

The Windows PHP installer available from the downloads page at http://www.php.net/, this installs the CGI version of PHP and, for IIS, PWS, and Xitami, configures the web server as well.

Install your selected HTTP server on your system and make sure that it works.

Run the executable installer and follow the instructions provided by the installation wizard. Two types of installation are supported - standard, which provides sensible defaults for all the settings it can, and advanced, which asks questions as it goes along.

The installation wizard gathers enough information to set up the `php.ini` file and configure the web server to use PHP. For IIS and also PWS on NT Workstation, a list of all the nodes on the server with script map settings is displayed, and you can choose those nodes to which you wish to add the PHP script mappings.

Once the installation has completed the installer will inform you if you need to restart your system, restart the server, or just start using PHP.

General Installation Steps

This install guide will help you manually install and configure PHP on your Windows 9x/Me/NT/2000 webservers. The original version of this guide was compiled by Bob Silva (mailto:bob_silva@mail.umesd.k12.or.us), and can be found at http://www.umesd.k12.or.us/php/win32install.html.

This guide provides manual installation support for:

- Personal Web Server 3 and 4 or newer
- Internet Information Server 3 and 4 or newer
- Apache 1.3.x
- OmniHTTPd 2.0b1 and up
- Oreilly Website Pro
- Xitami

PHP 4 for Windows comes in two flavours - a CGI executable (php.exe), and several SAPI modules (for exapmle php4isapi.dll). The latter form is new to PHP 4, and provides significantly improved performance and some new functionality. However, please note that the SAPI modules are *NOT* yet considered to be production quality. The reason for this is that the PHP SAPI modules are using the thread-safe version of the PHP code, which is new to PHP 4, and has not yet been tested and pounded enough to be considered completely stable, and there are actually a few known bugs. On the other hand, some people have reported very good results with the SAPI modules, even though we’re not aware of anyone actually running it on a production site. In short - your mileage may vary; If you need absolute stability, trade the performance of the SAPI modules with the stability of the CGI executable.

If you choose one of the SAPI modules and use Windows 95, be sure to download the DCOM update from the Microsoft DCOM pages (http://download.microsoft.com/msdownload/dcom/95/x86/en/dcom95.exe). For the ISAPI module, an ISAPI 4.0 compliant Web server is required (tested on IIS 4.0, PWS 4.0 and IIS 5.0). IIS 3.0 is

NOT supported; You should download and install the Windows NT 4.0 Option Pack with IIS 4.0 if you want native PHP support.

The following steps should be performed on all installations before the server specific instructions.

- Extract the distribution file to a directory of your choice. "C:\PHP\" is a good start.
- The PHP binary, the SAPI modules, and some extensions rely on external DLLs for execution. Make sure that these DLLs in the distribution exist in a directory that is in the Windows PATH. The best bet to do it is to copy the files below into your system directory, which is typically:

  c:\windows\system for Windows 95/98
  c:\winnt\system32 for Windows NT/2000

The files to copy are:

  'php4ts.dll', if it already exists there, overwrite it
  The files in your distribution's 'dlls' directory. If you have them already installed on your system, overwrite them on

- Copy the file, 'php.ini-dist' to your '%WINDOWS%' directory on Windows 95/98 or to your '%SYSTEMROOT%' directory under Windows NT or Windows 2000 and rename it to 'php.ini'. Your '%WINDOWS%' or '%SYSTEMROOT%' directory is typically:

  c:\windows for Windows 95/98
  c:\winnt or c:\winnt40 for NT/2000 servers

- Edit your 'php.ini' file:
  - You will need to change the 'extension_dir' setting to point to your php-install-dir, or where you have placed your 'php_*.dll' files. ex: c:\php
  - If you are using OmniHTTPd, do not follow the next step. Set the 'doc_root' to point to your webservers document_root. ex: c:\apache\htdocs or c:\webroot
  - Choose which extensions you would like to load when PHP starts. You can uncomment the: 'extension=php_*.dll' lines in `php.ini` to load these extensions. You can also load a module dynamically in your script using dl(). See the section about Windows extensions.
  - On PWS and IIS, you can set the `browscap.ini` to point to: 'c:\windows\system\inetsrv\browscap.ini' on Windows 9x/Me and 'c:\winnt\system32\inetsrv\browscap.ini' on NT/2000 Server. Additional information on using the browscap functionality in PHP can be found at this mirror (http://php.netvision.net.il/browser-id.php3), select the "source" button to see it in action.

Building from source

Before getting started, it is worthwhile answering the question: "Why is building on Windows so hard?" Two reasons come to mind:

1. Windows does not (yet) enjoy a large community of developers who are willing to freely share their source. As a direct result, the necessary investment in infrastructure required to support such development hasn't been made. By and large, what is available has been made possible by the porting of necessary utilities from Unix. Don't be surprised if some of this heritage shows through from time to time.
2. Pretty much all of the instructions that follow are of the "set and forget" variety. So sit back and try follow the instructions below as faithfully as you can.

Preparations

Before you get started, you have a lot to download....

- For starters, get the Cygwin toolkit from the closest cygwin (http://sources.redhat.com/cygwin/download.html) mirror site. This will provide you most of the popular GNU utilities used by the build process.
- Download the rest of the build tools you will need from the PHP site at http://www.php.net/extra/win32build.zip.
- Get the source code for the DNS name resolver used by PHP at http://www.php.net/extra/bindlib_w32.zip. This is a replacement for the `resolv.lib` library included in `win32build.zip`.
- If you don't already have an unzip utility, you will need one. A free version is available from InfoZip (http://www.cdrom.com/pub/infozip/UnZip.html).

Finally, you are going to need the source to PHP 4 itself. You can get the latest development version using anonymous CVS (http://www.php.net/anoncvs.php). If you get a snapshot (http://snaps.php.net/) or a source (http://www.php.net/downloads.php) tarball, you not only will have to untar and ungzip it, but you will have to convert the bare linefeeds to crlf's in the `*.dsp` and `*.dsw` files before Microsoft Visual C++ will have anything to do with them.

> **Nota:** Place the `Zend` and `TSRM` directories inside the `php4` directory in order for the projects to be found during the build process.

Putting it all together

- Follow the instructions for installing the unzip utility of your choosing.
- Execute `setup.exe` and follow the installation instructions. If you choose to install to a path other than `c:\cygnus`, let the build process know by setting the Cygwin environment variable. On Windows 95/98 setting an environment variable can be done by placing a line in your autoexec.bat. On Windows NT, go to My Computer => Control Panel => System and select the environment tab.

> **Attenzione**
>
> Make a temporary directory for Cygwin to use, otherwise many commands (particularly bison) will fail. On Windows 95/98, **`mkdir C:\TMP`**. For Windows NT, **`mkdir %SystemDrive%\tmp`**.

- Make a directory and unzip `win32build.zip` into it.
- Launch Microsoft Visual C++, and from the menu select Tools => Options. In the dialog, select the directories tab. Sequentially change the dropdown to Executables, Includes, and Library files, and ensure that `cygwin\bin`, `win32build\include`, and `win32build\lib` are in each list, respectively. (To add an entry, select a blank line at the end of the list and begin typing). Typical entries will look like this:
  - `c:\cygnus\bin`
  - `c:\php-win32build\include`
  - `c:\php-win32build\lib`

  Press OK, and exit out of Visual C++.
- Make another directory and unzip `bindlib_w32.zip` into it. Decide whether you want to have debug symbols available (bindlib - Win32 Debug) or not (bindlib - Win32 Release). Build the appropriate configuration:
  - For GUI users, launch VC++, and then select File => Open Workspace and select bindlib. Then select Build=>Set Active Configuration and select the desired configuration. Finally select Build=>Rebuild All.
  - For command line users, make sure that you either have the C++ environment variables registered, or have run **vcvars.bat**, and then execute one of the following:
    - **`msdev bindlib.dsp /MAKE "bindlib - Win32 Debug"`**
    - **`msdev bindlib.dsp /MAKE "bindlib - Win32 Release"`**
  - At this point, you should have a usable `resolv.lib` in either your `Debug` or `Release` subdirectories. Copy this file into your `win32build\lib` directory over the file by the same name found in there.

Compiling

The best way to get started is to build the standalone/CGI version.

- For GUI users, launch VC++, and then select File => Open Workspace and select php4ts. Then select Build=>Set Active Configuration and select the desired configuration. Finally select Build=>Rebuild All.

- For command line users, make sure that you either have the C++ environment variables registered, or have run **vcvars.bat**, and then execute one of the following:
  - **`msdev php4ts.dsp /MAKE "php4ts - Win32 Debug_TS"`**
  - **`msdev php4ts.dsp /MAKE "php4ts - Win32 Release_TS"`**
  - At this point, you should have a usable `php.exe` in either your `Debug_TS` or `Release_TS` subdirectories.

Repeat the above steps with `php4isapi.dsp` (which can be found in `sapi\isapi`) in order to build the code necessary for integrating PHP with Microsoft IIS.

Installation of Windows extensions

After installing PHP and a webserver on Windows, you will probably want to install some extensions for added functionality. The following table describes some of the extensions available. As described in the manual installation steps, you can choose which extensions you would like to load when PHP starts by uncommenting the: 'extension=php_*.dll' lines in `php.ini`. You can also load a module dynamically in your script using dl().

The DLLs for PHP extensions are prefixed with 'php_' in PHP 4 (and 'php3_' in PHP 3). This prevents confusion between PHP extensions and their supporting libraries.

> **Nota:** In PHP 4.0.6 BCMath, Calendar, COM, FTP, MySQL, ODBC, PCRE, Session, WDDX and XML support is *built-in*. You don't need to load any additional extensions in order to use these functions. See your distributions `README.txt` or `install.txt` for a list of built in modules.

> **Nota:** Some of these extensions need extra dlls to work. Couple of them can be found in the distribution package, in the 'dlls' folder but some, e.g. Oracle (php_oci8.dll) require dlls which are not bundled with the distribution package.
>
> Copy the bundled dlls from 'dlls' folder to your Windows PATH, safe places are:
>
> c:\windows\system for Windows 95/98
> c:\winnt\system32 for Windows NT/2000
>
> If you have them already installed on your system, overwrite them only if something doesn't work correctly (Before overwriting them, it is a good idea to make a back-up).

Tabella 2-1. PHP Extensions

Extension	Description	Notes

Servers-Apache

This section contains notes and hints specific to Apache installs of PHP, both for Unix and Windows versions.

Details of installing PHP with Apache on Unix.

You can select arguments to add to the **configure** on line 8 below from the Complete list of configure options.

Esempio 2-5. Installation Instructions (Apache Module Version)

```
1.  gunzip apache_1.3.x.tar.gz
2.  tar xvf apache_1.3.x.tar
3.  gunzip php-x.x.x.tar.gz
4.  tar xvf php-x.x.x.tar
5.  cd apache_1.3.x
6.  ./configure --prefix=/www
7.  cd ../php-x.x.x
8.  ./configure --with-mysql --with-apache=../apache_1.3.x --enable-track-
vars
9.  make
10. make install
11. cd ../apache_1.3.x
12. for PHP 3: ./configure --activate-module=src/modules/php3/libphp3.a
    for PHP 4: ./configure --activate-module=src/modules/php4/libphp4.a
13. make
14. make install

  Instead of this step you may prefer to simply copy the httpd binary
  overtop of your existing binary.  Make sure you shut down your
  server first though.

15. cd ../php-x.x.x
16. for PHP 3: cp php3.ini-dist /usr/local/lib/php3.ini
    for PHP 4: cp php.ini-dist /usr/local/lib/php.ini

  You can edit your .ini file to set PHP options.  If
  you prefer this file in another location, use
  --with-config-file-path=/path in step 8.

17. Edit your httpd.conf or srm.conf file and add:

     For PHP 3:   AddType application/x-httpd-php3 .php3
     For PHP 4:   AddType application/x-httpd-php .php

  You can choose any extension you wish here.  .php is simply the one
  we suggest. You can even include .html .
```

```
18. Use your normal procedure for starting the Apache server. (You must
    stop and restart the server, not just cause the server to reload by
    use a HUP or USR1 signal.)
```

Depending on your Apache install and Unix variant, there are many possible ways to stop and restart the server. Below are some typical lines used in restarting the server, for different apache/unix installations. You should replace `/path/to/` with the path to these applications on your systems.

```
1. Several Linux and SysV variants:
/etc/rc.d/init.d/httpd restart

2. Using apachectl scripts:
/path/to/apachectl stop
/path/to/apachectl start

3. httpdctl and httpsdctl (Using OpenSSL), similar to apachectl:
/path/to/httpsdctl stop
/path/to/httpsdctl start

4. Using mod_ssl, or another SSL server, you may want to manually
stop and start:
/path/to/apachectl stop
/path/to/apachectl startssl
```

The locations of the apachectl and http(s)dctl binaries often vary. If your system has `locate` or `whereis` or `which` commands, these can assist you in finding your server contrl programs.

Different examples of compiling PHP for apache are as follows:

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

This will create a `libphp4.so` shared library that is loaded into Apache using a LoadModule line in Apache's `httpd.conf` file. The PostgreSQL support is embedded into this `libphp4.so` library.

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

This will again create a `libphp4.so` shared library for Apache, but it will also create a `pgsql.so` shared library that is loaded into PHP either by using the extension directive in `php.ini` file or by loading it explicitly in a script using the dl() function.

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

This will create a `libmodphp4.a` library, a `mod_php4.c` and some accompanying files and copy this into the `src/modules/php4` directory in the Apache source tree. Then you compile Apache using `--activate-module=src/modules/php4/libphp4.a` and the Apache build system will create `libphp4.a` and link it statically into the `httpd` binary. The PostgreSQL support is included directly into this `httpd` binary, so the final result here is a single `httpd` binary that includes all of Apache and all of PHP.

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

Same as before, except instead of including PostgreSQL support directly into the final `httpd` you will get a `pgsql.so` shared library that you can load into PHP from either the `php.ini` file or directly using dl().

When choosing to build PHP in different ways, you should consider the advantages and drawbacks of each method. Building as a shared object will mean that you can compile apache separately, and don't have to recompile everything as you add to, or change, PHP. Building PHP into apache (static method) means that PHP will load and run faster. For more information, see the Apache webpage on DSO support (http://www.apache.org/docs/dso.html).

Details of installing PHP on Windows with Apache 1.3.x

There are two ways to set up PHP to work with Apache 1.3.x on Windows. One is to use the CGI binary (php.exe), the other is to use the Apache module dll. In either case you need to stop the Apache server, and edit your `srm.conf` or `httpd.conf` to configure Apache to work with PHP.

Although there can be a few variations of configuring PHP under Apache, these are simple enough to be used by the newcomer. Please consult the Apache Docs for further configuration directives.

If you unziped the PHP package to C:\PHP\ as desribed in the General Installation Steps section, you need to insert these lines to your Apache conf file to set up the CGI binary:

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

Remerber to restart the server, for example, `NET STOP APACHE` followed by `NET START APACHE`.

If you would like to use PHP as a module in Apache, you should move `php4ts.dll` to the windows/system (for Windows 9x/Me) or winnt/system32 (for Windows NT/2000) directory, overwriting any older file. Then you should add the following two lines to you Apache conf file:

- `LoadModule php4_module c:/php/sapi/php4apache.dll`

- `AddType application/x-httpd-php .php .phtml`

To use the source code highlighting feature, simply create a PHP script file and stick this code in: `<?php show_source ("original_php_script.php"); ?>`. Substitute `original_php_script.php` with the name of the file you wish to show the source of. (This is the only way of doing so).

> **Nota:** On Win-Apache all backslashes in a path statement such as: "c:\directory\file.ext", must be converted to forward slashes.

Servers-CGI/Commandline

The default is to build PHP as a CGI program. This creates a commandline interpreter, which can be used for CGI processing, or for non-web-related PHP scripting. If you are running a web server PHP has module support for, you should generally go for that solution for performance reasons. However, the CGI version enables Apache users to run different PHP-enabled pages under different user-ids. Please make sure you read through the Security chapter if you are going to run PHP as a CGI.

Testing

If you have built PHP as a CGI program, you may test your build by typing **make test**. It is always a good idea to test your build. This way you may catch a problem with PHP on your platform early instead of having to struggle with it later.

Benchmarking

If you have built PHP 3 as a CGI program, you may benchmark your build by typing **make bench**. Note that if safe mode is on by default, the benchmark may not be able to finish if it takes longer then the 30 seconds allowed. This is because the set_time_limit() can not be used in safe mode. Use the max_execution_time configuration setting to control this time for your own scripts. **make bench** ignores the configuration file.

> **Nota: make bench** is only available for PHP 3.

Servers-fhttpd

To build PHP as an fhttpd module, answer "yes" to "Build as an fhttpd module?" (the `--with-fhttpd=`*`DIR`* option to configure) and specify the fhttpd source base directory. The default

directory is `/usr/local/src/fhttpd`. If you are running fhttpd, building PHP as a module will give better performance, more control and remote execution capability.

Servers-Caudium

PHP 4 can be build as a Pike module for the Caudium webserver. Note that this is not supported with PHP 3. Follow the simple instructions below to install PHP 4 for Caudium.

Esempio 2-6. Caudium Installation Instructions

```
1.  Make sure you have Caudium installed prior to attempting to
    install PHP 4. For PHP 4 to work correctly, you will need Pike
    7.0.268 or newer. For the sake of this example we assume that
    Caudium is installed in /opt/caudium/server/.
2.  Change directory to php-x.y.z (where x.y.z is the version number).
3.  ./configure --with-caudium=/opt/caudium/server
4.  make
5.  make install
6.  Restart Caudium if it's currently running.
7.  Log into the graphical configuration interface and go to the
    virtual server where you want to add PHP 4 support.
8.  Click Add Module and locate and then add the PHP 4 Script Sup-
port module.
9.  If the documentation says that the 'PHP 4 interpreter isn't
    available', make sure that you restarted the server. If you did
    check /opt/caudium/logs/debug/default.1 for any errors related to
    PHP4.so. Also make sure that
    caudium/server/lib/[pike-version]/PHP4.so
    is present.
10. Configure the PHP Script Support module if needed.
```

You can of course compile your Caudium module with support for the various extensions available in PHP 4. See the complete list of configure options for an exhaustive rundown.

> **Nota:** When compiling PHP 4 with MySQL support you must make sure that the normal MySQL client code is used. Otherwise there might be conflicts if your Pike already has MySQL support. You do this by specifying a MySQL install directory the --with-mysql option.

Servers-IIS/PWS

This section contains notes and hints specific to IIS (Microsoft Internet Information Server) installing PHP for PWS/IIS 3, PWS 4 or newer and IIS 4 or newer versions.

Windows and PWS/IIS 3

The recommended method for configuring these servers is to use the INF file included with the distribution (php_iis_reg.inf). You may want to edit this file and make sure the extensions and PHP install directories match your configuration. Or you can follow the steps below to do it manually.

> **Attenzione**
>
> These steps involve working directly with the Windows registry. One error here can leave your system in an unstable state. We highly recommend that you back up your registry first. The PHP Development team will not be held responsible if you damage your registry.

- Run Regedit.
- Navigate to: `HKEY_LOCAL_MACHINE /System /CurrentControlSet /Services /W3Svc /Parameters /ScriptMap`.
- On the edit menu select: `New->String Value`.
- Type in the extension you wish to use for your php scripts. ex: `.php`
- Double click on the new string value and enter the path to `php.exe` in the value data field. ex: `c:\php\php.exe %s %s`. The '%s %s' is VERY important, PHP will not work properly without it.
- Repeat these steps for each extension you wish to associate with PHP scripts.
- Now navigate to: `HKEY_CLASSES_ROOT`
- On the edit menu select: `New->Key`.
- Name the key to the extension you setup in the previous section. ex: `.php`
- Highlight the new key and in the right side pane, double click the "default value" and enter `phpfile`.
- Repeat the last step for each extension you set up in the previous section.
- Now create another `New->Key` under `HKEY_CLASSES_ROOT` and name it `phpfile`.
- Highlight the new key `phpfile` and in the right side pane, double click the "default value" and enter `PHP Script`.
- Right click on the `phpfile` key and select `New->Key`, name it `Shell`.
- Right click on the `Shell` key and select `New->Key`, name it `open`.
- Right click on the `open` key and select `New->Key`, name it `command`.
- Highlight the new key `command` and in the right side pane, double click the "default value" and enter the path to `php.exe`. ex: `c:\php\php.exe -q %1`. (don't forget the `%1`).

- Exit Regedit.
- If using PWS on Windows, reboot to reload the registry.

PWS and IIS 3 users now have a fully operational system. IIS 3 users can use a nifty tool (http://www.genusa.com/iis/iiscfg.html) from Steven Genusa to configure their script maps.

Windows and PWS 4 or newer

When installing PHP on Windows with PWS 4 or newer version, you have two options. One to set up the PHP CGI binary, the other is to use the ISAPI module dll.

If you choose the CGI binary, do the following:

- Edit the enclosed `pws-php4cgi.reg` file (look into the sapi dir) to reflect the location of your php.exe. Forward slashes should be escaped, for example: `[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\w3svc\parameters\Script Map] ".php"="C:\\PHP\\php.exe"`
- In the PWS Manager, right click on a given directory you want to add PHP support to, and select Properties. Check the 'Execute' checkbox, and confirm.

If you choose the ISAPI module, do the following:

- Edit the enclosed `pws-php4isapi.reg` file (look into the sapi dir) to reflect the location of your php4isapi.dll. Forward slashes should be escaped, for example: `[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\w3svc\parameters\Script Map] ".php"="C:\\PHP\\sapi\\php4isapi.dll"`
- In the PWS Manager, right click on a given directory you want to add PHP support to, and select Properties. Check the 'Execute' checkbox, and confirm.

Windows NT/2000 and IIS 4 or newer

To install PHP on an NT/2000 Server running IIS 4 or newer, follow these instructions. You have two options to set up PHP, using the CGI binary (php.exe) or with the ISAPI module.

In either case, you need to start the Microsoft Management Console (may appear as 'Internet Services Manager', either in your Windows NT 4.0 Option Pack branch or the Control Panel=>Administrative Tools under Windows 2000). Then right click on your Web server node (this will most probably appear as 'Default Web Server'), and select 'Properties'.

If you want to use the CGI binary, do the following:

- Under 'Home Directory', 'Virtual Directory', or 'Directory', click on the 'Configuration' button, and then enter the App Mappings tab.

- Click Add, and in the Executable box, type: `c:\php\php.exe %s %s` (assuming that you have unziped PHP in c:\php\). You MUST have the %s %s on the end, PHP will not function properly if you fail to do this.
- In the Extension box, type the file name extension you want associated with PHP scripts. Leave 'Method exclusions' blank, and check the Script engine checkbox. You must repeat step 3 and 4 for each extension you want associated with PHP scripts. (`.php` and `.phtml` are common.)
- Set up the appropriate security. (This is done in Internet Service Manager), and if your NT Server uses NTFS file system, add execute rights for I_USR_ to the directory that contains `php.exe`.

To use the ISAPI module, do the following:

- If you don't want to perform HTTP Authentication using PHP, you can (and should) skip this step. Under ISAPI Filters, add a new ISAPI filter. Use PHP as the filter name, and supply a path to the php4isapi.dll.
- Under 'Home Directory', click on the 'Configuration' button. Add a new entry to the Application Mappings. Use the path to the php4isapi.dll as the Executable, supply .php as the extension, leave Method exclusions blank, and check the Script engine checkbox.
- Stop IIS completely
- Start IIS again

Servers-Netscape and iPlanet

To build PHP with NES or iPlanet web servers, enter the proper install directory for the `--with-nsapi = DIR` option. The default directory is usually `/opt/netscape/suitespot/`. Please also read `/php-xxx-version/sapi/nsapi/nsapi-readme.txt`.

Esempio 2-7. Installation Example for Netscape Enterprise on Solaris

```
Instructions for Sun Solaris 2.6 with Netscape Enterprise Server 3.6
From: bhager@invacare.com

1. Install the following packages from www.sunfreeware.com or another
download site:

    flex-2_5_4a-sol26-sparc-local
    gcc-2_95_2-sol26-sparc-local
    gzip-1.2.4-sol26-sparc-local
    perl-5_005_03-sol26-sparc-local
    bison-1_25-sol26-sparc-local
    make-3_76_1-sol26-sparc-local
    m4-1_4-sol26-sparc-local
```

```
    autoconf-2.13
    automake-1.4
    mysql-3.23.24-beta (if you want mysql support)
    tar-1.13 (GNU tar)

2. Make sure your path includes the proper directories
    PATH=.:/usr/local/bin:/usr/sbin:/usr/bin:/usr/ccs/bin
    export PATH

3. gunzip php-x.x.x.tar.gz (if you have a .gz dist, otherwise go to 4)
4. tar xvf php-x.x.x.tar
5. cd ../php-x.x.x

6. For the following step, make sure /opt/netscape/suitespot/ is where
   your netscape server is installed. Otherwise, change to correct path:
    /configure --with-mysql=/usr/local/mysql --with-
nsapi=/opt/netscape/suitespot/ --enable-track-vars --enable-libgcc
7. make
8. make install
```

After performing the base install and reading the appropriate readme file, you may need to performs some additional configuration steps.

Firstly you may need to add some paths to the LD_LIBRARY_PATH environment for Netscape to find all the shared libs. This can best done in the start script for your Netscape server. Windows users can probably skip this step. The start script is often located in: `/path/to/server/https-servername/start`

You may also need to edit the configuration files that are located in:`/path/to/server/https-servername/config/`.

Esempio 2-8. Configuration Example for Netscape Enterprise

```
Configuration Instructions for Netscape Enterprise Server
From: bhager@invacare.com

1. Add the following line to mime.types:
    type=magnus-internal/x-httpd-php exts=php

2. Add the following to obj.conf, shlib will vary depending on
    your OS, for unix it will be something like
    /opt/netscape/suitespot/bin/libphp4.so.

    You should place the following lines after mime types init.
    Init fn="load-
modules" funcs="php4_init,php4_close,php4_execute,php4_auth_trans" shlib="/php4/nsapiPHP4
    Init fn=php4_init errorString="Failed to initialize PHP!"
```

```
    <object name="default">
    .
    .
    .
    .#NOTE this next line should happen after all 'ObjectType' and be-
fore all 'AddLog' lines
    Service fn="php4_execute" type="magnus-internal/x-httpd-php"
    .
    .
    </Object>


    <Object name="x-httpd-php">
    ObjectType fn="force-type" type="magnus-internal/x-httpd-php"
    Service fn=php4_execute
    </Object>


    Authentication configuration

    PHP authentication cannot be used with any other authentica-
tion. ALL AUTHENTICATION IS
    PASSED TO YOUR PHP SCRIPT. To configure PHP Authentication for the en-
tire server, add
    the following line:

    <Object name="default">
    AuthTrans fn=php4_auth_trans
    .
    .
    .
    .
    </Object>

    To use PHP Authentication on a single directory, add the following:

    <Object ppath="d:\path\to\authenticated\dir\*">
    AuthTrans fn=php4_auth_trans
    </Object>
```

If you are running Netscape Enterprise 4.x, then you should use the following:

Esempio 2-9. Configuration Example for Netscape Enterprise 4.x

```
Place these lines after the mime types init, and every-
thing else is similar
to the example configuration above.
From: Graeme Hoose (GraemeHoose@BrightStation.com)
```

```
Init fn="load-
modules" shlib="/path/to/server4/bin/libphp4.so" funcs="php4_init,php4_close,php4_execute
Init fn="php4_init" LateInit="yes"
```

Servers-OmniHTTPd Server

This section contains notes and hints specific to OmniHTTPd.

OmniHTTPd 2.0b1 and up for Windows

This has got to be the easiest config there is:

- Step 1: Install OmniHTTPd server.
- Step 2: Right click on the blue OmniHTTPd icon in the system tray and select `Properties`
- Step 3: Click on `Web Server Global Settings`
- Step 4: On the 'External' tab, enter: `virtual = .php | actual = c:\path-to-php-dir\php.exe`, and use the Add button.
- Step 5: On the `Mime` tab, enter: `virtual = wwwserver/stdcgi | actual = .php`, and use the Add button.
- Step 6: Click `OK`

Repeat steps 2 - 6 for each extension you want to associate with PHP.

> **Nota:** Some OmniHTTPd packages come with built in PHP support. You can choose at setup time to do a custom setup, and uncheck the PHP component. We recommend you to use the latest PHP binaries. Some OmniHTTPd servers come with PHP 4 beta distributions, so you should choose not to set up the built in support, but install your own. If the server is already on your machine, use the Replace button in Step 4 and 5 to set the new, correct information.

Servers-Oreilly Website Pro

This section contains notes and hints specific to Oreilly Website Pro.

Oreilly Website Pro 2.5 and up for Windows

This list describes how to set up the PHP CGI binary or the ISAPI module to work with Oreilly Website Pro on Windows.

- Edit the Server Properties and select the tab "Mapping".
- From the List select "Associations" and enter the desired extension (".php") and the path to the CGI exe (ex. c:\php\php.exe) or the ISAPI dll file (ex. c:\php\sapi\php4isapi.dll).
- Select "Content Types" add the same extension ".php" and enter the content type. If you choose the CGI exe file, enter 'wwwserver/shellcgi', if you chosse the ISAPI module, enter 'wwwserver/isapi' (both without quotes).

Servers-Xitami

This section contains notes and hints specific to Xitami.

Xitami for Windows

This list describes how to set up the PHP CGI binary to work with Xitami on Windows.

- Make sure the webserver is running, and point your browser to xitamis admin console (usually http://127.0.0.1/admin), and click on Configuration.
- Navigate to the Filters, and put the extension which php should parse (i.e. .php) into the field File extensions (.xxx).
- In Filter command or script put the path and name of your php executable i.e. c:\php\php.exe.
- Press the 'Save' icon.

Servers-Other web servers

PHP can be built to support a large number of web servers. Please see Server-related options for a full list of server-related configure options. The PHP CGI binaries are compatible with almost all webservers supporting the CGI interface.

Problems?

Read the FAQ

Some problems are more common than others. The most common ones are listed in the PHP FAQ, found at http://www.php.net/FAQ.php

Other problems

If you are still stuck, someone on the PHP installation mailing list may be able to help you. You should check out the archive first, in case someone already answered someone else who had the same problem as you. The archives are available from the support page on http://www.php.net/. To subscribe to the PHP installation mailing list, send an empty mail to php-install-subscribe@lists.php.net (mailto:php-install-subscribe@lists.php.net). The mailing list address is `php-install@lists.php.net`.

If you want to get help on the mailing list, please try to be precise and give the necessary details about your environment (which operating system, what PHP version, what web server, if you are running PHP as CGI or a server module, etc.), and preferably enough code to make others able to reproduce and test your problem.

Bug reports

If you think you have found a bug in PHP, please report it. The PHP developers probably don't know about it, and unless you report it, chances are it won't be fixed. You can report bugs using the bug-tracking system at http://bugs.php.net/.

Read the Bugs-Dos-And-Donts (http://bugs.php.net/bugs-dos-and-donts.php) before submitting any bug reports!

Capitolo 3. Configuration

The configuration file

The configuration file (called `php3.ini` in PHP 3.0, and simply `php.ini` as of PHP 4.0) is read when PHP starts up. For the server module versions of PHP, this happens only once when the web server is started. For the CGI version, it happens on every invocation.

When using PHP as an Apache module, you can also change the configuration settings using directives in Apache configuration files and .htaccess files.

With PHP 3.0, there are Apache directives that correspond to each configuration setting in the `php3.ini` name, except the name is prefixed by "php3_".

With PHP 4.0, there are several Apache directives that allow you to change the PHP configuration from within the Apache configuration file itself.

php_value *`name value`*

This sets the value of the specified variable.

php_flag *`name on|off`*

This is used to set a Boolean configuration option.

php_admin_value *`name value`*

This sets the value of the specified variable. "Admin" configuration settings can only be set from within the main Apache configuration files, and not from .htaccess files.

php_admin_flag *`name on|off`*

This is used to set a Boolean configuration option.

You can view the settings of the configuration values in the output of phpinfo(). You can also access the values of individial configuration settings using get_cfg_var().

General Configuration Directives

`allow_url_fopen` boolean

This option enables the URL-aware fopen wrappers that enable accessing URL object like files. Default wrappers are provided for the access of remote files using the ftp or http protocol, some extensions like zlib may register additional wrappers.

Nota: This option was introduced immediately after the release of version 4.0.3. For versions up to and including 4.0.3 you can only disable this feature at compile time by using the configuration switch `--disable-url-fopen-wrapper`.

`asp_tags` boolean

Enables the use of ASP-like <% %> tags in addition to the usual <?php ?> tags. This includes the variable-value printing shorthand of <%= $value %>. For more information, see Escaping from HTML.

Nota: Support for ASP-style tags was added in 3.0.4.

`auto_append_file` string

Specifies the name of a file that is automatically parsed after the main file. The file is included as if it was called with the include() function, so include_path is used.

The special value none disables auto-appending.

Nota: If the script is terminated with exit(), auto-append will *not* occur.

`auto_prepend_file` string

Specifies the name of a file that is automatically parsed before the main file. The file is included as if it was called with the include() function, so include_path is used.

The special value none disables auto-prepending.

`cgi_ext` string

`display_errors` boolean

This determines whether errors should be printed to the screen as part of the HTML output or not.

`doc_root` string

PHP’s "root directory" on the server. Only used if non-empty. If PHP is configured with safe mode, no files outside this directory are served.

`engine` boolean

This directive is really only useful in the Apache module version of PHP. It is used by sites that would like to turn PHP parsing on and off on a per-directory or per-virtual server basis. By

putting **`engine off`** in the appropriate places in the `httpd.conf` file, PHP can be enabled or disabled.

`error_log` string

Name of file where script errors should be logged. If the special value `syslog` is used, the errors are sent to the system logger instead. On UNIX, this means syslog(3) and on Windows NT it means the event log. The system logger is not supported on Windows 95.

`error_reporting` integer

Set the error reporting level. The parameter is an integer representing a bit field. Add the values of the error reporting levels you want.

Tabella 3-1. Error Reporting Levels

bit value	enabled reporting

The default value for this directive is 7 (normal errors, normal warnings and parser errors are shown).

`html_errors` boolean

Turn off HTML tags in error messages.

`open_basedir` string

Limit the files that can be opened by PHP to the specified directory-tree.

When a script tries to open a file with, for example, fopen or gzopen, the location of the file is checked. When the file is outside the specified directory-tree, PHP will refuse to open it. All symbolic links are resolved, so it’s not possible to avoid this restriction with a symlink.

The special value . indicates that the directory in which the script is stored will be used as base-directory.

Under Windows, separate the directories with a semicolon. On all other systems, separate the directories with a colon. As an Apache module, open_basedir paths from parent directories are now automatically inherited.

Nota: Support for multiple directories was added in 3.0.7.

The default is to allow all files to be opened.

`gpc_order` string

Set the order of GET/POST/COOKIE variable parsing. The default setting of this directive is "GPC". Setting this to "GP", for example, will cause PHP to completely ignore cookies and to overwrite any GET method variables with POST-method variables of the same name.

`ignore_user_abort` string

On by default. If changed to Off scripts will be terminated as soon as they try to output something after a client has aborted their connection. ignore_user_abort().

include_path string

Specifies a list of directories where the require(), include() and **fopen_with_path()** functions look for files. The format is like the system's PATH environment variable: a list of directories separated with a colon in UNIX or semicolon in Windows.

Esempio 3-1. UNIX include_path

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

Esempio 3-2. Windows include_path

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

The default value for this directive is `.` (only the current directory).

isapi_ext string

log_errors boolean

Tells whether script error messages should be logged to the server's error log. This option is thus server-specific.

magic_quotes_gpc boolean

Sets the magic_quotes state for GPC (Get/Post/Cookie) operations. When magic_quotes are on, all ' (single-quote), " (double quote), \ (backslash) and NUL's are escaped with a backslash automatically. If magic_quotes_sybase is also on, a single-quote is escaped with a single-quote instead of a backslash.

magic_quotes_runtime boolean

If *magic_quotes_runtime* is enabled, most functions that return data from any sort of external source including databases and text files will have quotes escaped with a backslash. If *magic_quotes_sybase* is also on, a single-quote is escaped with a single-quote instead of a backslash.

magic_quotes_sybase boolean

If *magic_quotes_sybase* is also on, a single-quote is escaped with a single-quote instead of a backslash if *magic_quotes_gpc* or *magic_quotes_runtime* is enabled.

max_execution_time integer

This sets the maximum time in seconds a script is allowed to run before it is terminated by the parser. This helps prevent poorly written scripts from tying up the server. The default setting is `30`.

The maximum execution time is not affected by system calls, the sleep() function, etc. Please see the set_time_limit() function for more details.

`memory_limit` integer

This sets the maximum amount of memory in bytes that a script is allowed to allocate. This helps prevent poorly written scripts for eating up all available memory on a server.

`nsapi_ext` string

`register_globals` boolean

Tells whether or not to register the EGPCS (Environment, GET, POST, Cookie, Server) variables as global variables. You may want to turn this off if you don't want to clutter your scripts' global scope with user data. This makes the most sense when coupled with track_vars - in which case you can access all of the EGPCS variables through the `$HTTP_ENV_VARS`, `$HTTP_GET_VARS`, `$HTTP_POST_VARS`, `$HTTP_COOKIE_VARS`, and `$HTTP_SERVER_VARS` arrays in the global scope.

`short_open_tag` boolean

Tells whether the short form (**`<? ?>`**) of PHP's open tag should be allowed. If you want to use PHP in combination with XML, you have to disable this option. If disabled, you must use the long form of the open tag (**`<?php ?>`**).

`sql.safe_mode` boolean

`track_errors` boolean

If enabled, the last error message will always be present in the global variable $php_errormsg.

`track_vars` boolean

If enabled, then Environment, GET, POST, Cookie, and Server variables can be found in the global associative arrays `$HTTP_ENV_VARS`, `$HTTP_GET_VARS`, `$HTTP_POST_VARS`, `$HTTP_COOKIE_VARS`, and `$HTTP_SERVER_VARS`.

Note that as of PHP 4.0.3, track_vars is always turned on.

`upload_tmp_dir` string

The temporary directory used for storing files when doing file upload. Must be writable by whatever user PHP is running as.

upload_max_filesize integer

The maximum size of an uploaded file. The value is in bytes.

user_dir string

The base name of the directory used on a user's home directory for PHP files, for example `public_html`.

warn_plus_overloading boolean

If enabled, this option makes PHP output a warning when the plus (+) operator is used on strings. This is to make it easier to find scripts that need to be rewritten to using the string concatenator instead (`.`).

Mail Configuration Directives

SMTP string

DNS name or IP address of the SMTP server PHP under Windows should use for mail sent with the mail() function.

sendmail_from string

Which "From:" mail address should be used in mail sent from PHP under Windows.

sendmail_path string

Where the **sendmail** program can be found, usually `/usr/sbin/sendmail` or `/usr/lib/sendmail` **configure** does an honest attempt of locating this one for you and set a default, but if it fails, you can set it here.

Systems not using sendmail should set this directive to the sendmail wrapper/replacement their mail system offers, if any. For example, Qmail (http://www.qmail.org/) users can normally set it to `/var/qmail/bin/sendmail`.

Safe Mode Configuration Directives

safe_mode boolean

Whether to enable PHP's safe mode. Read the Security and Safe Mode chapters for more information.

safe_mode_exec_dir string

If PHP is used in safe mode, system() and the other functions executing system programs refuse to start programs that are not in this directory.

Debugger Configuration Directives

debugger.host string

DNS name or IP address of host used by the debugger.

debugger.port string

Port number used by the debugger.

debugger.enabled boolean

Whether the debugger is enabled.

Extension Loading Directives

enable_dl boolean

This directive is really only useful in the Apache module version of PHP. You can turn dynamic loading of PHP extensions with dl() on and off per virtual server or per directory.

The main reason for turning dynamic loading off is security. With dynamic loading, it's possible to ignore all the safe_mode and open_basedir restrictions.

The default is to allow dynamic loading, except when using safe-mode. In safe-mode, it's always imposible to use dl().

extension_dir string

In what directory PHP should look for dynamically loadable extensions.

extension string

Which dynamically loadable extensions to load when PHP starts up.

MySQL Configuration Directives

mysql.allow_persistent boolean

Whether to allow persistent MySQL connections.

mysql.default_host string

The default server host to use when connecting to the database server if no other host is specified.

mysql.default_user string

The default user name to use when connecting to the database server if no other name is specified.

mysql.default_password string

The default password to use when connecting to the database server if no other password is specified.

mysql.max_persistent integer

The maximum number of persistent MySQL connections per process.

mysql.max_links integer

The maximum number of MySQL connections per process, including persistent connections.

mSQL Configuration Directives

msql.allow_persistent boolean

Whether to allow persistent mSQL connections.

msql.max_persistent integer

The maximum number of persistent mSQL connections per process.

msql.max_links integer

The maximum number of mSQL connections per process, including persistent connections.

Postgres Configuration Directives

pgsql.allow_persistent boolean

Whether to allow persistent Postgres connections.

pgsql.max_persistent integer

The maximum number of persistent Postgres connections per process.

pgsql.max_links integer

The maximum number of Postgres connections per process, including persistent connections.

SESAM Configuration Directives

sesam_oml string

Name of BS2000 PLAM library containing the loadable SESAM driver modules. Required for using SESAM functions. The BS2000 PLAM library must be set ACCESS=READ,SHARE=YES because it must be readable by the apache server's user id.

sesam_configfile string

Name of SESAM application configuration file. Required for using SESAM functions. The BS2000 file must be readable by the apache server's user id.

The application configuration file will usually contain a configuration like (see SESAM reference manual):

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

sesam_messagecatalog string

Name of SESAM message catalog file. In most cases, this directive is not neccessary. Only if the SESAM message file is not installed in the system's BS2000 message file table, it can be set with this directive.

The message catalog must be set ACCESS=READ,SHARE=YES because it must be readable by the apache server's user id.

Sybase Configuration Directives

sybase.allow_persistent boolean

Whether to allow persistent Sybase connections.

sybase.max_persistent integer

The maximum number of persistent Sybase connections per process.

sybase.max_links integer

The maximum number of Sybase connections per process, including persistent connections.

Sybase-CT Configuration Directives

sybct.allow_persistent boolean

Whether to allow persistent Sybase-CT connections. The default is on.

sybct.max_persistent integer

The maximum number of persistent Sybase-CT connections per process. The default is -1 meaning unlimited.

sybct.max_links integer

The maximum number of Sybase-CT connections per process, including persistent connections. The default is -1 meaning unlimited.

sybct.min_server_severity integer

Server messages with severity greater than or equal to sybct.min_server_severity will be reported as warnings. This value can also be set from a script by calling sybase_min_server_severity(). The default is 10 which reports errors of information severity or greater.

sybct.min_client_severity integer

Client library messages with severity greater than or equal to sybct.min_client_severity will be reported as warnings. This value can also be set from a script by calling sybase_min_client_severity(). The default is 10 which effectively disables reporting.

sybct.login_timeout integer

The maximum time in seconds to wait for a connection attempt to succeed before returning failure. Note that if max_execution_time has been exceeded when a connection attempt times out, your script will be terminated before it can take action on failure. The default is one minute.

sybct.timeout integer

The maximum time in seconds to wait for a select_db or query operation to succeed before returning failure. Note that if max_execution_time has been exceeded when am operation times out, your script will be terminated before it can take action on failure. The default is no limit.

sybct.hostname string

The name of the host you claim to be connecting from, for display by sp_who. The default is none.

Informix Configuration Directives

ifx.allow_persistent boolean

Whether to allow persistent Informix connections.

ifx.max_persistent integer

The maximum number of persistent Informix connections per process.

ifx.max_links integer

The maximum number of Informix connections per process, including persistent connections.

ifx.default_host string

The default host to connect to when no host is specified in ifx_connect() or ifx_pconnect().

ifx.default_user string

The default user id to use when none is specified in ifx_connect() or ifx_pconnect().

ifx.default_password string

The default password to use when none is specified in ifx_connect() or ifx_pconnect().

ifx.blobinfile boolean

Set to TRUE if you want to return blob columns in a file, FALSE if you want them in memory. You can override the setting at runtime with ifx_blobinfile_mode().

ifx.textasvarchar boolean

Set to TRUE if you want to return TEXT columns as normal strings in select statements, FALSE if you want to use blob id parameters. You can override the setting at runtime with ifx_textasvarchar().

ifx.byteasvarchar boolean

Set to TRUE if you want to return BYTE columns as normal strings in select queries, FALSE if you want to use blob id parameters. You can override the setting at runtime with ifx_textasvarchar().

ifx.charasvarchar boolean

Set to TRUE if you want to trim trailing spaces from CHAR columns when fetching them.

ifx.nullformat boolean

Set to TRUE if you want to return NULL columns as the literal string "NULL", FALSE if you want them returned as the empty string "". You can override this setting at runtime with ifx_nullformat().

BC Math Configuration Directives

bcmath.scale integer

Number of decimal digits for all bcmath functions.

Browser Capability Configuration Directives

browscap string

Name of browser capabilities file. See also get_browser().

Unified ODBC Configuration Directives

uodbc.default_db string

ODBC data source to use if none is specified in odbc_connect() or odbc_pconnect().

uodbc.default_user string

User name to use if none is specified in odbc_connect() or odbc_pconnect().

uodbc.default_pw string

Password to use if none is specified in odbc_connect() or odbc_pconnect().

uodbc.allow_persistent boolean

Whether to allow persistent ODBC connections.

uodbc.max_persistent integer

The maximum number of persistent ODBC connections per process.

uodbc.max_links integer

The maximum number of ODBC connections per process, including persistent connections.

Multi-Byte String Configuration Directives

mbstring.internal_encoding string

`mbstring.internal_encoding` defines default internal character encoding.

mbstring.http_input string

`mbstring.http_input` defines default HTTP input character encoding.

mbstring.http_output string

`mbstring.http_output` defines default HTTP output character encoding.

mbstring.detect_order string

`mbstring.detect_order` defines default character encoding detection order.

`mbstring.substitute_character` string

`mbstring.substitute_character` defines character to substitute for invalid character codes.

Capitolo 4. Security

PHP is a powerful language and the interpreter, whether included in a web server as a module or executed as a separate CGI binary, is able to access files, execute commands and open network connections on the server. These properties make anything run on a web server insecure by default. PHP is designed specifically to be a more secure language for writing CGI programs than Perl or C, and with correct selection of compile-time and runtime configuration options, and proper coding practices, it can give you exactly the combination of freedom and security you need.

As there are many different ways of utilizing PHP, there are many configuration options controlling its behaviour. A large selection of options guarantees you can use PHP for a lot of purposes, but it also means there are combinations of these options and server configurations that result in an insecure setup.

The configuration flexibility of PHP is equally rivalled by the code flexibility. PHP can be used to build complete server applications, with all the power of a shell user, or it can be used for simple server-side includes with little risk in a tightly controlled environment. How you build that environment, and how secure it is, is largely up to the PHP developer.

This chapter starts by explaining the different configuration option combinations and the situations they can be safely used. It then describes different considerations in coding for different levels of security, and ends with some general security advice.

Installed as CGI binary

Possible attacks

Using PHP as a CGI binary is an option for setups that for some reason do not wish to integrate PHP as a module into server software (like Apache), or will use PHP with different kinds of CGI wrappers to create safe chroot and setuid environments for scripts. This setup usually involves installing executable PHP binary to the web server cgi-bin directory. CERT advisory CA-96.11 (http://www.cert.org/advisories/CA-96.11.interpreters_in_cgi_bin_dir.html) recommends against placing any interpreters into cgi-bin. Even if the PHP binary can be used as a standalone interpreter, PHP is designed to prevent the attacks this setup makes possible:

- Accessing system files: `http://my.host/cgi-bin/php?/etc/passwd`

  The query information in a url after the question mark (?) is passed as command line arguments to the interpreter by the CGI interface. Usually interpreters open and execute the file specified as the first argument on the command line.

  When invoked as a CGI binary, PHP refuses to interpret the command line arguments.

- Accessing any web document on server: `http://my.host/cgi-bin/php/secret/doc.html`

The path information part of the url after the PHP binary name, `/secret/doc.html` is conventionally used to specify the name of the file to be opened and interpreted by the CGI program. Usually some web server configuration directives (Apache: Action) are used to redirect requests to documents like `http://my.host/secret/script.php` to the PHP interpreter. With this setup, the web server first checks the access permissions to the directory `/secret`, and after that creates the redirected request `http://my.host/cgi-bin/php/secret/script.php`. Unfortunately, if the request is originally given in this form, no access checks are made by web server for file `/secret/script.php`, but only for the `/cgi-bin/php` file. This way any user able to access `/cgi-bin/php` is able to access any protected document on the web server.

In PHP, compile-time configuration option --enable-force-cgi-redirect and runtime configuration directives doc_root and user_dir can be used to prevent this attack, if the server document tree has any directories with access restrictions. See below for full the explanation of the different combinations.

Case 1: only public files served

If your server does not have any content that is not restricted by password or ip based access control, there is no need for these configuration options. If your web server does not allow you to do redirects, or the server does not have a way to communicate to the PHP binary that the request is a safely redirected request, you can specify the option --enable-force-cgi-redirect to the configure script. You still have to make sure your PHP scripts do not rely on one or another way of calling the script, neither by directly `http://my.host/cgi-bin/php/dir/script.php` nor by redirection `http://my.host/dir/script.php`.

Redirection can be configured in Apache by using AddHandler and Action directives (see below).

Case 2: using --enable-force-cgi-redirect

This compile-time option prevents anyone from calling PHP directly with a url like `http://my.host/cgi-bin/php/secretdir/script.php`. Instead, PHP will only parse in this mode if it has gone through a web server redirect rule.

Usually the redirection in the Apache configuration is done with the following directives:

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

This option has only been tested with the Apache web server, and relies on Apache to set the non-standard CGI environment variable REDIRECT_STATUS on redirected requests. If your web server does not support any way of telling if the request is direct or redirected, you cannot use this option and you must use one of the other ways of running the CGI version documented here.

Case 3: setting doc_root or user_dir

To include active content, like scripts and executables, in the web server document directories is sometimes consider an insecure practice. If, because of some configuration mistake, the scripts are not executed but displayed as regular HTML documents, this may result in leakage of intellectual property or security information like passwords. Therefore many sysadmins will prefer setting up another directory structure for scripts that are accessible only through the PHP CGI, and therefore always interpreted and not displayed as such.

Also if the method for making sure the requests are not redirected, as described in the previous section, is not available, it is necessary to set up a script doc_root that is different from web document root.

You can set the PHP script document root by the configuration directive doc_root in the configuration file, or you can set the environment variable PHP_DOCUMENT_ROOT. If it is set, the CGI version of PHP will always construct the file name to open with this *`doc_root`* and the path information in the request, so you can be sure no script is executed outside this directory (except for *`user_dir`* below).

Another option usable here is user_dir. When user_dir is unset, only thing controlling the opened file name is *`doc_root`*. Opening an url like `http://my.host/~user/doc.php` does not result in opening a file under users home directory, but a file called `~user/doc.php` under doc_root (yes, a directory name starting with a tilde [~]).

If user_dir is set to for example `public_php`, a request like `http://my.host/~user/doc.php` will open a file called `doc.php` under the directory named `public_php` under the home directory of the user. If the home of the user is `/home/user`, the file executed is `/home/user/public_php/doc.php`.

`user_dir` expansion happens regardless of the *`doc_root`* setting, so you can control the document root and user directory access separately.

Case 4: PHP parser outside of web tree

A very secure option is to put the PHP parser binary somewhere outside of the web tree of files. In `/usr/local/bin`, for example. The only real downside to this option is that you will now have to put a line similar to:

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

as the first line of any file containing PHP tags. You will also need to make the file executable. That is, treat it exactly as you would treat any other CGI script written in Perl or sh or any other common scripting language which uses the `#!` shell-escape mechanism for launching itself.

To get PHP to handle PATH_INFO and PATH_TRANSLATED information correctly with this setup, the php parser should be compiled with the --enable-discard-path configure option.

Installed as an Apache module

When PHP is used as an Apache module it inherits Apache’s user permissions (typically those of the "nobody" user). This has several impacts on security and authorization. For example, if you are using PHP to access a database, unless that database has built-in access control, you will have to make the database accessable to the "nobody" user. This means a malicious script could access and modify the database, even without a username and password. It’s entirely possible that a web spider could stumble across a database administrator’s web page, and drop all of your databases. You can protect against this with Apache authorization, or you can design your own access model using LDAP, .htaccess files, etc. and include that code as part of your PHP scripts.

Often, once security is established to the point where the PHP user (in this case, the apache user) has very little risk attached to it, it is discovered that PHP is now prevented from writing any files to user directories. Or perhaps it has been prevented from accessing or changing databases. It has equally been secured from writing good and bad files, or entering good and bad database transactions.

A frequent security mistake made at this point is to allow apache root permissions, or to escalate apache’s abilitites in some other way.

Escalating the Apache user’s permissions to root is extremely dangerous and may compromise the entire system, so sudo’ing, chroot’ing, or otherwise running as root should not be considered by those who are not security professionals.

There are some simpler solutions. By using **open_basedir()()** you can control and restrict what directories are allowed to be used for PHP. You can also set up apache-only areas, to restrict all web based activity to non-user, or non-system, files.

Filesystem Security

PHP is subject to the security built into most server systems with respect to permissions on a file and directory basis. This allows you to control which files in the filesystem may be read. Care should be taken with any files which are world readable to ensure that they are safe for reading by all users who have access to that filesystem.

Since PHP was designed to allow user level access to the filesystem, it’s entirely possible to write a PHP script that will allow you to read system files such as /etc/password, modify your ethernet connections, send massive printer jobs out, etc. This has some obvious implications, in that you need to ensure that the files that you read from and write to are the appropriate ones.

Consider the following script, where a user indicates that they’d like to delete a file in their home directory. This assumes a situation where a PHP web interface is regularly used for file management, so the Apache user is allowed to delete files in the user home directories.

Esempio 4-1. Poor variable checking leads to....

```
<?php
// remove a file from the user’s home directory
```

```
$username = $HTTP_POST_VARS['user_submitted_name'];
$homedir = "/home/$username";
$file_to_delete = "$userfile";
unlink ($homedir/$userfile);
echo "$file_to_delete has been deleted!";
?>
```

Since the username is postable from a user form, they can submit a username and file belonging to someone else, and delete files. In this case, you'd want to use some other form of authentication. Consider what could happen if the variables submitted were "../etc/" and "passwd". The code would then effectively read:

Esempio 4-2. ... A filesystem attack

```
<?php
// removes a file from anywhere on the hard drive that
// the PHP user has access to. If PHP has root access:
$username = "../etc/";
$homedir = "/home/../etc/";
$file_to_delete = "passwd";
unlink ("/home/../etc/passwd");
echo "/home/../etc/passwd has been deleted!";
?>
```

There are two important measures you should take to prevent these issues.

- Only allow limited permissions to the PHP web user binary.
- Check all variables which are submitted.

Here is an improved script:

Esempio 4-3. More secure file name checking

```
<?php
// removes a file from the hard drive that
// the PHP user has access to.
$username = $HTTP_SERVER_VARS['REMOTE_USER']; // using an authentica-
tion mechanisim

$homedir = "/home/$username";

$file_to_delete = basename("$userfile"); // strip paths
unlink ($homedir/$file_to_delete);
```

```
$fp = fopen("/home/logging/filedelete.log","+a"); //log the deletion
$logstring = "$username $homedir $file_to_delete";
fputs ($fp, $logstring);
fclose($fp);

echo "$file_to_delete has been deleted!";
?>
```

However, even this is not without it's flaws. If your authentication system allowed users to create their own user logins, and a user chose the login "../etc/", the system is once again exposed. For this reason, you may prefer to write a more customized check:

Esempio 4-4. More secure file name checking

```
<?php
$username = $HTTP_SERVER_VARS['REMOTE_USER']; // using an authentica-
tion mechanisim
$homedir = "/home/$username";

if (!ereg('^[^./][^/]*$', $userfile))
     die('bad filename'); //die, do not process

if (!ereg('^[^./][^/]*$', $username))
     die('bad username'); //die, do not process
//etc...
?>
```

Depending on your operating system, there are a wide variety of files which you should be concerned about, including device entries (/dev/ or COM1), configuration files (/etc/ files and the .ini files), well known file storage areas (/home/, My Documents), etc. For this reason, it's usually easier to create a policy where you forbid everything except for what you explicitly allow.

Error Reporting

With PHP security, there are two sides to error reporting. One is beneficial to increasing security, the other is detrimental.

A standard attack tactic involves profiling a system by feeding it improper data, and checking for the kinds, and contexts, of the errors which are returned. This allows the system cracker to probe for information about the server, to determine possible weaknesses. For example, if an attacker had gleaned information about a page based on a prior form submission, they may attempt to override variables, or modify them:

Esempio 4-5. Attacking Variables with a custom HTML page

```
<form method="post" action="attacktarget?username=badfoo&password=badfoo">
<input type="hidden" name="username" value="badfoo">
<input type="hidden" name="password" value="badfoo">
</form>
```

The PHP errors which are normally returned can be quite helpful to a developer who is trying to debug a script, indicating such things as the function or file that failed, the PHP file it failed in, and the line number which the failure occured in. This is all information that can be exploited. It is not uncommon for a php developer to use show_source(), highlight_string(), or highlight_file() as a debugging measure, but in a live site, this can expose hidden variables, unchecked syntax, and other dangerous information. Especially dangerous is running code from known sources with built-in debugging handlers, or using common debugging techniques. If the attacker can determine what general technique you are using, they may try to brute-force a page, by sending various common debugging strings:

Esempio 4-6. Exploiting common debugging variables

```
<form method="post" action="attacktarget?errors=Y&showerrors=1"&debug=1">
<input type="hidden" name="errors" value="Y">
<input type="hidden" name="showerrors" value="1">
<input type="hidden" name="debug" value="1">
</form>
```

Regardless of the method of error handling, the ability to probe a system for errors leads to providing an attacker with more information.

For example, the very style of a generic PHP error indicates a system is running PHP. If the attacker was looking at an .html page, and wanted to probe for the back-end (to look for known weaknesses in the system), by feeding it the wrong data they may be able to determine that a system was built with PHP.

A function error can indicate whether a system may be running a specific database engine, or give clues as to how a web page or programmed or designed. This allows for deeper investigation into open database ports, or to look for specific bugs or weaknesses in a web page. By feeding different pieces of bad data, for example, an attacker can determine the order of authentication in a script, (from the line number errors) as well as probe for exploits that may be exploited in different locations in the script.

A filesystem or general PHP error can indicate what permissions the webserver has, as well as the structure and organization of files on the web server. Developer written error code can aggravate this problem, leading to easy exploitation of formerly "hidden" information.

There are three major solutions to this issue. The first is to scrutinize all functions, and attempt to compensate for the bulk of the errors. The second is to disable error reporting entirely on the running

code. The third is to use PHP's custom error handling functions to create your own error handler. Depending on your security policy, you may find all three to be applicable to your situation.

One way of catching this issue ahead of time is to make use of PHP's own error_reporting(), to help you secure your code and find variable usage that may be dangerous. By testing your code, prior to deployment, with E_ALL, you can quickly find areas where your variables may be open to poisoning or modification in other ways. Once you are ready for deployment, by using E_NONE, you insulate your code from probing.

Esempio 4-7. Finding dangerous variables with E_ALL

```
<?php
if ($username) {  // Not initialized or checked before usage
    $good_login = 1;
}
if ($good_login == 1) { // If above test fails, not initial-
ized or checked before usage
    fpassthru ("/highly/sensitive/data/index.html");
}
?>
```

Using Register Globals

One feature of PHP that can be used to enhance security is configuring PHP with register_globals = off. By turning off the ability for any user-submitted variable to be injected into PHP code, you can reduce the amount of variable poisoning a potential attacker may inflict. They will have to take the additional time to forge submissions, and your internal variables are effectively isolated from user submitted data.

While it does slightly increase the amount of effort required to work with PHP, it has been argued that the benefits far outweigh the effort.

Esempio 4-8. Working without register_globals=off

```
<?php
if ($username) {  // can be forged by a user in get/post/cookies
    $good_login = 1;
}

if ($good_login == 1) { // can be forged by a user in get/post/cookies,
    fpassthru ("/highly/sensitive/data/index.html");
}
?>
```

Esempio 4-9. Working with register_globals = off

```
<?php
if($HTTP_COOKIE_VARS[’username’]){
    // can only come from a cookie, forged or otherwise
    $good_login = 1;
    fpassthru ("/highly/sensitive/data/index.html");
}
?>
```

By using this wisely, it’s even possible to take preventative measures to warn when forging is being attempted. If you know ahead of time exactly where a variable should be coming from, you can check to see if submitted data is coming from an inappropriate kind of submission. While it doesn’t guarantee that data has not been forged, it does require an attacker to guess the right kind of forging.

Esempio 4-10. Detecting simple variable poisoning

```
<?php
if ($HTTP_COOKIE_VARS[’username’] &&
    !$HTTP_POST_VARS[’username’] &&
    !$HTTP_GET_VARS[’username’] ) {
    // Perform other checks to validate the user name...
    $good_login = 1;
    fpassthru ("/highly/sensitive/data/index.html");
} else {
   mail("admin@example.com", "Possible breakin at-
tempt", $HTTP_SERVER_VARS[’REMOTE_ADDR’]);
   echo "Security violation, admin has been alerted.";
   exit;
}
?>
```

Of course, simply turning on register globals does not mean code is secure. For every piece of data that is submitted, it should also be checked in other ways.

User Submitted Data

The greatest weakness in many PHP programs is not inherent in the language itself, but merely an issue of code not being written with security in mind. For this reason, you should always take the time to consider the implications of a given piece of code, to ascertain the possible damage if an unexpected variable is submitted to it.

Esempio 4-11. Dangerous Variable Usage

```
<?php
// remove a file from the user’s home directory... or maybe
// somebody else’s?
unlink ($evil_var);

// Write logging of their access... or maybe an /etc/password entry?
fputs ($fp, $evil_var);

// Execute something trivial.. or rm -rf *?
system ($evil_var);
exec ($evil_var);

?>
```

You should always carefully examine your code to make sure that any variables being submitted from a web browser are being properly checked, and ask yourself the following questions:

- Will this script only affect the intended files?
- Can unusual or undesirable data be acted upon?
- Can this script be used in unintended ways?
- Can this be used in conjunction with other scripts in a negative manner?
- Will any transactions be adequately logged?

By adequately asking these questions while writing the script, rather than later, you prevent an unfortunate re-write when you need to increase your security. By starting out with this mindset, you won’t guarantee the security of your system, but you can help improve it.

You may also want to consider turning off register_globals, magic_quotes, or other convenience settings which may confuse you as to the validity, source, or value of a given variable. Working with PHP in error_reporting(E_ALL) mode can also help warn you about variables being used before they are checked or initialized (so you can prevent unusual data from being operated upon).

Hiding PHP

A few simple techniques can help to hide PHP, possibly slowing down an attacker who is attempting to disover weaknesses in your system. By setting expose_php = off in your php.ini file, you reduce the amount of information available to them.

Another tactic is to configure web servers such as apache to parse different filetypes through PHP, either with an .htaccess directive, or in the apache configuration file itself. You can then use misleading file extensions:

Esempio 4-12. Hiding PHP as another language

```
# Make PHP code look like other code types
AddType application/x-httpd-php .asp .py .pl
```

Or obscure it completely:

Esempio 4-13. Using unknown types for PHP extensions

```
# Make PHP code look like unknown types
AddType application/x-httpd-php .bop .foo .133t
```

Or hide it as html code, which has a slight performance hit because all html will be parsed through the PHP engine:

Esempio 4-14. Using html types for PHP extensions

```
# Make all PHP code look like html
AddType application/x-httpd-php .htm .html
```

For this to work effectively, you must rename your PHP files with the above extensions. While it is a form of security through obscurity, it’s a minor preventative measure with few drawbacks.

General considerations

A completely secure system is a virtual impossibility, so an approach often used in the security profession is one of balancing risk and usability. If every variable submitted by a user required two forms of biometric validation (such as a retinal scan and a fingerprint), you would have an extremely high level of accountability. It would also take half an hour to fill out a fairly complex form, which would tend to encourage users to find ways of bypassing the security.

The best security is often inobtrusive enough to suit the requirements without the user being prevented from accomplishing their work, or over-burdening the code author with excessive complexity. Indeed, some security attacks are merely exploits of this kind of overly built security, which tends to erode over time.

A phrase worth remembering: A system is only as good as the weakest link in a chain. If all transactions are heavily logged based on time, location, transaction type, etc. but the user is only verified based on a single cookie, the validity of tying the users to the transaction log is severely weakened.

When testing, keep in mind that you will not be able to test all possibilities for even the simplest of pages. The input you may expect will be completely unrelated to the input given by a disgruntled employee, a cracker with months of time on their hands, or a housecat walking across the keyboard.

This is why it’s best to look at the code from a logical perspective, to discern where unexpected data can be introduced, and then follow how it is modified, reduced, or amplified.

The Internet is filled with people trying to make a name for themselves by breaking your code, crashing your site, posting inappropriate content, and otherwise making your day interesting. It doesn’t matter if you have a small or large site, you are a target by simply being online, by having a server that can be connected to. Many cracking programs do not discern by size, they simply trawl massive IP blocks looking for victims. Try not to become one.

Keeping Current

PHP, like any other large system, is under constant scrutiny and improvement. Each new version will often include both major and minor changes to enhance and repair security flaws, configuration mishaps, and other issues that will affect the overall security and stability of your system.

Like other system-level scripting languages and programs, the best approach is to update often, and maintain awareness of the latest versions and their changes.

II. Struttura del Linguaggio

Capitolo 5. Sintassi Fondamentale

Modi per uscire dalla modalità HTML

Esistono quattro modi per passare dalla modalità HTML alla modalità PHP:

Esempio 5-1. Metodi per uscire dalla modalità HTML

```
1.  <? echo ("questo è il più semplice, ovvero un'istruzione di elabo-
razione SGML\n"); ?>

2.  <?php echo("se vuoi produrre documenti XML, uti-
lizza questo modo\n"); ?>

3.  <script language="php">
        echo ("alcuni editor (tipo Front-
Page) non amano le istruzioni di elaborazione");
    </script>

4.  <% echo ("Opzionalmente puoi utilizzare tag nello stile ASP"); %>
    <%= $variable; # Questo è una abbreviazione per "<%echo .." %>
```

Il primo è disponibile solo se sono stati abilitati i tags abbreviati. Ciò può essere impostato sia utilizzando la funzione **short_tags()**, che abilitando nel file di configurazione del PHP l'opzione short_open_tag, oppure compilando il PHP utilizzando l'opzione --enable-short-tags nel comando **configure**.

Il quarto modo è disponibile solo se sono stati attivati i tag in stile ASP tramite l'opzione asp_tags nel file di configurazione.

Nota: Il supporto per i tag nello stile ASP è stato aggiunto nella versione 3.0.4.

I tag di chiusura del blocco includono, se presente, il carattere di newline immediatamente sucessivo.

Separazione delle istruzioni

Le istruzioni sono separate come nel C o in perl - ogni istruzione termina con un punto e virgola.

Il tag di chiusura (?>) implica anche la fine di un'istruzione, perciò le seguenti sono equivalenti:

```
<?php
    echo "Questo ‘ un test";
```

```
?>

<?php echo "Questo ‘ un test" ?>
```

Commenti

PHP supporta i commenti dei linguaggi ’C’, ’C++’ e della shell Unix. Per esempio:

```
<?php
    echo "Questo ‘ un test"; // Questo è un com-
mento su una linea nella stile c++
    /* Questo è un commento su più’; linee
       ancora un’altra linea di commento */
    echo "Questo è un altro test";
    echo "Un ultimo test"; # Questo è un commento stile shell Unix
?>
```

Lo stile di commento su "una linea", attualmente commenta solo fino alla fine della linea o del blocco corrente di codice PHP.

```
<h1>Questo è un <?# echo "semplice";?> esempio.</h1>
<p>L’intestazione qui sopra dirà ’Questo è un esempio’.
```

Occorre fare attenzione nel non annidare i commenti di stile C, situazione che si presenta quando si commentano larghi blocchi di codice.

```
<?php
 /*
    echo "Questo è un test"; /* Questo commento causerà dei problemi */
 */
?>
```

Capitolo 6. Types

Introduction

PHP supports eight primitive types.

Four scalar types:

- boolean
- integer
- floating-point number (float)
- string

Two compound types:

- array
- object

And finally two special types:

- resource
- `NULL`

> **Nota:** In this manual you'll often find `mixed` parameters. This pseudo-type indicates multiple possiblities for that parameter.

The type of a variable is usually not set by the programmer; rather, it is decided at runtime by PHP depending on the context in which that variable is used.

If you would like to force a variable to be converted to a certain type, you may either cast the variable or use the settype() function on it.

Note that a variable may behave in different manners in certain situations, depending on what type it is at the time. For more information, see the section on Type Juggling.

Booleans

This is the easiest type. A boolean expresses a truth value. It can be either `TRUE` or `FALSE`.

> **Nota:** The boolean-type was introduced in PHP 4.

Syntax

To specify a boolean-literal, use either the keyword `TRUE` or `FALSE`. Both are case-insensitive.

```
$foo = True; // assign the value TRUE to $foo
```

Usually you use some kind of operator which returns a boolean value, and then pass it on to a control structure.

```
if ($action == "show_version") { // == is an operator which re-
turns a boolean
    echo "The version is 1.23";
}

// this is not necessary:
if ($show_separators == true) {
    echo "<hr>\n";
}

// because you can simply type this:
if ($show_separators) {
    echo "<hr>\n";
}
```

Converting to boolean

To explicitly convert a value to boolean, use either the `(bool)` or the `(boolean)` cast. However, in most cases you do not need to use the cast, since a value will be automatically converted if an operator, function or control structure requires a boolean argument.

See also Type Juggling.

When converting to boolean, the following values are considered `FALSE`:

- the boolean `FALSE`
- the integer 0 (zero)
- the float 0.0 (zero)
- the empty string, and the string "0"
- an array with zero elements
- an object with zero elements
- the special value `NULL`

Every other value is considered `TRUE` (including any resource).

> **Attenzione**
>
> `-1` is considered `TRUE`, like any other non-zero (whether negative or positive) number!

Integers

An integer is a number of the set Z = {..., -2, -1, 0, 1, 2, ...}.

See also: Arbitrary precision integers and Floating point numbers

Syntax

Integers can be specified in decimal (10-based), hexadecimal (16-based) or octal (8-based) notation, optionally preceded by a sign (- or +).

If you use the octal notation, you must precede the number with a `0` (zero), to use hexadecimal notation precede the number with `0x`.

Esempio 6-1. Integer literals

```
$a = 1234; # decimal number
$a = -123; # a negative number
$a = 0123; # octal number (equivalent to 83 decimal)
$a = 0x1A; # hexadecimal number (equivalent to 26 decimal)
```

The size of an integer is platform-dependent, although a maximum value of about two billion is the usual value (that’s 32 bits signed). PHP does not support unsigned integers.

Integer overflow

If you specify a number beyond the bounds of the integer-type, it will be interpreted as a float instead.

In PHP there is also no such thing as integer division. `1/2` yields the float `0.5`.

```
$large_number =  2147483647;
var_dump($large_number);
// output: int(2147483647)
$large_number =  2147483648;
var_dump($large_number);
```

```
// output: float(2147483648)

// this goes also for hexadecimal specified integers:

var_dump( 0x80000000 );
// output: float(2147483648)

var_dump( 25/7 );
// output: float(3.5714285714286)
```

Furthermore, if some function or operator yields a number that is beyond the boundaries of integer, it will also be automatically converted to float.

```
$million = 1000000;
$large_number =  50000 * $million;
var_dump($large_number);
// output: float(50000000000)
```

> **Attenzione**
>
> Unfortunately, there was a bug in PHP so that this does not always work correctly when there are negative numbers involved. For example: when you do `-50000 * $million`, the result will be `-429496728`. However, when both operands are positive there is no problem.
>
> This is solved in PHP 4.0.7

Converting to integer

To explicitly convert a value to integer, use either the `(int)` or the `(integer)` cast. However, in most cases you do not need to use the cast, since a value will be autmatically converted if an operator, function or control structure requires a integer-argument.

See also type-juggling.

From booleans

`FALSE` will yield `0` (zero), and `TRUE` will yield `1` (one).

From floating point numbers

When converting from float to integer, the number will be rounded *towards zero*.

If the float is beyond the boundaries of integer (usually `+/- 2.15e+9 = 2^31`), the result is undefined, since the float hasn’t got enough precision to give an exact integer result. No warning, not even a notice will be issued in this case!

> **Attenzione**
>
> Never cast an unknown fraction to integer, as this can sometimes lead to unexpected results.
>
> ```
> echo (int) ( (0.1+0.7) * 10 ); // echoes 7!
> ```
>
> See for more information the warning about float-precision.

From strings

See String conversion

From other types

> **Cautela**
>
> Behaviour of converting to integer is undefined for other types. Currently, the behaviour is the same as if the value was first converted to boolean. However, do *not* relay on this behaviour, as it can change without notice.

Floating point numbers

Floating point numbers (AKA "floats", "doubles" or "real numbers") can be specified using any of the following syntaxes:

```
$a = 1.234; $a = 1.2e3; $a = 7E-10;
```

The size of a float is platform-dependent, although a maximum of ~1.8e308 with a precision of roughly 14 decimal digits is a common value (that’s 64 bit IEEE format).

Floating point precision

It is quite usual that simple decimal fractions like `0.1` or `0.7` cannot be converted into their internal binary counterparts without a little loss of precision. This can lead to confusing results: for example, `floor((0.1+0.7)*10)` will usually return `7` instead of the expected `8` as the result of the internal representation really being something like `7.9999999999...`.

This is related to the fact that it is impossible to exactly express some fractions in decimal notation with a finite number of digits. For instance, `1/3` in decimal form becomes `0.3333333. . ..`

So never trust floating number results to the last digit and never compare floating point numbers for equality. If you really need higher precision, you should use the arbitrary precision math functions or gmp functions instead.

Strings

A string is series of characters. In PHP, a character is the same as a byte, that is, there are exactly 256 different characters possible. This also implies that PHP has no native support of Unicode.

> **Nota:** It is no problem for a string to become very large. There is no practical bound to the size of strings imposed by PHP, so there is no reason at all to worry about long strings.

Syntax

A string literal can be specified in three different ways.

- single quoted
- double quoted
- heredoc syntax

Single quoted

The easiest way to specify a simple string is to enclose it in single quotes (the character `'`).

To specify a literal single quote, you will need to escape it with a backslash (\), like in many other languages. If a backslash needs to occur before a single quote or at the end of the string, you need to double it. Note that if you try to escape any other character, the backslash too will be printed! So usually there is no need to escape the backslash itself.

> **Nota:** In PHP 3, a warning will be issued at the `E_NOTICE` level when this happens.

> **Nota:** Unlike the two other syntaxes, variables will *not* be expanded when they occur in single quoted strings.

```
echo 'this is a simple string';
echo 'You can also have embedded newlines in strings,
like this way.';
echo 'Arnold once said: "I\'ll be back"';
// output: ... "I'll be back"
echo 'Are you sure you want to delete C:\\*.*?';
// output: ... delete C:\*.*?
echo 'Are you sure you want to delete C:\*.*?';
// output: ... delete C:\*.*?
echo 'I am trying to include at this point: \n a newline';
// output: ... this point: \n a newline
```

Double quoted

If the string is enclosed in double-quotes ("), PHP understands more escape sequences for special characters:

Tabella 6-1. Escaped characters

sequence	meaning

Again, if you try to escape any other character, the backspace will be printed too!

But the most important pre of double-quoted strings is the fact that variable names will be expanded. See string parsing for details.

Heredoc

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. Also, the identifier used must follow the same naming rules as any other label in PHP: it must contain only alphanumeric characters and underscores, and must start with a non-digit character or underscore.

> **Attenzione**
>
> It is very important to note that the line with the closing identifier contains no other characters, except *possibly* a semicolon (`;`). That means especially that the identifier *may not be indented*, and there may not be any spaces or tabs after or before the semicolon.
>
> Probably the nastiest gotcha is that there may also not be a carriage return (`\r`) at the end of the line, only a form feed, AKA newline (`\n`). Since Microsoft Windows uses the sequence `\r\n` as a line terminator, your heredoc may not work if you write your script in a Windows editor. However, most programming editors provide a way to save your files with a UNIX line terminator.

Here doc text behaves just like a double-quoted string, without the double-quotes. This means that you do not need to escape quotes in your here docs, but you can still use the escape codes listed above. Variables are expanded, but the same care must be taken when expressing complex variables inside a here doc as with strings.

Esempio 6-2. Here doc string quoting example

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

/* More complex example, with variables. */
class foo
{
    var $foo;
    var $bar;

    function foo()
    {
        $this->foo = 'Foo';
        $this->bar = array('Bar1', 'Bar2', 'Bar3');
    }
}

$foo = new foo();
$name = 'MyName';

echo <<<EOT
My name is "$name". I am printing some $foo->foo.
Now, I am printing some {$foo->bar[1]}.
This should print a capital 'A': \x41
EOT;
```

```
?>
```

Nota: Here doc support was added in PHP 4.

Variable parsing

When a string is specified in double quotes or with heredoc, variables are parsed within it.

There are two types of syntax, a simple one and a complex one. The simple syntax is the most common and convenient, it provides a way to parse a variable, an array value, or an object property.

The complex syntax was introduced in PHP 4, and can by recognised by the curly braces surrounding the expression.

Simple syntax

If a dollar sign (`$`) is encountered, the parser will greedily take as much tokens as possible to form a valid variable name. Enclose the variable name in curly braces if you want to explicitly specify the end of the name.

```
$beer = 'Heineken';
echo "$beer's taste is great"; // works, "'" is an invalid charac-
ter for varnames
echo "He drunk some $beers"; // won't work, 's' is a valid charac-
ter for varnames
echo "He drunk some ${beer}s"; // works
```

Similary, you can also have an array index or an object property parsed. With array indices, the closing square bracket (`]`) marks the end of the index. For object properties the same rules apply as to simple variables, though with object properties there doesn't exist a trick like the one with variables.

```
$fruits = array( 'strawberry' => 'red' , 'banana' => 'yellow' );
echo "A banana is $fruits[banana].";  // note that this works differently
outside string-quotes. See $foo[bar] outside strings
echo "This square is $square->width meters broad.";
echo "This square is $square->width00 centimeters broad."; // won't work,
   // for a solution, see the complex syntax.
```

For anything more complex, you should use the complex syntax.

Complex (curly) syntax

This isn’t called complex because the syntax is complex, but because you can include complex expressions this way.

In fact, you can include any value that is in the namespace in strings with this syntax. You simply write the expression the same way as you would outside the string, and then include it in { and }. Since you can’t escape ’{’, this syntax will only be recognised when the $ is immediately following the {. (Use "{\$" or "\{$" to get a literal "{$"). Some examples to make it clear:

```
$great = ’fantastic’;
echo "This is { $great}"; // won’t work, outputs: This is { fantastic}
echo "This is {$great}";  // works, outputs: This is fantastic
echo "This square is {$square->width}00 centimeters broad.";
echo "This works: {$arr[4][3]}";
echo "This is wrong: {$arr[foo][3]}"; // for the same reason
   // as $foo[bar] is wrong outside a string.
echo "You should do it this way: {$arr[’foo’][3]}";
echo "You can even write {$obj->values[3]->name}";
echo "This is the value of the var named $name: {${$name}}";
```

String access by character

Characters within strings may be accessed by specifying the zero-based offset of the desired character after the string in curly braces.

> **Nota:** For backwards compatibility, you can still use the array-braces. However, this syntax is deprecated as of PHP 4.

Esempio 6-3. Some string 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};
?>
```

Useful functions

Strings may be concatenated using the '.' (dot) operator. Note that the '+' (addition) operator will not work for this. Please see String operators for more information.

There are a lot of useful functions for string modification.

See the string functions section for general functions, the regular expression functions for advanced find&replacing (in two tastes: Perl and POSIX extended).

There are also functions for URL-strings, and functions to encrypt/decrypt strings (mcrypt and mhash).

Finally, if you still didn't find what you're looking for, see also the character type functions.

String conversion

When a string is evaluated as a numeric value, the resulting value and type are determined as follows.

The string will evaluate as a float if it contains any of the characters '.', 'e', or 'E'. Otherwise, it will evaluate as an integer.

The value is given by the initial portion of the string. If the string starts with valid numeric data, this will be the value used. Otherwise, the value will be 0 (zero). Valid numeric data is an optional sign, followed by one or more digits (optionally containing a decimal point), followed by an optional exponent. The exponent is an 'e' or 'E' followed by one or more digits.

When the first expression is a string, the type of the variable will depend on the second expression.

```
$foo = 1 + "10.5";              // $foo is float (11.5)
```

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

For more information on this conversion, see the Unix manual page for 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";
```

Arrays

An array in PHP is actually an ordered map. A map is a type that maps *values* to *keys*. This type is optimized in several ways, so you can use it as a real array, or a list (vector), hashtable (which is an implementation of a map), dictionary, collection, stack, queue and probably more. Because you can have another PHP-array as a value, you can also quite easily simulate trees.

Explanation of those structures is beyond the scope of this manual, but you'll find at least one example for each of those structures. For more information about those structures, we refer you to external literature about this broad topic.

Syntax

Specifying with array()

An array can be created by the array() language-construct. It takes a certain number of comma-separated `key => value` pairs.

A `key` is either a nonnegative integer or a string. If a key is the standard representation of a non-negative integer, it will be interpreted as such (i.e. `'8'` will be interpreted as `8`, while `'08'` will be interpreted as `'08'`).

A value can be anything.

Omitting keys. If you omit a key, the maximum of the integer-indices is taken, and the new key will be that maximum + 1. If no integer-indices exist yet, the key will be `0` (zero). If you specify a key that already has a value assigned to it, that value will be overwritten.

```
array( [key =>] value
     , ...
     )
// key is either string or nonnegative integer
// value can be anything
```

Creating/modifying with square-bracket syntax

You can also modify an existing array, by explicitly setting values.

This is done by assigning values to the array while specifying the key in brackets. You can also omit the key, add an empty pair of brackets ("[]") to the variable-name in that case.

```
$arr[key] = value;
$arr[] = value;
// key is either string or nonnegative integer
// value can be anything
```

If $arr doesn't exist yet, it will be created. So this is also an alternative way to specify an array. To change a certain value, just assign a new value to it. If you want to remove a key/value pair, you need to unset() it.

Useful functions

There are quite some useful function for working with arrays, see the array-functions section.

The foreach control structure exists specificly for arrays. It provides an easy way to traverse an array.

Array do's and don'ts

Why is $foo[bar] wrong?

You might have seen the following syntax in old scripts:

```
$foo[bar] = 'enemy';
echo $foo[bar];
// etc
```

This is wrong, but it works. Then, why is it wrong? The reason is that, as stated in the syntax section, there must be an expression between the square brackets ('[' and ']'). That means that you can write things like this:

```
echo $arr[ foo(true) ];
```

This is an example of using a function return value as the array index. PHP knows also about constants, and you may have seen the `E_*` before.

```
$error_descriptions[E_ERROR] = "A fatal error has occured";
$error_descriptions[E_WARNING] = "PHP issued a warning";
$error_descriptions[E_NOTICE] = "This is just an informal notice";
```

Note that `E_ERROR` is also a valid identifier, just like `bar` in the first example. But the last example is in fact the same as writing:

```
$error_descriptions[1] = "A fatal error has occured";
$error_descriptions[2] = "PHP issued a warning";
$error_descriptions[8] = "This is just an informal notice";
```

because `E_ERROR` equals `1`, etc.

Then, how is it possible that `$foo[bar]` works? It works, because `bar` is due to its syntax expected to be a constant expression. However, in this case no constant with the name `bar` exists. PHP now assumes that you meant `bar` literally, as the string `"bar"`, but that you forgot to write the quotes.

So why is it bad then?

At some point in the future, the PHP team might want to add another constant or keyword, and then you get in trouble. For example, you already cannot use the words `empty` and `default` this way, since they are special keywords.

And, if these arguments don't help: this syntax is simply deprecated, and it might stop working some day.

> **Suggerimento:** When you turn error_reporting to `E_ALL`, you will see that PHP generates warnings whenever this construct is used. This is also valid for other deprecated 'features'. (put the line `error_reporting(E_ALL);` in your script)

> **Nota:** Inside a double-quoted string, an other syntax is valid. See variable parsing in strings for more details.

Examples

The array-type in PHP is very versatile, so here will be some examples to show you the full power of arrays.

```
// this
$a = array( 'color' => 'red'
          , 'taste' => 'sweet'
          , 'shape' => 'round'
          , 'name'  => 'apple'
          ,            4        // key will be 0
          );

// is completely equivalent with
$a['color'] = 'red';
$a['taste'] = 'sweet';
$a['shape'] = 'round';
$a['name'] = 'apple';
$a[]        = 4;        // key will be 0

$b[] = 'a';
$b[] = 'b';
$b[] = 'c';
// will result in the array array( 0 => 'a' , 1 => 'b' , 2 => 'c' ),
// or simply array('a', 'b', 'c')
```

Esempio 6-4. Using array()

```
// Array as (property-)map
$map = array( 'version'    => 4
            , 'OS'         => 'Linux'
            , 'lang'       => 'english'
            , 'short_tags' => true
            );

// strictly numerical keys
$array = array( 7
              , 8
              , 0
              , 156
              , -10
              );
// this is the same as array( 0 => 7, 1 => 8, ...)

$switching = array(         10 // key = 0
                  , 5    =>  6
                  , 3    =>  7
                  , 'a'  =>  4
```

```
                  ,         11 // key = 6 (maximum of integer-
indices was 5)
                  , '8'  =>  2 // key = 8 (integer!)
                  , '02' => 77 // key = '02'
                  , 0    => 12 // the value 10 will be overwritten by 12
                  );


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

Esempio 6-5. Collection

```
$colors = array('red','blue','green','yellow');

foreach ( $colors as $color ) {
    echo "Do you like $color?\n";
}

/* output:
Do you like red?
Do you like blue?
Do you like green?
Do you like yellow?
*/
```

Note that it is currently not possible to change the values of the array directly in such a loop. A workaround is the following:

Esempio 6-6. Collection

```
foreach ( $colors as $key => $color ) {
    // won't work:
    //$color = strtoupper($color);

    //works:
    $colors[$key] = strtoupper($color);
}
print_r($colors);

/* output:
Array
(
    [0] => RED
    [1] => BLUE
```

```
    [2] => GREEN
    [3] => YELLOW
)
*/
```

This example creates a one-based array.

Esempio 6-7. One-based index

```
$firstquarter  = array(1 => 'January', 'February', 'March');
print_r($firstquarter);

/* output:
Array
(
    [1] => 'January'
    [2] => 'February'
    [3] => 'March'
)
*/
```

Esempio 6-8. Filling real array

```
// fill an array with all items from a directory
$handle = opendir('.');
while ($file = readdir($handle))
{
    $files[] = $file;
}
closedir($handle);
```

Arrays are ordered. You can also change the order using various sorting-functions. See array-functions for more information.

Esempio 6-9. Sorting array

```
sort($files);
print_r($files);
```

Because the value of an array can be everything, it can also be another array. This way you can make recursive and multi-dimensional arrays.

Esempio 6-10. Recursive and multi-dimensional arrays

```
$fruits = array ( "fruits"  => array ( "a" => "orange"
                                     , "b" => "banana"
                                     , "c" => "apple"
                                     )
                , "numbers" => array ( 1
                                     , 2
                                     , 3
                                     , 4
                                     , 5
                                     , 6
                                     )
                , "holes"   => array (      "first"
                                     , 5 => "second"
                                     ,      "third"
                                     )
                );
```

Objects

Object Initialization

To initialize an object, you use the new statement to instantiate the object to a variable.

```
<?php
class foo
{
    function do_foo()
    {
        echo "Doing foo.";
    }
}

$bar = new foo;
$bar->do_foo();
?>
```

For a full discussion, please read the section Classes and Objects.

Resource

A resource is a special variable, holding a reference to an external resource. Resources are created and used by special functions. See the appendix for a listing of all these functions and the corresponding resource-types.

> **Nota:** The resource-type was introduced in PHP 4

Freeing resources

Due to the reference-counting system introduced with PHP4's Zend-engine, it is automatically detected when a resource is no longer referred to (just like Java). When this is the case, all resources that were in use for this resource are made free by the garbage collector. For this reason, it is rarely ever necessary to free the memory manually by using some free_result function.

> **Nota:** Persistent database-links are special, they are *not* destroyed by the gc. See also persistent links

`NULL`

The special `NULL` value represents that a variable has no value.

> **Nota:** The null-type was introduced in PHP 4

Syntax

There is only one value of type `NULL`, and that is the case-insensitive keyword `NULL`.

```
$var = Null;
```

Type Juggling

PHP does not require (or support) explicit type definition in variable declaration; a variable's type is determined by the context in which that variable is used. That is to say, if you assign a string value to variable *`var`*, *`var`* becomes a string. If you then assign an integer value to *`var`*, it becomes an integer.

An example of PHP's automatic type conversion is the addition operator '+'. If any of the operands is a float, then all operands are evaluated as floats, and the result will be a float. Otherwise, the operands will be interpreted as integers, and the result will also be an integer. Note that this does NOT change the types of the operands themselves; the only change is in how the operands are evaluated.

```
$foo = "0";  // $foo is string (ASCII 48)

$foo += 2;   // $foo is now an integer (2)
$foo = $foo + 1.3;  // $foo is now a float (3.3)
$foo = 5 + "10 Little Piggies"; // $foo is integer (15)
$foo = 5 + "10 Small Pigs";     // $foo is integer (15)
```

If the last two examples above seem odd, see String conversion.

If you wish to force a variable to be evaluated as a certain type, see the section on Type casting. If you wish to change the type of a variable, see settype().

If you would like to test any of the examples in this section, you can use the var_dump() function.

> **Nota:** 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.

Type Casting

Type casting in PHP works much as it does in C: the name of the desired type is written in parentheses before the variable which is to be cast.

```
$foo = 10;   // $foo is an integer
```

```
$bar = (float) $foo;   // $bar is a float
```

The casts allowed are:

- (int), (integer) - cast to integer
- (bool), (boolean) - cast to boolean
- (float), (double), (real) - cast to float
- (string) - cast to string
- (array) - cast to array
- (object) - cast to object

> **Suggerimento:** Instead of casting a variable to string, you can also enclose the variable in double quotes.

Note that tabs and spaces are allowed inside the parentheses, so the following are functionally equivalent:

```
$foo = (int) $bar;
$foo = ( int ) $bar;
```

It may not be obvious exactly what will happen when casting between certain types. For more info, see these sections:

- Converting to boolean
- Converting to integer

When casting or forcing a conversion from array to string, the result will be the word `Array`. When casting or forcing a conversion from object to string, the result will be the word `Object`.

When casting from a scalar or a string variable to an array, the variable will become the first element of the array:

```
$var = 'ciao';
$arr = (array) $var;
echo $arr[0];  // outputs 'ciao'
```

When casting from a scalar or a string variable to an object, the variable will become an attribute of the object; the attribute name will be 'scalar':

```
$var = 'ciao';
$obj = (object) $var;
echo $obj->scalar;  // outputs 'ciao'
```

Capitolo 7. Variables

Basics

Variables in PHP are represented by a dollar sign followed by the name of the variable. The variable name is case-sensitive.

Variable names follow the same rules as other labels in PHP. A valid variable name starts with a letter or underscore, followed by any number of letters, numbers, or underscores. As a regular expression, it would be expressed thus: ’[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*’

> **Nota:** For our purposes here, a letter is a-z, A-Z, and the ASCII characters from 127 through 255 (0x7f-0xff).

```
$var = "Bob";
$Var = "Joe";
echo "$var, $Var";      // outputs "Bob, Joe"

$4site = ’not yet’;     // invalid; starts with a number
$_4site = ’not yet’;    // valid; starts with an underscore
$täyte = ’mansikka’;    // valid; ’ä’ is ASCII 228.
```

In PHP 3, variables are always assigned by value. That is to say, when you assign an expression to a variable, the entire value of the original expression is copied into the destination variable. This means, for instance, that after assigning one variable’s value to another, changing one of those variables will have no effect on the other. For more information on this kind of assignment, see Expressions.

PHP 4 offers another way to assign values to variables: *assign by reference*. This means that the new variable simply references (in other words, "becomes an alias for" or "points to") the original variable. Changes to the new variable affect the original, and vice versa. This also means that no copying is performed; thus, the assignment happens more quickly. However, any speedup will likely be noticed only in tight loops or when assigning large arrays or objects.

To assign by reference, simply prepend an ampersand (&) to the beginning of the variable which is being assigned (the source variable). For instance, the following code snippet outputs ’My name is Bob’ twice:

```
<?php
$foo = ’Bob’;              // Assign the value ’Bob’ to $foo
$bar = &$foo;          // Reference $foo via $bar.
$bar = "My name is $bar";  // Alter $bar...
echo $foo;                 // $foo is altered too.
```

```
echo $bar;
?>
```

One important thing to note is that only named variables may be assigned by reference.

```
<?php
$foo = 25;
$bar = &$foo;      // This is a valid assignment.
$bar = &(24 * 7);  // Invalid; references an unnamed expression.

function test()
{
   return 25;
}

$bar = &test();    // Invalid.
?>
```

Predefined variables

PHP provides a large number of predefined variables to any script which it runs. Many of these variables, however, cannot be fully documented as they are dependent upon which server is running, the version and setup of the server, and other factors. Some of these variables will not be available when PHP is run on the command-line.

Despite these factors, here is a list of predefined variables available under a stock installation of PHP 3 running as a module under a stock installation of Apache (http://www.apache.org/) 1.3.6.

For a list of all predefined variables (and lots of other useful information), please see (and use) phpinfo().

> **Nota:** This list is neither exhaustive nor intended to be. It is simply a guideline as to what sorts of predefined variables you can expect to have access to in your script.

Apache variables

These variables are created by the Apache (http://www.apache.org/) webserver. If you are running another webserver, there is no guarantee that it will provide the same variables; it may omit some, or provide others not listed here. That said, a large number of these variables are accounted for in the CGI 1.1 specification (http://hoohoo.ncsa.uiuc.edu/cgi/env.html), so you should be able to expect those.

Note that few, if any, of these will be available (or indeed have any meaning) if running PHP on the command line.

$GATEWAY_INTERFACE

What revision of the CGI specification the server is using; i.e. 'CGI/1.1'.

$SERVER_NAME

The name of the server host under which the current script is executing. If the script is running on a virtual host, this will be the value defined for that virtual host.

$SERVER_SOFTWARE

Server identification string, given in the headers when responding to requests.

$SERVER_PROTOCOL

Name and revision of the information protocol via which the page was requested; i.e. 'HTTP/1.0';

$REQUEST_METHOD

Which request method was used to access the page; i.e. 'GET', 'HEAD', 'POST', 'PUT'.

$QUERY_STRING

The query string, if any, via which the page was accessed.

$DOCUMENT_ROOT

The document root directory under which the current script is executing, as defined in the server's configuration file.

$HTTP_ACCEPT

Contents of the `Accept:` header from the current request, if there is one.

$HTTP_ACCEPT_CHARSET

Contents of the `Accept-Charset:` header from the current request, if there is one. Example: 'iso-8859-1,*,utf-8'.

$HTTP_ACCEPT_ENCODING

Contents of the `Accept-Encoding:` header from the current request, if there is one. Example: 'gzip'.

$HTTP_ACCEPT_LANGUAGE

Contents of the `Accept-Language:` header from the current request, if there is one. Example: 'en'.

$HTTP_CONNECTION

Contents of the `Connection:` header from the current request, if there is one. Example: 'Keep-Alive'.

$HTTP_HOST

Contents of the `Host:` header from the current request, if there is one.

$HTTP_REFERER

The address of the page (if any) which referred the browser to the current page. This is set by the user's browser; not all browsers will set this.

$HTTP_USER_AGENT

Contents of the `User_Agent:` header from the current request, if there is one. This is a string denoting the browser software being used to view the current page; i.e. `Mozilla/4.5 [en] (X11; U; Linux 2.2.9 i586)`. Among other things, you can use this value with get_browser() to tailor your page's functionality to the capabilities of the user's browser.

$REMOTE_ADDR

The IP address from which the user is viewing the current page.

$REMOTE_PORT

The port being used on the user's machine to communicate with the web server.

$SCRIPT_FILENAME

The absolute pathname of the currently executing script.

$SERVER_ADMIN

The value given to the SERVER_ADMIN (for Apache) directive in the web server configuration file. If the script is running on a virtual host, this will be the value defined for that virtual host.

$SERVER_PORT

The port on the server machine being used by the web server for communication. For default setups, this will be '80'; using SSL, for instance, will change this to whatever your defined secure HTTP port is.

$SERVER_SIGNATURE

String containing the server version and virtual host name which are added to server-generated pages, if enabled.

$PATH_TRANSLATED

Filesystem- (not document root-) based path to the current script, after the server has done any virtual-to-real mapping.

$SCRIPT_NAME

Contains the current script's path. This is useful for pages which need to point to themselves.

$REQUEST_URI

The URI which was given in order to access this page; for instance, '/index.html'.

Environment variables

These variables are imported into PHP's global namespace from the environment under which the PHP parser is running. Many are provided by the shell under which PHP is running and different systems are likely running different kinds of shells, a definitive list is impossible. Please see your shell's documentation for a list of defined environment variables.

Other environment variables include the CGI variables, placed there regardless of whether PHP is running as a server module or CGI processor.

PHP variables

These variables are created by PHP itself. The `$HTTP_*_VARS` variables are available only if the track_vars configuration is turned on. When enabled, the variables are always set, even if they are empty arrays. This prevents a malicious user from spoofing these variables.

Nota: As of PHP 4.0.3, track_vars is always turned on, regardless of the configuration file setting.

If the register_globals directive is set, then these variables will also be made available in the global scope of the script; i.e., separate from the `$HTTP_*_VARS` arrays. This feature should be used with care, and turned off if possible; while the `$HTTP_*_VARS` variables are safe, the bare global equivalents can be overwritten by user input, with possibly malicious intent. If you cannot turn off register_globals, you must take whatever steps are necessary to ensure that the data you are using is safe.

$argv

Array of arguments passed to the script. When the script is run on the command line, this gives C-style access to the command line parameters. When called via the GET method, this will contain the query string.

$argc

Contains the number of command line parameters passed to the script (if run on the command line).

$PHP_SELF

The filename of the currently executing script, relative to the document root. If PHP is running as a command-line processor, this variable is not available.

$HTTP_COOKIE_VARS

An associative array of variables passed to the current script via HTTP cookies.

$HTTP_GET_VARS

An associative array of variables passed to the current script via the HTTP GET method.

$HTTP_POST_VARS

An associative array of variables passed to the current script via the HTTP POST method.

$HTTP_POST_FILES

An associative array of variables containing information about files uploaded via the HTTP POST method. See POST method uploads for information on the contents of `$HTTP_POST_FILES`.

`$HTTP_POST_FILES` is available only in PHP 4.0.0 and later.

$HTTP_ENV_VARS

An associative array of variables passed to the current script via the parent environment.

$HTTP_SERVER_VARS

An associative array of variables passed to the current script from the HTTP server. These variables are analogous to the Apache variables described above.

Variable scope

The scope of a variable is the context within which it is defined. For the most part all PHP variables only have a single scope. This single scope spans included and required files as well. For example:

```
$a = 1;
include "b.inc";
```

Here the `$a` variable will be available within the included `b.inc` script. However, within user-defined functions a local function scope is introduced. Any variable used inside a function is by default limited to the local function scope. For example:

```
$a = 1; /* global scope */
```

```
function Test()
{
    echo $a; /* reference to local scope variable */
}

Test();
```

This script will not produce any output because the echo statement refers to a local version of the `$a` variable, and it has not been assigned a value within this scope. You may notice that this is a little bit different from the C language in that global variables in C are automatically available to functions unless specifically overridden by a local definition. This can cause some problems in that people may inadvertently change a global variable. In PHP global variables must be declared global inside a function if they are going to be used in that function. An example:

```
$a = 1;
$b = 2;

function Sum()
{
    global $a, $b;

    $b = $a + $b;
}

Sum();
echo $b;
```

The above script will output "3". By declaring `$a` and `$b` global within the function, all references to either variable will refer to the global version. There is no limit to the number of global variables that can be manipulated by a function.

A second way to access variables from the global scope is to use the special PHP-defined `$GLOBALS` array. The previous example can be rewritten as:

```
$a = 1;
$b = 2;

function Sum()
{
    $GLOBALS["b"] = $GLOBALS["a"] + $GLOBALS["b"];
}

Sum();
echo $b;
```

The `$GLOBALS` array is an associative array with the name of the global variable being the key and the contents of that variable being the value of the array element.

Another important feature of variable scoping is the *static* variable. A static variable exists only in a local function scope, but it does not lose its value when program execution leaves this scope. Consider the following example:

```
function Test ()
{
    $a = 0;
    echo $a;
    $a++;
}
```

This function is quite useless since every time it is called it sets `$a` to `0` and prints "0". The `$a++` which increments the variable serves no purpose since as soon as the function exits the `$a` variable disappears. To make a useful counting function which will not lose track of the current count, the `$a` variable is declared static:

```
function Test()
{
    static $a = 0;
    echo $a;
    $a++;
}
```

Now, every time the Test() function is called it will print the value of `$a` and increment it.

Static variables also provide one way to deal with recursive functions. A recursive function is one which calls itself. Care must be taken when writing a recursive function because it is possible to make it recurse indefinitely. You must make sure you have an adequate way of terminating the recursion. The following simple function recursively counts to 10, using the static variable `$count` to know when to stop:

```
function Test()
{
    static $count = 0;

    $count++;
    echo $count;
    if ($count < 10) {
        Test ();
    }
    $count--;
}
```

Variable variables

Sometimes it is convenient to be able to have variable variable names. That is, a variable name which can be set and used dynamically. A normal variable is set with a statement such as:

```
$a = "hello";
```

A variable variable takes the value of a variable and treats that as the name of a variable. In the above example, *hello*, can be used as the name of a variable by using two dollar signs. i.e.

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

At this point two variables have been defined and stored in the PHP symbol tree: `$a` with contents "hello" and `$hello` with contents "world". Therefore, this statement:

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

produces the exact same output as:

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

i.e. they both produce: `hello world`.

In order to use variable variables with arrays, you have to resolve an ambiguity problem. That is, if you write `$$a[1]` then the parser needs to know if you meant to use `$a[1]` as a variable, or if you wanted `$$a` as the variable and then the [1] index from that variable. The syntax for resolving this ambiguity is: `${$a[1]}` for the first case and `${$a}[1]` for the second.

Variables from outside PHP

HTML Forms (GET and POST)

When a form is submitted to a PHP script, any variables from that form will be automatically made available to the script by PHP. If the track_vars configuration option is turned on, then these variables will be located in the associative arrays `$HTTP_POST_VARS`, `$HTTP_GET_VARS`, and/or `$HTTP_POST_FILES`, according to the source of the variable in question.

For more information on these variables, please read Predefined variables.

Esempio 7-1. Simple form variable

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

When the above form is submitted, the value from the text input will be available in `$HTTP_POST_VARS['username']`. If the register_globals configuration directive is turned on, then the variable will also be available as `$username` in the global scope.

PHP also understands arrays in the context of form variables. You may, for example, group related variables together, or use this feature to retrieve values from a multiple select input:

Esempio 7-2. More complex form variables

```
<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>
```

In PHP 3, the array form variable usage is limited to single-dimensional arrays. In PHP 4, no such restriction applies.

IMAGE SUBMIT variable names

When submitting a form, it is possible to use an image instead of the standard submit button with a tag like:

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

When the user clicks somewhere on the image, the accompanying form will be transmitted to the server with two additional variables, sub_x and sub_y. These contain the coordinates of the user click within the image. The experienced may note that the actual variable names sent by the browser contains a period rather than an underscore, but PHP converts the period to an underscore automatically.

HTTP Cookies

PHP transparently supports HTTP cookies as defined by Netscape's Spec (http://www.netscape.com/newsref/std/cookie_spec.html). Cookies are a mechanism for storing data in the remote browser and thus tracking or identifying return users. You can set cookies using the setcookie() function. Cookies are part of the HTTP header, so the SetCookie function must be called before any output is sent to the browser. This is the same restriction as for the header() function. Any cookies sent to you from the client will automatically be turned into a PHP variable just like GET and POST method data.

If you wish to assign multiple values to a single cookie, just add *[]* to the cookie name. For example:

```
setcookie("MyCookie[]", "Testing", time()+3600);
```

Note that a cookie will replace a previous cookie by the same name in your browser unless the path or domain is different. So, for a shopping cart application you may want to keep a counter and pass this along. i.e.

Esempio 7-3. SetCookie Example

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

Environment variables

PHP automatically makes environment variables available as normal PHP variables.

```
echo $HOME;  /* Shows the HOME environment variable, if set. */
```

Since information coming in via GET, POST and Cookie mechanisms also automatically create PHP variables, it is sometimes best to explicitly read a variable from the environment in order to make sure that you are getting the right version. The getenv() function can be used for this. You can also set an environment variable with the putenv() function.

Dots in incoming variable names

Typically, PHP does not alter the names of variables when they are passed into a script. However, it should be noted that the dot (period, full stop) is not a valid character in a PHP variable name. For the reason, look at it:

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

Now, what the parser sees is a variable named `$varname`, followed by the string concatenation operator, followed by the barestring (i.e. unquoted string which doesn't match any known key or reserved words) 'ext'. Obviously, this doesn't have the intended result.

For this reason, it is important to note that PHP will automatically replace any dots in incoming variable names with underscores.

Determining variable types

Because PHP determines the types of variables and converts them (generally) as needed, it is not always obvious what type a given variable is at any one time. PHP includes several functions which find out what type a variable is. They are gettype(), is_long(), is_double(), is_string(), is_array(), and is_object().

Capitolo 8. Constants

A constant is a identifier (name) for a simple value. As the name suggests, that value cannot change during the execution of the script (the magic constants `__FILE__` and `__LINE__` are the only exception). A constant is case-sensitive by default. By convention constants are always uppercase.

The name of a constant follows the same rules as any label in PHP. A valid constant name starts with a letter or underscore, followed by any number of letters, numbers, or underscores. As a regular expression, it would be expressed thus: `[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*`

> **Nota:** For our purposes here, a letter is a-z, A-Z, and the ASCII characters from 127 through 255 (0x7f-0xff).

The scope of a constant is global.

Syntax

You can define a constant by using the define()-function. Once a constant is defined, it can never be changed or undefined.

Only scalar data (boolean, integer, double and string) can be contained in constants.

You can get the value of a constant by simply specifying its name. Unlike with variables, you should *not* prepend a constant with a $. You can also use the function constant(), to read a constant's value, if you are to obtain the constant's name dynamically. Use get_defined_constants() to get a list of all defined constants.

> **Nota:** Constants and (global) variables are in a different namespace. This implies that for example `TRUE` and `$TRUE` are generally different.

If you use an undefined constant, PHP assumes that you mean the name of the constant itself. A notice will be issued when this happens. Use the defined()-function if you want to know if a constant is set.

This are the differences with variables:

- Constants do not have a dollar sign ($) before them;
- Constants may be defined and accessed anywhere without regard to variable scoping rules;
- Constants may not be redefined or undefined once they have been set; and
- Constants may only evaluate to scalar values.

Esempio 8-1. Defining Constants

```
<?php
define("CONSTANT", "Hello world.");
echo CONSTANT; // outputs "Hello world."
echo Constant; // outputs "Constant" and issues a notice.
?>
```

Predefined constants

The predefined constants (always available) are:

__FILE__ (case-insensitive)

The name of the script file presently being parsed. If used within a file which has been included or required, then the name of the included file is given, and not the name of the parent file.

__LINE__ (case-insensitive)

The number of the line within the current script file which is being parsed. If used within a file which has been included or required, then the position within the included file is given.

PHP_VERSION

The string representation of the version of the PHP parser presently in use; e.g. ’4.0.7-dev’.

PHP_OS

The name of the operating system on which the PHP parser is executing;. Possible values may be : "AIX", "Darwin" (MacOS), "Linux", "SunOS", "WIN32", "WINNT". Note: other values may be available too.

`TRUE` (case-insensitive)

A `TRUE` value (see the boolean type).

`FALSE` (case-insensitive)

A `FALSE` value (see the boolean type).

`NULL` (case-insensitive)

A `NULL` value (see the null type).

E_ERROR

Denotes an error other than a parsing error from which recovery is not possible.

E_WARNING

Denotes a condition where PHP knows something is wrong, but will continue anyway; these can be caught by the script itself. An example would be an invalid regexp in ereg().

E_PARSE

The parser choked on invalid syntax in the script file. Recovery is not possible.

E_NOTICE

Something happened which may or may not be an error. Execution continues. Examples include using an unquoted string as an array index, or accessing a variable which has not been set.

E_ALL

All of the E_* constants rolled into one. If used with error_reporting(), will cause any and all problems noticed by PHP to be reported.

The `E_*` constants are typically used with the error_reporting() function for setting the error reporting level. See all these constants at Error handling.

Esempio 8-2. Using __FILE__ and __LINE__

```
<?php
function report_error($file, $line, $message)
{
    echo "An error occured in $file on line $line: $message.";
}

report_error(__FILE__, __LINE__, "Something went wrong!");
?>
```

Capitolo 9. Expressions

Expressions are the most important building stones of PHP. In PHP, almost anything you write is an expression. The simplest yet most accurate way to define an expression is "anything that has a value".

The most basic forms of expressions are constants and variables. When you type "$a = 5", you're assigning '5' into $a. '5', obviously, has the value 5, or in other words '5' is an expression with the value of 5 (in this case, '5' is an integer constant).

After this assignment, you'd expect $a's value to be 5 as well, so if you wrote $b = $a, you'd expect it to behave just as if you wrote $b = 5. In other words, $a is an expression with the value of 5 as well. If everything works right, this is exactly what will happen.

Slightly more complex examples for expressions are functions. For instance, consider the following function:

```
function foo ()
{
    return 5;
}
```

Assuming you're familiar with the concept of functions (if you're not, take a look at the chapter about functions), you'd assume that typing `$c = foo()` is essentially just like writing `$c = 5`, and you're right. Functions are expressions with the value of their return value. Since foo() returns 5, the value of the expression 'foo()' is 5. Usually functions don't just return a static value but compute something.

Of course, values in PHP don't have to be integers, and very often they aren't. PHP supports three scalar value types: integer values, floating point values and string values (scalar values are values that you can't 'break' into smaller pieces, unlike arrays, for instance). PHP also supports two composite (non-scalar) types: arrays and objects. Each of these value types can be assigned into variables or returned from functions.

So far, users of PHP/FI 2 shouldn't feel any change. However, PHP takes expressions much further, in the same way many other languages do. PHP is an expression-oriented language, in the sense that almost everything is an expression. Consider the example we've already dealt with, '$a = 5'. It's easy to see that there are two values involved here, the value of the integer constant '5', and the value of $a which is being updated to 5 as well. But the truth is that there's one additional value involved here, and that's the value of the assignment itself. The assignment itself evaluates to the assigned value, in this case 5. In practice, it means that '$a = 5', regardless of what it does, is an expression with the value 5. Thus, writing something like '$b = ($a = 5)' is like writing '$a = 5; $b = 5;' (a semicolon marks the end of a statement). Since assignments are parsed in a right to left order, you can also write '$b = $a = 5'.

Another good example of expression orientation is pre- and post-increment and decrement. Users of PHP/FI 2 and many other languages may be familiar with the notation of variable++ and variable--.

These are increment and decrement operators. In PHP/FI 2, the statement ’$a++’ has no value (is not an expression), and thus you can’t assign it or use it in any way. PHP enhances the increment/decrement capabilities by making these expressions as well, like in C. In PHP, like in C, there are two types of increment - pre-increment and post-increment. Both pre-increment and post-increment essentially increment the variable, and the effect on the variable is idential. The difference is with the value of the increment expression. Pre-increment, which is written ’++$variable’, evaluates to the incremented value (PHP increments the variable before reading its value, thus the name ’pre-increment’). Post-increment, which is written ’$variable++’ evaluates to the original value of $variable, before it was incremented (PHP increments the variable after reading its value, thus the name ’post-increment’).

A very common type of expressions are comparison expressions. These expressions evaluate to either 0 or 1, meaning `FALSE` or `TRUE` (respectively). PHP supports > (bigger than), >= (bigger than or equal to), == (equal), != (not equal), < (smaller than) and <= (smaller than or equal to). These expressions are most commonly used inside conditional execution, such as `if` statements.

The last example of expressions we’ll deal with here is combined operator-assignment expressions. You already know that if you want to increment $a by 1, you can simply write ’$a++’ or ’++$a’. But what if you want to add more than one to it, for instance 3? You could write ’$a++’ multiple times, but this is obviously not a very efficient or comfortable way. A much more common practice is to write ’$a = $a + 3’. ’$a + 3’ evaluates to the value of $a plus 3, and is assigned back into $a, which results in incrementing $a by 3. In PHP, as in several other languages like C, you can write this in a shorter way, which with time would become clearer and quicker to understand as well. Adding 3 to the current value of $a can be written ’$a += 3’. This means exactly "take the value of $a, add 3 to it, and assign it back into $a". In addition to being shorter and clearer, this also results in faster execution. The value of ’$a += 3’, like the value of a regular assignment, is the assigned value. Notice that it is NOT 3, but the combined value of $a plus 3 (this is the value that’s assigned into $a). Any two-place operator can be used in this operator-assignment mode, for example ’$a -= 5’ (subtract 5 from the value of $a), ’$b *= 7’ (multiply the value of $b by 7), etc.

There is one more expression that may seem odd if you haven’t seen it in other languages, the ternary conditional operator:

```
$first ? $second : $third
```

If the value of the first subexpression is `TRUE` (non-zero), then the second subexpression is evaluated, and that is the result of the conditional expression. Otherwise, the third subexpression is evaluated, and that is the value.

The following example should help you understand pre- and post-increment and expressions in general a bit better:

```
function double($i)
{
    return $i*2;
}
```

```
$b = $a = 5;        /* assign the value five into the vari-
able $a and $b */
$c = $a++;          /* post-increment, assign original value of $a
                       (5) to $c */
$e = $d = ++$b;     /* pre-increment, assign the incremented value of
                       $b (6) to $d and $e */

/* at this point, both $d and $e are equal to 6 */

$f = double($d++);  /* assign twice the value of $d before
                       the increment, 2*6 = 12 to $f */
$g = double(++$e);  /* assign twice the value of $e after
                       the increment, 2*7 = 14 to $g */
$h = $g += 10;      /* first, $g is incremented by 10 and ends with the
                       value of 24. the value of the assignment (24) is
                       then assigned into $h, and $h ends with the value
                       of 24 as well. */
```

In the beginning of the chapter we said that we'll be describing the various statement types, and as promised, expressions can be statements. However, not every expression is a statement. In this case, a statement has the form of 'expr' ';' that is, an expression followed by a semicolon. In '$b=$a=5;', $a=5 is a valid expression, but it's not a statement by itself. '$b=$a=5;' however is a valid statement.

One last thing worth mentioning is the truth value of expressions. In many events, mainly in conditional execution and loops, you're not interested in the specific value of the expression, but only care about whether it means `TRUE` or `FALSE`. The constants `TRUE` and `FALSE` (case-insensitive) are the two possible boolean values. When necessary, an expression is autmatically converted to boolean. See the section about type-casting for details about how.

PHP provides a full and powerful implementation of expressions, and documenting it entirely goes beyond the scope of this manual. The above examples should give you a good idea about what expressions are and how you can construct useful expressions. Throughout the rest of this manual we'll write *`expr`* to indicate any valid PHP expression.

Capitolo 10. Operators

Arithmetic Operators

Remember basic arithmetic from school? These work just like those.

Tabella 10-1. Arithmetic Operators

Example	Name	Result

The division operator ("/") returns an integer value (the result of an integer division) if the two operands are integers (or strings that get converted to integers) and the quotient is an integer. If either operand is a floating-point value, or the operation results in a non-integer value, a floating-point value is returned.

Assignment Operators

The basic assignment operator is "=". Your first inclination might be to think of this as "equal to". Don’t. It really means that the the left operand gets set to the value of the expression on the rights (that is, "gets set to").

The value of an assignment expression is the value assigned. That is, the value of "$a = 3" is 3. This allows you to do some tricky things:

```
$a = ($b = 4) + 5; // $a is equal to 9 now, and $b has been set to 4.
```

In addition to the basic assignment operator, there are "combined operators" for all of the binary arithmetic and string operators that allow you to use a value in an expression and then set its value to the result of that expression. For example:

```
$a = 3;
$a += 5; // sets $a to 8, as if we had said: $a = $a + 5;
$b = "Hello ";
$b .= "There!"; // sets $b to "Hello There!", just like $b = $b . "There!";
```

Note that the assignment copies the original variable to the new one (assignment by value), so changes to one will not affect the other. This may also have relevance if you need to copy something like a large array inside a tight loop. PHP 4 supports assignment by reference, using the `$var = &$othervar;` syntax, but this is not possible in PHP 3. ’Assignment by reference’ means that both

variables end up pointing at the same data, and nothing is copied anywhere. To learn more about references, please read References explained.

Bitwise Operators

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

Tabella 10-2. Bitwise Operators

Example	Name	Result

Comparison Operators

Comparison operators, as their name implies, allow you to compare two values.

Tabella 10-3. Comparison Operators

Example	Name	Result

Another conditional operator is the "?:" (or ternary) operator, which operates as in C and many other languages.

```
(expr1) ? (expr2) : (expr3);
```

This expression evaluates to *`expr2`* if *`expr1`* evaluates to `TRUE`, and *`expr3`* if *`expr1`* evaluates to `FALSE`.

Error Control Operators

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 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 file error */
$my_file = @file (’non_existent_file’) or
    die ("Failed opening file: error was ’$php_errormsg’");
```

```
// this works for any expression, not just functions:
$value = @$cache[$key];
// will not issue a notice if the index $key doesn’t exist.

?>
```

> **Nota:** The @-operator works only on expressions. A simple rule of thumb is: if you can take the value of something, you can prepend the @ operator to it. For instance, you can prepend it to variables, function and include() calls, constants, and so forth. You cannot prepend it to function or class definitions, or conditional structures such as `if` and `foreach`, and so forth.

See also error_reporting().

> **Attenzione**
>
> Currently the "@" error-control operator prefix will even disable error reporting for critical errors that will terminate script execution. Among other things, this means that if you use "@" to suppress errors from a certain function and either it isn’t available or has been mistyped, the script will die right there with no indication as to why.

Execution Operators

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>";
```

> **Nota:** The backtick operator is disabled when safe mode is enabled.

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

Incrementing/Decrementing Operators

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

Tabella 10-4. Increment/decrement Operators

Example	Name	Effect

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";
?>
```

Logical Operators

Tabella 10-5. Logical Operators

Example	Name	Result

The reason for the two different variations of "and" and "or" operators is that they operate at different precedences. (See Operator Precedence.)

Operator Precedence

The precedence of an operator specifies how "tightly" it binds two expressions together. For example, in the expression `1 + 5 * 3`, the answer is `16` and not `18` because the multiplication ("*") operator has a higher precedence than the addition ("+") operator. Parentheses may be used to force precedence, if necessary. For instance: `(1 + 5) * 3` evaluates to `18`.

The following table lists the precedence of operators with the lowest-precedence operators listed first.

Tabella 10-6. Operator Precedence

Associativity	Operators

String Operators

There are two string operators. The first is the concatenation operator ('.'), which returns the concatenation of its right and left arguments. The second is the concatenating assignment operator ('.='), which appends the argument on the right side to the argument on the left side. Please read Assignment Operators for more information.

```
$a = "Hello ";
$b = $a . "World!"; // now $b contains "Hello World!"

$a = "Hello ";
$a .= "World!";     // now $a contains "Hello World!"
```

Capitolo 11. Control Structures

Any PHP script is built out of a series of statements. A statement can be an assignment, a function call, a loop, a conditional statement of even a statement that does nothing (an empty statement). Statements usually end with a semicolon. In addition, statements can be grouped into a statement-group by encapsulating a group of statements with curly braces. A statement-group is a statement by itself as well. The various statement types are described in this chapter.

`if`

The `if` construct is one of the most important features of many languages, PHP included. It allows for conditional execution of code fragments. PHP features an `if` structure that is similar to that of C:

```
if (expr)
    statement
```

As described in the section about expressions, expr is evaluated to its truth value. If *`expr`* evaluates to `TRUE`, PHP will execute statement, and if it evaluates to `FALSE` - it'll ignore it.

The following example would display `a is bigger than b` if *`$a`* is bigger than *`$b`*:

```
if ($a > $b)
    print "a is bigger than b";
```

Often you'd want to have more than one statement to be executed conditionally. Of course, there's no need to wrap each statement with an `if` clause. Instead, you can group several statements into a statement group. For example, this code would display `a is bigger than b` if *`$a`* is bigger than *`$b`*, and would then assign the value of *`$a`* into *`$b`*:

```
if ($a > $b) {
    print "a is bigger than b";
    $b = $a;
}
```

If statements can be nested indefinitely within other `if` statements, which provides you with complete flexibility for conditional execution of the various parts of your program.

`else`

Often you'd want to execute a statement if a certain condition is met, and a different statement if the condition is not met. This is what `else` is for. `else` extends an `if` statement to execute a statement in case the expression in the `if` statement evaluates to `FALSE`. For example, the following code would display `a is bigger than b` if *$a* is bigger than *$b*, and `a is NOT bigger than b` otherwise:

```
if ($a > $b) {
    print "a is bigger than b";
} else {
    print "a is NOT bigger than b";
}
```

The `else` statement is only executed if the `if` expression evaluated to `FALSE`, and if there were any `elseif` expressions - only if they evaluated to `FALSE` as well (see elseif).

`elseif`

`elseif`, as its name suggests, is a combination of `if` and `else`. Like `else`, it extends an `if` statement to execute a different statement in case the original `if` expression evaluates to `FALSE`. However, unlike `else`, it will execute that alternative expression only if the `elseif` conditional expression evaluates to `TRUE`. For example, the following code would display `a is bigger than b`, `a equal to b` or `a is smaller than b`:

```
if ($a > $b) {
    print "a is bigger than b";
} elseif ($a == $b) {
    print "a is equal to b";
} else {
    print "a is smaller than b";
}
```

There may be several `elseif`s within the same `if` statement. The first `elseif` expression (if any) that evaluates to `TRUE` would be executed. In PHP, you can also write 'else if' (in two words) and the behavior would be identical to the one of 'elseif' (in a single word). The syntactic meaning is slightly different (if you're familiar with C, this is the same behavior) but the bottom line is that both would result in exactly the same behavior.

The `elseif` statement is only executed if the preceding `if` expression and any preceding `elseif` expressions evaluated to `FALSE`, and the current `elseif` expression evaluated to `TRUE`.

Alternative syntax for control structures

> **Attenzione**
>
> Alternative syntax is deprecated as of PHP 4. Basically, it just generates unreadable code, and it gets very complicated when mixing it with the normal syntax. Although there are no plans to break this syntax, it cannot be ruled out that one day this will stop working. You are warned.

PHP offers an alternative syntax for some of its control structures; namely, `if`, `while`, `for`, `foreach`, and `switch`. In each case, the basic form of the alternate syntax is to change the opening brace to a colon (:) and the closing brace to `endif;`, `endwhile;`, `endfor;`, `endforeach;`, or `endswitch;`, respectively.

```
<?php if ($a == 5): ?>
A is equal to 5
<?php endif; ?>
```

In the above example, the HTML block "A = 5" is nested within an `if` statement written in the alternative syntax. The HTML block would be displayed only if $a is equal to 5.

The alternative syntax applies to `else` and `elseif` as well. The following is an `if` structure with `elseif` and `else` in the alternative format:

```
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;
```

See also while, for, and if for further examples.

`while`

`while` loops are the simplest type of loop in PHP. They behave just like their C counterparts. The basic form of a `while` statement is:

```
while (expr) statement
```

The meaning of a `while` statement is simple. It tells PHP to execute the nested statement(s) repeatedly, as long as the `while` expression evaluates to `TRUE`. The value of the expression is checked each time at the beginning of the loop, so even if this value changes during the execution of the nested statement(s), execution will not stop until the end of the iteration (each time PHP runs the statements in the loop is one iteration). Sometimes, if the `while` expression evaluates to `FALSE` from the very beginning, the nested statement(s) won't even be run once.

Like with the `if` statement, you can group multiple statements within the same `while` loop by surrounding a group of statements with curly braces, or by using the alternate syntax:

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

The following examples are identical, and both print numbers from 1 to 10:

```
/* example 1 */

$i = 1;
while ($i <= 10) {
    print $i++;  /* the printed value would be
                    $i before the increment
                    (post-increment) */
}

/* example 2 */

$i = 1;
while ($i <= 10):
    print $i;
    $i++;
endwhile;
```

do..while

`do..while` loops are very similar to `while` loops, except the truth expression is checked at the end of each iteration instead of in the beginning. The main difference from regular `while` loops is that the first iteration of a `do..while` loop is guarenteed to run (the truth expression is only checked at the end of the iteration), whereas it's may not necessarily run with a regular `while` loop (the truth expression is checked at the beginning of each iteration, if it evaluates to `FALSE` right from the beginning, the loop execution would end immediately).

There is just one syntax for `do..while` loops:

```
$i = 0;
```

```
do {
   print $i;
} while ($i>0);
```

The above loop would run one time exactly, since after the first iteration, when truth expression is checked, it evaluates to FALSE ($i is not bigger than 0) and the loop execution ends.

Advanced C users may be familiar with a different usage of the do..while loop, to allow stopping execution in the middle of code blocks, by encapsulating them with do..while(0), and using the break statement. The following code fragment demonstrates this:

```
do {
    if ($i < 5) {
        print "i is not big enough";
        break;
    }
    $i *= $factor;
    if ($i < $minimum_limit) {
        break;
    }
    print "i is ok";

     ...process i...

} while(0);
```

Don’t worry if you don’t understand this right away or at all. You can code scripts and even powerful scripts without using this ‘feature’.

for

for loops are the most complex loops in PHP. They behave like their C counterparts. The syntax of a for loop is:

```
for (expr1; expr2; expr3) statement
```

The first expression (*expr1*) is evaluated (executed) once unconditionally at the beginning of the loop.

In the beginning of each iteration, *expr2* is evaluated. If it evaluates to TRUE, the loop continues and the nested statement(s) are executed. If it evaluates to FALSE, the execution of the loop ends.

At the end of each iteration, *expr3* is evaluated (executed).

Each of the expressions can be empty. *`expr2`* being empty means the loop should be run indefinitely (PHP implicitly considers it as `TRUE`, like C). This may not be as useless as you might think, since often you'd want to end the loop using a conditional `break` statement instead of using the `for` truth expression.

Consider the following examples. All of them display numbers from 1 to 10:

```
/* example 1 */

for ($i = 1; $i <= 10; $i++) {
    print $i;
}

/* example 2 */

for ($i = 1;;$i++) {
    if ($i > 10) {
        break;
    }
    print $i;
}

/* example 3 */

$i = 1;
for (;;) {
    if ($i > 10) {
        break;
    }
    print $i;
    $i++;
}

/* example 4 */

for ($i = 1; $i <= 10; print $i, $i++);
```

Of course, the first example appears to be the nicest one (or perhaps the fourth), but you may find that being able to use empty expressions in `for` loops comes in handy in many occasions.

PHP also supports the alternate "colon syntax" for `for` loops.

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

Other languages have a `foreach` statement to traverse an array or hash. PHP 3 has no such construct; PHP 4 does (see foreach). In PHP 3, you can combine while with the list() and each() functions to achieve the same effect. See the documentation for these functions for an example.

`foreach`

PHP 4 (not PHP 3) includes a `foreach` construct, much like Perl and some other languages. This simply gives an easy way to iterate over arrays. There are two syntaxes; the second is a minor but useful extension of the first:

```
foreach(array_expression as $value) statement
foreach(array_expression as $key => $value) statement
```

The first form loops over the array given by `array_expression`. On each loop, the value of the current element is assigned to `$value` and the internal array pointer is advanced by one (so on the next loop, you'll be looking at the next element).

The second form does the same thing, except that the current element's key will be assigned to the variable `$key` on each loop.

> **Nota:** When `foreach` first starts executing, the internal array pointer is automatically reset to the first element of the array. This means that you do not need to call reset() before a `foreach` loop.

> **Nota:** Also note that `foreach` operates on a copy of the specified array, not the array itself, therefore the array pointer is not modified as with the each() construct and changes to the array element returned are not reflected in the original array.

> **Nota:** `foreach` does not support the ability to suppress error messages using '@'.

You may have noticed that the following are functionally identical:

```
reset ($arr);
while (list(, $value) = each ($arr)) {
    echo "Value: $value<br>\n";
}

foreach ($arr as $value) {
    echo "Value: $value<br>\n";
}
```

The following are also functionally identical:

```
reset ($arr);
while (list($key, $value) = each ($arr)) {
    echo "Key: $key; Value: $value<br>\n";
}

foreach ($arr as $key => $value) {
    echo "Key: $key; Value: $value<br>\n";
}
```

Some more examples to demonstrate usages:

```
/* foreach example 1: value only */

$a = array (1, 2, 3, 17);

foreach ($a as $v) {
   print "Current value of \$a: $v.\n";
}

/* foreach example 2: value (with key printed for illustration) */

$a = array (1, 2, 3, 17);

$i = 0; /* for illustrative purposes only */

foreach($a as $v) {
    print "\$a[$i] => $v.\n";
}

/* foreach example 3: key and value */

$a = array (
    "one" => 1,
    "two" => 2,
    "three" => 3,
    "seventeen" => 17
);

foreach($a as $k => $v) {
    print "\$a[$k] => $v.\n";
}

/* foreach example 4: multi-dimensional arrays */

$a[0][0] = "a";
```

```
$a[0][1] = "b";
$a[1][0] = "y";
$a[1][1] = "z";

foreach($a as $v1) {
    foreach ($v1 as $v2) {
        print "$v2\n";
    }
}

/* foreach example 5: dynamic arrays */

foreach(array(1, 2, 3, 4, 5) as $v) {
    print "$v\n";
}
```

`break`

`break` ends execution of the current `for`, `foreach` `while`, `do..while` or `switch` structure.

`break` accepts an optional numeric argument which tells it how many nested enclosing structures are to be broken out of.

```
$arr = array (’one’, ’two’, ’three’, ’four’, ’stop’, ’five’);
while (list (, $val) = each ($arr)) {
    if ($val == ’stop’) {
        break;    /* You could also write ’break 1;’ here. */
    }
    echo "$val<br>\n";
}

/* Using the optional argument. */

$i = 0;
while (++$i) {
    switch ($i) {
    case 5:
        echo "At 5<br>\n";
        break 1;  /* Exit only the switch. */
    case 10:
        echo "At 10; quitting<br>\n";
        break 2;  /* Exit the switch and the while. */
    default:
        break;
    }
```

```
}
```

continue

`continue` is used within looping structures to skip the rest of the current loop iteration and continue execution at the beginning of the next iteration.

`continue` accepts an optional numeric argument which tells it how many levels of enclosing loops it should skip to the end of.

```
while (list ($key, $value) = each ($arr)) {
    if (!($key % 2)) { // skip odd members
        continue;
    }
    do_something_odd ($value);
}

$i = 0;
while ($i++ < 5) {
    echo "Outer<br>\n";
    while (1) {
        echo "  Middle<br>\n";
        while (1) {
            echo "  Inner<br>\n";
            continue 3;
        }
        echo "This never gets output.<br>\n";
    }
    echo "Neither does this.<br>\n";
}
```

switch

The `switch` statement is similar to a series of IF statements on the same expression. In many occasions, you may want to compare the same variable (or expression) with many different values, and execute a different piece of code depending on which value it equals to. This is exactly what the `switch` statement is for.

The following two examples are two different ways to write the same thing, one using a series of `if` statements, and the other using the `switch` statement:

```
if ($i == 0) {
```

```
    print "i equals 0";
}
if ($i == 1) {
    print "i equals 1";
}
if ($i == 2) {
    print "i equals 2";
}

switch ($i) {
    case 0:
        print "i equals 0";
        break;
    case 1:
        print "i equals 1";
        break;
    case 2:
        print "i equals 2";
        break;
}
```

It is important to understand how the `switch` statement is executed in order to avoid mistakes. The `switch` statement executes line by line (actually, statement by statement). In the beginning, no code is executed. Only when a `case` statement is found with a value that matches the value of the `switch` expression does PHP begin to execute the statements. PHP continues to execute the statements until the end of the `switch` block, or the first time it sees a `break` statement. If you don't write a `break` statement at the end of a case's statement list, PHP will go on executing the statements of the following case. For example:

```
switch ($i) {
    case 0:
        print "i equals 0";
    case 1:
        print "i equals 1";
    case 2:
        print "i equals 2";
}
```

Here, if $i equals to 0, PHP would execute all of the print statements! If $i equals to 1, PHP would execute the last two print statements, and only if $i equals to 2, you'd get the 'expected' behavior and only 'i equals 2' would be displayed. So, it's important not to forget `break` statements (even though you may want to avoid supplying them on purpose under certain circumstances).

In a `switch` statement, the condition is evaluated only once and the result is compared to each `case` statement. In an `elseif` statement, the condition is evaluated again. If your condition is more complicated than a simple compare and/or is in a tight loop, a `switch` may be faster.

The statement list for a case can also be empty, which simply passes control into the statement list for the next case.

```
switch ($i) {
    case 0:
    case 1:
    case 2:
        print "i is less than 3 but not negative";
        break;
    case 3:
        print "i is 3";
}
```

A special case is the default case. This case matches anything that wasn't matched by the other cases, and should be the last `case` statement. For example:

```
switch ($i) {
    case 0:
        print "i equals 0";
        break;
    case 1:
        print "i equals 1";
        break;
    case 2:
        print "i equals 2";
        break;
    default:
        print "i is not equal to 0, 1 or 2";
}
```

The `case` expression may be any expression that evaluates to a simple type, that is, integer or floating-point numbers and strings. Arrays or objects cannot be used here unless they are dereferenced to a simple type.

The alternative syntax for control structures is supported with switches. For more information, see Alternative syntax for control structures .

```
switch ($i):
    case 0:
        print "i equals 0";
        break;
    case 1:
```

```
        print "i equals 1";
        break;
    case 2:
        print "i equals 2";
        break;
    default:
        print "i is not equal to 0, 1 or 2";
endswitch;
```

declare

The `declare` construct is used to set execution directives for a block of code. The syntax of `declare` is similiar to the syntax of other flow control constructs:

```
declare (directive) statement
```

The `directive` section allows the behavior of the `declare` block to be set. Currently only one directive is recognized: the `ticks` directive. (See below for more information on the ticks directive)

The `statement` part of the `declare` block will be executed - how it is executed and what side-effects occur during execution may depend on the directive set in the `directive` block.

Ticks

A tick is an event that occurs for every *`N`* low-level statements executed by the parser within the `declare` block. The value for *`N`* is specified using `ticks=`*`N`* within the `declare` blocks's `directive` section.

The event(s) that occurs on each tick is specified using the **register_tick_function()**. See the example below for more details. Note that more than one event can occur for each tick.

Esempio 11-1. Profile a section of PHP code

```
<pre>
<?php
// A function that records the time when it is called
function profile ($dump = FALSE)
{
    static $profile;

    // Return the times stored in profile, then erase it
    if ($dump) {
        $temp = $profile;
```

```
        unset ($profile);
        return ($temp);
    }

    $profile[] = microtime ();
}

// Set up a tick handler
register_tick_function("profile");

// Initialize the function before the declare block
profile ();

// Run a block of code, throw a tick every 2nd statement
declare (ticks=2) {
    for ($x = 1; $x < 50; ++$x) {
        echo similar_text (md5($x), md5($x*$x)), "<br>";
    }
}

// Display the data stored in the profiler
print_r (profile (TRUE));
?>
</pre>
```

The example profiles the PHP code within the ’declare’ block, recording the time at which every second low-level statement in the block was executed. This information can then be used to find the slow areas within particular segments of code. This process can be performed using other methods: using ticks is more convenient and easier to implement.

Ticks are well suited for debugging, implementing simple multitasking, backgrounded I/O and many other tasks.

See also **register_tick_function()** and **unregister_tick_function()**.

require()

The require() statement replaces itself with the specified file, much like the C preprocessor’s `#include` works.

If "URL fopen wrappers" are enabled in PHP (which they are in the default configuration), you can specify the file to be require()ed using an URL instead of a local pathname. See Remote files and fopen() for more information.

An important note about how this works is that when a file is include()ed or require()ed, parsing drops out of PHP mode and into HTML mode at the beginning of the target file, and resumes PHP

mode again at the end. For this reason, any code inside the target file which should be executed as PHP code must be enclosed within valid PHP start and end tags.

require() is not actually a function in PHP; rather, it is a language construct. It is subject to some different rules than functions are. For instance, require() is not subject to any containing control structures. For another, it does not return any value; attempting to read a return value from a require() call results in a parse error.

Unlike include(), require() will *always* read in the target file, *even if the line it's on never executes*. If you want to conditionally include a file, use include(). The conditional statement won't affect the require(). However, if the line on which the require() occurs is not executed, neither will any of the code in the target file be executed.

Similarly, looping structures do not affect the behaviour of require(). Although the code contained in the target file is still subject to the loop, the require() itself happens only once.

This means that you can't put a require() statement inside of a loop structure and expect it to include the contents of a different file on each iteration. To do that, use an include() statement.

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

When a file is require()ed, the code it contains inherits the variable scope of the line on which the require() occurs. Any variables available at that line in the calling file will be available within the called file. If the require() occurs inside a function within the calling file, then all of the code contained in the called file will behave as though it had been defined inside that function.

If the require()ed file is called via HTTP using the fopen wrappers, and if the target server interprets the target file as PHP code, variables may be passed to the require()ed file using an URL request string as used with HTTP GET. This is not strictly speaking the same thing as require()ing the file and having it inherit the parent file's variable scope; the script is actually being run on the remote server and the result is then being included into the local script.

```
/* This example assumes that someserver is configured to parse .php
 * files and not .txt files. Also, 'works' here means that the variables
 * $varone and $vartwo are available within the require()ed file. */

/* Won't work; file.txt wasn't handled by someserver. */
require ("http://someserver/file.txt?varone=1&vartwo=2");

/* Won't work; looks for a file named 'file.php?varone=1&vartwo=2'
 * on the local filesystem. */
require ("file.php?varone=1&vartwo=2");

/* Works. */
require ("http://someserver/file.php?varone=1&vartwo=2");

$varone = 1;
$vartwo = 2;
```

```
require ("file.txt");  /* Works. */
require ("file.php");  /* Works. */
```

In PHP 3, it is possible to execute a `return` statement inside a require()ed file, as long as that statement occurs in the global scope of the require()ed file. It may not occur within any block (meaning inside braces ({}). In PHP 4, however, this ability has been discontinued. If you need this functionality, see include().

See also include(), require_once(), include_once(), readfile(), and virtual().

include()

The include() statement includes and evaluates the specified file.

If "URL fopen wrappers" are enabled in PHP (which they are in the default configuration), you can specify the file to be include()ed using an URL instead of a local pathname. See Remote files and fopen() for more information.

An important note about how this works is that when a file is include()ed or require()ed, parsing drops out of PHP mode and into HTML mode at the beginning of the target file, and resumes again at the end. For this reason, any code inside the target file which should be executed as PHP code must be enclosed within valid PHP start and end tags.

This happens each time the include() statement is encountered, so you can use an include() statement within a looping structure to include a number of different files.

```
$files = array (’first.inc’, ’second.inc’, ’third.inc’);
for ($i = 0; $i < count($files); $i++) {
    include $files[$i];
}
```

include() differs from require() in that the include statement is re-evaluated each time it is encountered (and only when it is being executed), whereas the require() statement is replaced by the required file when it is first encountered, whether the contents of the file will be evaluated or not (for example, if it is inside an if statement whose condition evaluated to `FALSE`).

Because include() is a special language construct, you must enclose it within a statement block if it is inside a conditional block.

```
/* This is WRONG and will not work as desired. */

if ($condition)
    include($file);
else
    include($other);
```

```
/* This is CORRECT. */

if ($condition) {
    include($file);
} else {
    include($other);
}
```

In both PHP 3 and PHP 4, it is possible to execute a `return` statement inside an include()ed file, in order to terminate processing in that file and return to the script which called it. Some differences in the way this works exist, however. The first is that in PHP 3, the `return` may not appear inside a block unless it's a function block, in which case the `return` applies to that function and not the whole file. In PHP 4, however, this restriction does not exist. Also, PHP 4 allows you to return values from include()ed files. You can take the value of the include() call as you would a normal function. This generates a parse error in PHP 3.

Esempio 11-2. include() in PHP 3 and PHP 4

Assume the existence of the following file (named `test.inc`) in the same directory as the main file:

```
<?php
echo "Before the return <br>\n";
if (1) {
    return 27;
}
echo "After the return <br>\n";
?>
```

Assume that the main file (`main.html`) contains the following:

```
<?php
$retval = include ('test.inc');
echo "File returned: '$retval'<br>\n";
?>
```

When `main.html` is called in PHP 3, it will generate a parse error on line 2; you can't take the value of an include() in PHP 3. In PHP 4, however, the result will be:

```
Before the return
File returned: '27'
```

Now, assume that `main.html` has been altered to contain the following:

```
<?php
include ('test.inc');
echo "Back in main.html<br>\n";
```

```
?>
```

In PHP 4, the output will be:

```
Before the return
Back in main.html
```

However, PHP 3 will give the following output:

```
Before the return
27Back in main.html

Parse error: parse er-
ror in /home/torben/public_html/phptest/main.html on line 5
```

The above parse error is a result of the fact that the `return` statement is enclosed in a non-function block within `test.inc`. When the return is moved outside of the block, the output is:

```
Before the return
27Back in main.html
```

The spurious ’27’ is due to the fact that PHP 3 does not support `return`ing values from files like that.

When a file is include()ed, the code it contains inherits the variable scope of the line on which the include() occurs. Any variables available at that line in the calling file will be available within the called file. If the include() occurs inside a function within the calling file, then all of the code contained in the called file will behave as though it had been defined inside that function.

If the include()ed file is called via HTTP using the fopen wrappers, and if the target server interprets the target file as PHP code, variables may be passed to the include()ed file using an URL request string as used with HTTP GET. This is not strictly speaking the same thing as include()ing the file and having it inherit the parent file’s variable scope; the script is actually being run on the remote server and the result is then being included into the local script.

```
/* This example assumes that someserver is configured to parse .php
 * files and not .txt files. Also, 'works' here means that the variables
 * $varone and $vartwo are available within the include()ed file. */

/* Won't work; file.txt wasn't handled by someserver. */
include ("http://someserver/file.txt?varone=1&vartwo=2");

/* Won't work; looks for a file named 'file.php?varone=1&vartwo=2'
 * on the local filesystem. */
include ("file.php?varone=1&vartwo=2");

/* Works. */
include ("http://someserver/file.php?varone=1&vartwo=2");
```

```
$varone = 1;
$vartwo = 2;
include ("file.txt");  /* Works. */
include ("file.php");  /* Works. */
```

See also require(), require_once(), include_once(), readfile(), and virtual().

require_once()

The require_once() statement replaces itself with the specified file, much like the C preprocessor's `#include` works, and in that respect is similar to the require() statement. The main difference is that in an inclusion chain, the use of require_once() will assure that the code is added to your script only once, and avoid clashes with variable values or function names that can happen.

For example, if you create the following 2 include files `utils.inc` and `foolib.inc`

Esempio 11-3. utils.inc

```
<?php
define("PHPVERSION", floor(phpversion()));
echo "GLOBALS ARE NICE\n";
function goodTea()
{
    return "Oolong tea tastes good!";
}
?>
```

Esempio 11-4. foolib.inc

```
<?php
require ("utils.inc");
function showVar($var)
{
    if (PHPVERSION == 4) {
        print_r($var);
    } else {
        var_dump($var);
    }
}

// bunch of other functions ...
?>
```

And then you write a script `cause_error_require.php`

Esempio 11-5. cause_error_require.php

```
<?php
require("foolib.inc");
/* the following will generate an error */
require("utils.inc");
$foo = array("1",array("complex","quaternion"));
echo "this is requiring utils.inc again which is also\n";
echo "required in foolib.inc\n";
echo "Running goodTea: ".goodTea()."\n";
echo "Printing foo: \n";
showVar($foo);
?>
```

When you try running the latter one, the resulting ouptut will be (using PHP 4.01pl2):

```
GLOBALS ARE NICE
GLOBALS ARE NICE

Fatal error:  Cannot redeclare goodTea() in utils.inc on line 5
```

By modifying `foolib.inc` and `cause_errror_require.php` to use require_once() instead of require() and renaming the last one to `avoid_error_require_once.php`, we have:

Esempio 11-6. foolib.inc (fixed)

```
...
require_once("utils.inc");
function showVar($var)
{
...
```

Esempio 11-7. avoid_error_require_once.php

```
...
require_once("foolib.inc");
require_once("utils.inc");
$foo = array("1",array("complex","quaternion"));
...
```

And when running the latter, the output will be (using PHP 4.0.1pl2):

```
GLOBALS ARE NICE
this is requiring globals.inc again which is also
required in foolib.inc
Running goodTea: Oolong tea tastes good!
Printing foo:
Array
(
    [0] => 1
    [1] => Array
        (
            [0] => complex
            [1] => quaternion
        )

)
```

Also note that, analogous to the behavior of the `#include` of the C preprocessor, this statement acts at "compile time", e.g. when the script is parsed and before it is executed, and should not be used for parts of the script that need to be inserted dynamically during its execution. You should use include_once() or include() for that purpose.

For more examples on using require_once() and include_once(), look at the PEAR code included in the latest PHP source code distributions.

See also: require(), include(), include_once(), get_required_files(), get_included_files(), readfile(), and virtual().

include_once()

The include_once() statement includes and evaluates the specified file during the execution of the script. This is a behavior similar to the include() statement, with the important difference that if the code from a file has already been included, it will not be included again.

As mentioned in the require_once() description, the include_once() should be used in the cases in which the same file might be included and evaluated more than once during a particular execution of a script, and you want to be sure that it is included exactly once to avoid problems with function redefinitions, variable value reassignments, etc.

For more examples on using require_once() and include_once(), look at the PEAR code included in the latest PHP source code distributions.

include_once() was added in PHP 4.0.1pl2

See also: require(), include(), require_once(), get_required_files(), get_included_files(), readfile(), and virtual().

Capitolo 12. Functions

User-defined functions

A function may be defined using syntax such as the following:

```
function foo ($arg_1, $arg_2, ..., $arg_n)
{
    echo "Example function.\n";
    return $retval;
}
```

Any valid PHP code may appear inside a function, even other functions and class definitions.

In PHP 3, functions must be defined before they are referenced. No such requirement exists in PHP 4.

PHP does not support function overloading, nor is it possible to undefine or redefine previously-declared functions.

PHP 3 does not support variable numbers of arguments to functions, although default arguments are supported (see Default argument values for more information). PHP 4 supports both: see Variable-length argument lists and the function references for func_num_args(), func_get_arg(), and func_get_args() for more information.

Function arguments

Information may be passed to functions via the argument list, which is a comma-delimited list of variables and/or constants.

PHP supports passing arguments by value (the default), passing by reference, and default argument values. Variable-length argument lists are supported only in PHP 4 and later; see Variable-length argument lists and the function references for func_num_args(), func_get_arg(), and func_get_args() for more information. A similar effect can be achieved in PHP 3 by passing an array of arguments to a function:

```
function takes_array($input)
{
    echo "$input[0] + $input[1] = ", $input[0]+$input[1];
}
```

Making arguments be passed by reference

By default, function arguments are passed by value (so that if you change the value of the argument within the function, it does not get changed outside of the function). If you wish to allow a function to modify its arguments, you must pass them by reference.

If you want an argument to a function to always be passed by reference, you can prepend an ampersand (&) to the argument name in the function definition:

```
function add_some_extra(&$string)
{
    $string .= 'and something extra.';
}
$str = 'This is a string, ';
add_some_extra($str);
echo $str;    // outputs 'This is a string, and something extra.'
```

If you wish to pass a variable by reference to a function which does not do this by default, you may prepend an ampersand to the argument name in the function call:

```
function foo ($bar)
{
    $bar .= ' and something extra.';
}
$str = 'This is a string, ';
foo ($str);
echo $str;    // outputs 'This is a string, '
foo (&$str);
echo $str;    // outputs 'This is a string, and something extra.'
```

Default argument values

A function may define C++-style default values for scalar arguments as follows:

```
function makecoffee ($type = "cappucino")
{
    return "Making a cup of $type.\n";
}
echo makecoffee ();
echo makecoffee ("espresso");
```

The output from the above snippet is:

```
Making a cup of cappucino.
```

```
Making a cup of espresso.
```

The default value must be a constant expression, not (for example) a variable or class member.

Note that when using default arguments, any defaults should be on the right side of any non-default arguments; otherwise, things will not work as expected. Consider the following code snippet:

```
function makeyogurt ($type = "acidophilus", $flavour)
{
    return "Making a bowl of $type $flavour.\n";
}

echo makeyogurt ("raspberry");   // won't work as expected
```

The output of the above example is:

```
Warning: Missing argument 2 in call to makeyogurt() in
/usr/local/etc/httpd/htdocs/php3test/functest.html on line 41
Making a bowl of raspberry .
```

Now, compare the above with this:

```
function makeyogurt ($flavour, $type = "acidophilus")
{
    return "Making a bowl of $type $flavour.\n";
}

echo makeyogurt ("raspberry");   // works as expected
```

The output of this example is:

```
Making a bowl of acidophilus raspberry.
```

Variable-length argument lists

PHP 4 has support for variable-length argument lists in user-defined functions. This is really quite easy, using the func_num_args(), func_get_arg(), and func_get_args() functions.

No special syntax is required, and argument lists may still be explicitly provided with function definitions and will behave as normal.

Returning values

Values are returned by using the optional return statement. Any type may be returned, including lists and objects.

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

You can't return multiple values from a function, but similar results can be obtained by returning a list.

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

To return a reference from a function, you have to use the reference operator & in both the function declaration and when assigning the returned value to a variable:

```
function &returns_reference()
{
    return $someref;
}

$newref =& returns_reference();
```

`old_function`

The `old_function` statement allows you to declare a function using a syntax identical to PHP/FI2 (except you must replace 'function' with 'old_function'.

This is a deprecated feature, and should only be used by the PHP/FI2->PHP 3 convertor.

Attenzione

Functions declared as `old_function` cannot be called from PHP's internal code. Among other things, this means you can't use them in functions such as usort(), array_walk(), and register_shutdown_function(). You can get around this limitation by writing a wrapper function (in normal PHP 3 form) to call the `old_function`.

Variable functions

PHP supports the concept of variable functions. This means that if a variable name has parentheses appended to it, PHP will look for a function with the same name as whatever the variable evaluates to, and will attempt to execute it. Among other things, this can be used to implement callbacks, function tables, and so forth.

Variable functions won't work with language constructs such as echo(), unset(), isset() and empty(). This is one of the major differences between PHP functions and language constructs.

Esempio 12-1. Variable function example

```
<?php
function foo()
{
    echo "In foo()<br>\n";
}

function bar($arg = ”)
{
    echo "In bar(); argument was ’$arg’.<br>\n";
}

$func = ’foo’;
$func();
$func = ’bar’;
$func(’test’);
?>
```

Capitolo 13. Classes and Objects

`class`

A class is a collection of variables and functions working with these variables. A class is defined using the following syntax:

```
<?php
class Cart {
    var $items;  // Items in our shopping cart

    // Add $num articles of $artnr to the cart

    function add_item ($artnr, $num) {
        $this->items[$artnr] += $num;
    }

    // Take $num articles of $artnr out of the cart

    function remove_item ($artnr, $num) {
        if ($this->items[$artnr] > $num) {
            $this->items[$artnr] -= $num;
            return true;
        } else {
            return false;
        }
    }
}
?>
```

This defines a class named Cart that consists of an associative array of articles in the cart and two functions to add and remove items from this cart.

Cautela

The following cautionary notes are valid for PHP 4.

The name `stdClass` is used interally by Zend and is reserved. You cannot have a class named `stdClass` in PHP.

The function names `__sleep` and `__wakeup` are magical in PHP classes. You cannot have functions with these names in any of your classes unless you want the magic functionality associated with them. See below for more information.

PHP reserves all function names starting with __ as magical. It is recommended that you do not use function names with __ in PHP unless you want some documented magic functionality.

Nota: In PHP 4, only constant initializers for `var` variables are allowed. To initialize variables with non-constant values, you need an initialization function which is called automatically when an object is being constructed from the class. Such a function is called a constructor (see below).

```
/* None of these will work in PHP 4. */
class Cart {
    var $todays_date = date("Y-m-d");
    var $name = $firstname;
    var $owner = 'Fred ' . 'Jones';
    var $items = array("VCR", "TV");
}

/* This is how it should be done. */
class Cart {
    var $todays_date;
    var $name;
    var $owner;
    var $items;

    function Cart() {
        $this->todays_date = date("Y-m-d");
        $this->name = $GLOBALS['firstname'];
        /* etc. . . */
    }
}
```

Classes are types, that is, they are blueprints for actual variables. You have to create a variable of the desired type with the `new` operator.

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

$another_cart = new Cart;
```

```
$another_cart->add_item("0815", 3);
```

This creates the objects $cart and $another_cart, both of the class Cart. The function add_item() of the $cart object is being called to add 1 item of article number 10 to the $cart. 3 items of article number 0815 are being added to $another_cart.

Both, $cart and $another_cart, have functions add_item(), remove_item() and a variable items. These are distinct functions and variables. You can think of the objects as something similar to directories in a filesystem. In a filesystem you can have two different files README.TXT, as long as they are in different directories. Just like with directories where you'll have to type the full pathname in order to reach each file from the toplevel directory, you have to specify the complete name of the function you want to call: In PHP terms, the toplevel directory would be the global namespace, and the pathname separator would be ->. Thus, the names $cart->items and $another_cart->items name two different variables. Note that the variable is named $cart->items, not $cart->$items, that is, a variable name in PHP has only a single dollar sign.

```
// correct, single $
$cart->items  = array("10" => 1);

// invalid, because $cart->$items becomes $cart->""
$cart->$items = array("10" => 1);

// correct, but may or may not be what was intended:
// $cart->$myvar becomes $ncart->items
$myvar = 'items';
$cart->$myvar = array("10" => 1);
```

Within a class definition, you do not know under which name the object will be accessible in your program: At the time the Cart class was written, it was unknown that the object will be named $cart or $another_cart later. Thus, you cannot write $cart->items within the Cart class itself. Instead, in order to be able to access it's own functions and variables from within a class, one can use the pseudo-variable $this which can be read as 'my own' or 'current object'. Thus, '$this->items[$artnr] += $num' can be read as 'add $num to the $artnr counter of my own items array' or 'add $num to the $artnr counter of the items array within the current object'.

extends

Often you need classes with similar variables and functions to another existing class. In fact, it is good practice to define a generic class which can be used in all your projects and adapt this class for the needs of each of your specific projects. To facilitate this, Classes can be extensions of other classes. The extended or derived class has all variables and functions of the base class (this is called 'inheritance' despite the fact that nobody died) and what you add in the extended definition. It is not possible to substract from a class, that is, to undefine any existing functions or variables. An

extended class is always dependent on a single base class, that is, multiple inheritance is not supported. Classes are extended using the keyword 'extends'.

```
class Named_Cart extends Cart {
    var $owner;

    function set_owner ($name) {
        $this->owner = $name;
    }
}
```

This defines a class Named_Cart that has all variables and functions of Cart plus an additional variable $owner and an additional function set_owner(). You create a named cart the usual way and can now set and get the carts owner. You can still use normal cart functions on named carts:

```
$ncart = new Named_Cart;    // Create a named cart
$ncart->set_owner ("kris"); // Name that cart
print $ncart->owner;        // print the cart owners name
$ncart->add_item ("10", 1); // (inherited functionality from cart)
```

Constructors

> **Cautela**
>
> In PHP 3 and PHP 4 constructors behave differently. The PHP 4 semantics are strongly preferred.

Constructors are functions in a class that are automatically called when you create a new instance of a class with new. In PHP 3, a function becomes a constructor when it has the same name as the class. In PHP 4, a function becomes a constructor, when it has the same name as the class it is defined in - the difference is subtle, but crucial (see below).

```
// Works in PHP 3 and PHP 4.
class Auto_Cart extends Cart {
    function Auto_Cart () {
        $this->add_item ("10", 1);
    }
}
```

This defines a class Auto_Cart that is a Cart plus a constructor which initializes the cart with one item of article number "10" each time a new Auto_Cart is being made with "new". Constructors can take arguments and these arguments can be optional, which makes them much more useful. To be

able to still use the class without parameters, all parameters to constructors should be made optional by providing default values.

```
// Works in PHP 3 and PHP 4.
class Constructor_Cart extends Cart {
    function Constructor_Cart ($item = "10", $num = 1) {
        $this->add_item ($item, $num);
    }
}

// Shop the same old boring stuff.

$default_cart   = new Constructor_Cart;

// Shop for real...

$different_cart = new Constructor_Cart ("20", 17);
```

Cautela

In PHP 3, derived classes and constructors have a number of limitations. The following examples should be read carefully to understand these limitations.

```
class A {
  function A() {
    echo "I am the constructor of A.<br>\n";
  }
}

class B extends A {
  function C() {
    "I am a regular function.<br>\n";
  }
}

// no constructor is being called in PHP 3.
$b = new B;
```

In PHP 3, no constructor is being called in the above example. The rule in PHP 3 is: ’A constructor is a function of the same name as the class.’. The name of the class is B, and there is no function called B() in class B. Nothing happens.

This is fixed in PHP 4 by introducing another rule: If a class has no constructor, the constructor of the base class is being called, if it exists. The above example would have printed ’I am the constructor of A.
’ in PHP 4.

```
class A {
  function A() {
    echo "I am the constructor of A.<br>\n";
  }

  function B() {
    echo "I am a regular function named B in class A.<br>\n";
    echo "I am not a constructor in A.<br>\n";
  }
}

class B extends A {
  function C() {
    echo "I am a regular function.<br>\n";
  }
}

// This will call B() as a constructor.
$b = new B;
```

In PHP 3, the function B() in class A will suddenly become a constructor in class B, although it was never intended to be. The rule in PHP 3 is: 'A constructor is a function of the same name as the class.'. PHP 3 does not care if the function is being defined in class B, or if it has been inherited.

This is fixed in PHP 4 by modifying the rule to: 'A constructor is a function of the same name as the class it is being defined in.'. Thus in PHP 4, the class B would have no constructor function of its own and the constructor of the base class would have been called, printing 'I am the constructor of A.
'.

Cautela

Neither PHP 3 nor PHP 4 call constructors of the base class automatically from a constructor of a derived class. It is your responsibility to propagate the call to constructors upstream where appropriate.

> **Nota:** There are no destructors in PHP 3 or PHP 4. You may use register_shutdown_function() instead to simulate most effects of destructors.

Destructors are functions that are called automatically when a variable is destroyed, either with unset() or by simply going out of scope. There are no destructors in PHP.

::

> **Cautela**
>
> The following is valid for PHP 4 only.

Sometimes it is useful to refer to functions and variables in base classes or to refer to functions in classes that have not yet any instances. The :: operator is being used for this.

```
class A {
  function example() {
    echo "I am the original function A::example().<br>\n";
  }
}

class B extends A {
  function example() {
    echo "I am the redefined function B::example().<br>\n";
    A::example();
  }
}

// there is no object of class A.
// this will print
//   I am the original function A::example().<br>
A::example();

// create an object of class B.
$b = new B;

// this will print
//   I am the redefined function B::example().<br>
//   I am the original function A::example().<br>
$b->example();
```

The above example calls the function example() in class A, but there is no object of class A, so that we cannot write $a->example() or similar. Instead we call example() as a 'class function', that is, as a function of the class itself, not any object of that class.

There are class functions, but there are no class variables. In fact, there is no object at all at the time of the call. Thus, a class function may not use any object variables (but it can use local and global variables), and it may no use $this at all.

In the above example, class B redefines the function example(). The original definition in class A is shadowed and no longer available, unless you are refering specifically to the implementation of example() in class A using the ::-operator. Write A::example() to do this (in fact, you should be writing parent::example(), as shown in the next section).

In this context, there is a current object and it may have object variables. Thus, when used from WITHIN an object function, you may use $this and object variables.

`parent`

You may find yourself writing code that refers to variables and functions in base classes. This is particularly `TRUE` if your derived class is a refinement or specialisation of code in your base class.

Instead of using the literal name of the base class in your code, you should be using the special name `parent`, which refers to the name of your base class as given in the `extends` declation of your class. By doing this, you avoid using the name of your base class in more than one place. Should your inheritance tree change during implementation, the change is easily made by simply changing the `extends` declaration of your class.

```
class A {
  function example() {
    echo "I am A::example() and provide basic functionality.<br>\n";
  }
}

class B extends A {
  function example() {
    echo "I am B::example and provide additional functionality().<br>\n";
    parent::example();
  }
}

$b = new B;

// This will call B::example(), which will in turn call A::example().
$b->example();
```

Serializing objects - objects in sessions

Nota: In PHP 3, objects will lose their class association throughout the process of serialization and unserialization. The resulting variable is of type object, but has no class and no methods, thus it is pretty useless (it has become just like an array with a funny syntax).

Cautela

The following information is valid for PHP 4 only.

serialize() returns a string containing a byte-stream representation of any value that can be stored in PHP. **unserialize()** can use this string to recreate the original variable values. Using serialize to save an object will save all variables in an object. The functions in an object will not be saved, only the name of the class.

In order to be able to **unserialize()** an object, the class of that object needs to be defined. That is, if you have an object $a of class A on page1.php and serialize this, you'll get a string that refers to class A and contains all values of variabled contained in $a. If you want to be able to unserialize this on page2.php, recreating $a of class A, the definition of class A must be present in page2.php. This can be done for example by storing the class defintion of class A in an include file and including this file in both page1.php and page2.php.

```
classa.inc:
  class A {
    var $one = 1;

    function show_one() {
      echo $this->one;
    }
  }

page1.php:
  include("classa.inc");

  $a = new A;
  $s = serialize($a);
  // store $s somewhere where page2.php can find it.
  $fp = fopen("store", "w");
  echo $s;
  fclose($fp);

page2.php:
  // this is needed for the unserialize to work properly.
  include("classa.inc");

  $s = implode("", @file("store"));
  unserialize($s);

  // now use the function show_one of the $a object.
  $a->show_one();
```

If you are using sessions and use session_register() to register objects, these objects are serialized automatically at the end of each PHP page, and are unserialized automatically on each of the following pages. This basically means that these objects can show up on any of your pages once they become part of your session.

It is strongly recommended that you include the class definitions of all such registered objects on all of your pages, even if you do not actually use these classes on all of your pages. If you don’t and an object is being unserialized without its class definition being present, it will lose its class association and become an object of class `stdClass` without any functions available at all, that is, it will become quite useless.

So if the in the example above $a became part of a session by running `session_register("a")`, you should include the file `classa.inc` on all of your pages, not only page1.php and page2.php.

The magic functions `__sleep` and `__wakeup`

serialize() checks if your class has a function with the magic name `__sleep`. If so, that function is being run prior to any serialization. It can clean up the object and is supposed to return an array with the names of all variables of that object that should be serialized.

The intended use of `__sleep` is to close any database connections that object may have, committing pending data or perform similar cleanup tasks. Also, the function is useful if you have very large objects which need not be saved completely.

Conversely, **unserialize()** checks for the presence of a function with the magic name `__wakeup`. If present, this function can reconstruct any ressources that object may have.

The intended use of `__wakeup` is to reestablish any database connections that may have been lost during serialization and perform other reinitialization tasks.

References inside the constructor

Creating references within the constructor can lead to confusing results. This tutorial-like section helps you to avoid problems.

```
class foo {
    function foo($name) {
        // create a reference inside the global array $globalref
        global $globalref;
$globalref[] = &$this;
        // set name to passed value
        $this->setName($name);
// and put it out
        $this->echoName();
    }

    function echoName() {
        echo "<br>",$this->Name;
    }
```

```
function setName($name) {
$this->Name = $name;
    }
}
```

Let us check out if there is a difference between `$bar1` which has been created using the copy `=` operator and `$bar2` which has been created using the reference `=&` operator...

```
    $bar1 = new foo('set in constructor');
    $bar1->echoName();
    $globalref[0]->echoName();

    /* output:
    set in constructor
    set in constructor
    set in constructor */

    $bar2 =& new foo('set in constructor');
    $bar2->echoName();
    $globalref[1]->echoName();

    /* output:
    set in constructor
    set in constructor
    set in constructor */
```

Apparently there is no difference, but in fact there is a very significant one: `$bar1` and `$globalref[0]` are _NOT_ referenced, they are NOT the same variable. This is because "new" does not return a reference by default, instead it returns a copy.

Nota: There is no performance loss (since php 4 and up use reference counting) returning copies instead of references. On the contrary it is most often better to simply work with copies instead of references, because creating references takes some time where creating copies virtually takes no time (unless none of them is a large array or object and one of them gets changed and the other(s) one(s) subsequently, then it would be wise to use references to change them all concurrently).

To prove what is written above let us watch the code below.

```
        // now we will change the name. what do you expect?
        // you could expect that both $bar and $global-
ref[0] change their names...
        $bar1->setName('set from outside');
```

```
        // as mentioned before this is not the case.
        $bar1->echoName();
        $globalref[0]->echoName();

        /* output:
        set on object creation
        set from outside */

        // let us see what is different with $bar2 and $globalref[1]
        $bar2->setName('set from outside');

        // luckily they are not only equyl, they are thesame variable
        // thus $bar2->Name and $globalref[1]->Name are the same too
        $bar2->echoName();
        $globalref[1]->echoName();

        /* output:
        set from outside
        set from outside */
```

Another final example, try to understand it.

```
class a {
    function a($i) {
        $this->value = $i;
        // try to figure out why we do not need a reference here
        $this->b = new b($this);
    }

    function createRef() {
        $this->c = new b($this);
    }

    function echoValue() {
        echo "<br>","class ",get_class($this),': ',$this->value;
        }
}


class b  {

    function b(&$a) {
        $this->a = &$a;
    }
```

```
   function echoValue() {
        echo "<br>","class ",get_class($this),': ',$this->a->value;
        }

}

// try to undestand why using a simple copy here would yield
// in an undesired result in the *-marked line
$a =& new a(10);
$a->createRef();

$a->echoValue();
$a->b->echoValue();
$a->c->echoValue();

$a->value = 11;

$a->echoValue();
$a->b->echoValue(); // *
$a->c->echoValue();

/*
output:
class a: 10
class b: 10
class b: 10
class a: 11
class b: 11
class b: 11
*/
```

Capitolo 14. References Explained

What References Are

References are a means in PHP to access the same variable content by different names. They are not like C pointers, they are symbol table aliases. Note that in PHP, variable name and variable content are different, so the same content can have different names. The most close analogy is with Unix filenames and files - variable names are directory entries, while variable contents is the file itself. References can be thought of as hardlinking in Unix filesystem.

What References Do

PHP references allow you to make two variables to refer to the same content. Meaning, when you do:

```
$a =& $b
```

> it means that `$a` and `$b` point to the same variable.
>
> **Nota:** `$a` and `$b` are completely equal here, that's not `$a` is pointing to `$b` or vice versa, that's `$a` and `$b` pointing to the same place.

The same syntax can be used with functions, that return references, and with `new` operator (in PHP 4.0.4 and later):

```
$bar =& new fooclass();
$foo =& find_var ($bar);
```

> **Nota:** Unless you use the syntax above, the result of `$bar = new fooclass()` will not be the same variable as `$this` in the constructor, meaning that if you have used reference to `$this` in the constructor, you should use reference assignment, or you get two different objects.

The second thing references do is to pass variables by-reference. This is done by making a local variable in a function and a variable in the calling scope reference to the same content. Example:

```
function foo (&$var)
{
    $var++;
```

```
}

$a=5;
foo ($a);
```

will make `$a` to be 6. This happens because in the function `foo` the variable `$var` refers to the same content as `$a`. See also more detailed explanations about passing by reference.

The third thing reference can do is return by reference.

What References Are Not

As said before, references aren't pointers. That means, the following construct won't do what you expect:

```
function foo (&$var)
{
    $var =& $GLOBALS["baz"];
}
foo($bar);
```

What happens is that `$var` in `foo` will be bound with `$bar` in caller, but then it will be re-bound with `$GLOBALS["baz"]`. There's no way to bind `$bar` in the calling scope to something else using the reference mechanism, since `$bar` is not available in the function `foo` (it is represented by `$var`, but `$var` has only variable contents and not name-to-value binding in the calling symbol table).

Passing by Reference

You can pass variable to function by reference, so that function could modify its arguments. The syntax is as follows:

```
function foo (&$var)
{
    $var++;
}

$a=5;
foo ($a);
// $a is 6 here
```

Note that there’s no reference sign on function call - only on function definition. Function definition alone is enough to correctly pass the argument by reference.

Following things can be passed by reference:

- Variable, i.e. `foo($a)`
- New statement, i.e. `foo(new foobar())`
- Reference, returned from a function, i.e.:

```
function &bar()
{
    $a = 5;
    return $a;
}
foo(bar());
```

See also explanations about returning by reference.

Any other expression should not be passed by reference, as the result is undefined. For example, the following examples of passing by reference are invalid:

```
function bar() // Note the missing &
{
    $a = 5;
    return $a;
}
foo(bar());

foo($a = 5) // Expression, not variable
foo(5) // Constant, not variable
```

These requirements are for PHP 4.0.4 and later.

Returning References

Returning by-reference is useful when you want to use a function to find which variable a reference should be bound to. When returning references, use this syntax:

```
function &find_var ($param)
{
    ...code...
    return $found_var;
}
```

```
$foo =& find_var ($bar);
$foo->x = 2;
```

In this example, the property of the object returned by the `find_var` function would be set, not the copy, as it would be without using reference syntax.

> **Nota:** Unlike parameter passing, here you have to use `&` in both places - to indicate that you return by-reference, not a copy as usual, and to indicate that reference binding, rather than usual assignment, should be done for `$foo`.

Unsetting References

When you unset the reference, you just break the binding between variable name and variable content. This does not mean that variable content will be destroyed. For example:

```
$a = 1;
$b =& $a;
unset ($a);
```

won't unset `$b`, just `$a`.

Again, it might be useful to think about this as analogous to Unix **unlink** call.

Spotting References

Many syntax constructs in PHP are implemented via referencing mechanisms, so everything told above about reference binding also apply to these constructs. Some constructs, like passing and returning by-reference, are mentioned above. Other constructs that use references are:

`global` References

When you declare variable as **global $var** you are in fact creating reference to a global variable. That means, this is the same as:

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

That means, for example, that unsetting `$var` won't unset global variable.

`$this`

In an object method, `$this` is always reference to the caller object.

III. Caratteristiche

Capitolo 15. Gestione degli errori

In PHP sono presenti diversi tipi di errori e avvertimenti (warning):

Tabella 15-1. PHP - Tipi di errore

valore	simbolo	descrizione	note

I valori presenti nella tabella (sia numerici che simbolici) sono utilizzati per creare delle bitmask per specificare quali errori da cercare. Si possono usare gli operatori sui bit '|','&' e '~' per combinare questi valori e mascherare certi tipi di errori. Le configurazioni predefinite per effettuare il report di tutti gli errori eccetto le notifiche sono `E_ALL & ~E_NOTICE` per PHP4 e 7 per PHP3 (PHP3 non supporta le costanti simboliche).

Le configurazioni possono essere cambiate nel file ini con la direttiva error_reporting. Si può anche utilizzare il file di configurazione di Apache httpd.conf con la direttiva php_error_reporting (php3_error_reporting per PHP 3) oppure ancora in fase di esecuzione di uno script con la funzione error_reporting().

> **Attenzione**
>
> Quando si esegue un'upgrade del codice o dei server da PHP3 a PHP4 è necessario controllare questi settaggi e le chiamate a error_reporting() oppure potrebbe disabilitarsi il report dei nuovi tipi di errore, specialmente E_COMPILE_ERROR. Questo potrebbe portare a documenti vuoti senza alcun feedback sulle cause o dove guardare per trovare il problema.

Tutte le espressioni PHP possono anche venir chiamate con il prefisso "@", che disabilita il report degli errori per quella particolare espressione. Se capita un errore in una di queste espressioni e l'opzione track_errors è attivata, si può trovare il messaggio d'errore nella variabile globale $php_errormsg.

> **Attenzione**
>
> Attualmente il prefisso "@" disabilita il report anche per gli errori critici che terminano l'esecuzione dello script.

Capitolo 16. Creazione e manipolazione di immagini

PHP non è limitato alla creazione di output HTML. Può anche essere usato per creare e manipolare file di immagini in una varietà di differenti formati, inclusi gif, png, jpg, wbmp e xpm. Ancora più convenientemente, php può visualizzare una immagine da esso creata, direttamente in un browser. E‘ necessario compilare PHP con le librerie GD per poter usare queste funzioni. GD e PHP potrebbero necessitare di altre librerie, a seconda di quali formati immagine si desidera usare. GD ha smesso di supportare le immagini Gif dalla versione 1.6.

Esempio 16-1. Creazione di PNG usando PHP

```
<?php
    Header("Content-type: image/png");
    $string=implode($argv," ");
    $im = imageCreateFromPng("images/button1.png");
    $orange = ImageColorAllocate($im, 220, 210, 60);
    $px = (imagesx($im)-7.5*strlen($string))/2;
    ImageString($im,3,$px,9,$string,$orange);
    ImagePng($im);
    ImageDestroy($im);
?>
```

Questo esempio può essere chiamato da una pagina con un tag tipo: <img src="button.php?text"> Lo script button.php presentato sopra prende la stringa "text" e la sovrappone ad una immagine base che, in questo caso è "images/button1.png" e visualizza l’immagine risultante. Questo è un modo molto conveniente per evitare di disegnare nuove immagini di bottoni ogni volta che si desidera modificare il testo di un bottone. Con questo metodo esse sono generate dinamicamente.

Capitolo 17. Autenticazione HTTP usando PHP

I meccanismi di Autenticazione HTTP sono disponibili in PHP solo quando questo viene usato come un modulo di Apache ed esso non è quindi disponibile nella versione CGI. In uno script PHP modulo di Apache, è possibile usare la funzione header() per inviare un messaggio di "Authentication Required" al browser dell'utente, provocando quindi l'apertura di una finestra contenente una richiesta di Nome utente/Password. Una volta che l'utente ha compilato i campi nome utente e password, l'URL contenente lo script PHP verrà richiamato nuovamente usando le variabili, $PHP_AUTH_USER, $PHP_AUTH_PW e $PHP_AUTH_TYPE impostate con, rispettivamente: nome, password e tipo di autenticazione. Solamente il tipo di autenticazione "Basic" è al momento supportato. Fare riferimento alla funzione header() per ulteriori informazioni.

Un frammento di script di esempio che richiede l'autenticazione da parte del browser su una pagina, potrebbe essere il seguente:

Esempio 17-1. Esempio di Autenticazione HTTP

```
<?php
  if(!isset($PHP_AUTH_USER)) {
    Header("WWW-Authenticate: Basic realm=\"Il mio Regno\"");
    Header("HTTP/1.0 401 Unauthorized");
    echo "Messaggio da inviare se si preme il tasto Cancel\n";
    exit;
  } else {
    echo "Ciao $PHP_AUTH_USER.<P>";
    echo "Hai inserito $PHP_AUTH_PW come tua password.<P>";
  }
?>
```

> **Note:** Fare molta attenzione quando si scrive codice per le intestazioni HTTP. Per ottenere la massima compatibilità con tutti i client, la paorla-chiave "Basic" deve essere scritta con una "B" maiuscola, la stringa realm deve essere racchiusa in virgolette doppie (non singole), ed esattamente uno spazio deve precedere il codice "401" nella linea di intestazione "HTTP/1.0 401".

Invece di stampare semplicemente $PHP_AUTH_USER e $PHP_AUTH_PW, probabilmente si vorrà controllare la validità dello username e della password. Probabilemnte inviando una chiamata al database, o cercando un utente in un file dbm.

Si faccia attenzione ad alcune versioni di Internet Explorer. Sembra che siano molto schizzinosi riguardo all'ordine delle intestazioni. Inviare l'intestazione *WWW-Authenticate* prima dell'intestazione HTTP/1.0 401 sembra sistemare le cose per il momento.

Al fine di prevenire che qualcuno scriva uno script che rivela la password di una pagina che era stata autenticata tramite un tradizionale meccanismo esterno, le variabili PHP_AUTH non verranno impostate se è abilitata l'autenticazione esterna per quella determinata pagina. In questo caso, la variabile $REMOTE_USER può essere usata per identificare un utente autenticato esternamente.

> **Nota sulla Configurazione:** PHP usa la presenza di una direttiva `AuthType` per determinare se viene utilizzata l'autenticazione esterna. Evitare questa direttiva nel contesto dove si intende usare l'autenticazione con PHP (altrimenti ogni tentativo di autenticazione fallirà).

Si fa notare, però, che quanto scritto sopra non impedisce a qualcuno che controlla un URL non autenticato di sottrarre password da URL autenticati presenti sullo stesso server.

Sia Netscape Navigator che Internet Explorer cancellano la cache locale della finestra del browser, per quanto riguarda il realm, al ricevimento di una risposta 401 da parte del server. Questo è effettivamente un meccanismo di "log out" per l'utente, che lo forza a reinserire lo username e la password. Alcune persone usano questo per fare il "time out" dei login, o per rendere disponibili bottoni di "log-out".

Esempio 17-2. Esempio di Autenticazione HTTP che impone l'inserimento di nuovo username/password

```
<?php
  function authenticate() {
    Header( "WWW-
Authenticate: Basic realm=\"Prova del Sistema di Autenticazione\"");
    Header( "HTTP/1.0 401 Unauthorized");
    echo "Per poter accedere a questa risorsa occorre inserire una cop-
pia login e password valide\n";
    exit;
  }

  if(!isset($PHP_AUTH_USER) || ($SeenBefore == 1 && !str-
cmp($OldAuth, $PHP_AUTH_USER)) ) {
   authenticate();
  }
  else {
   echo "Benvenuto: $PHP_AUTH_USER<BR>";
   echo "Vecchio: $OldAuth";
   echo "<FORM ACTION=\"$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";
  }
?>
```

Questo comportamento non è richiesto dallo standard di autenticazione HTTP Basic, quindi non si dovrebbe mai fare affidamento su di esso. Test effettuati con Lynx mostrano che Lynx non cancella le credenziali di autenticazione al ricevimento del codice di risposta 401 da parte del server, quindi, premendo indietro e avanti nuovamente, darà nuovamente accesso alla risorsa, ammesso che le rispettive richieste di credenziali non siano cambiate. L'utente può però premere il tasto '_' al fine di cancellare le sue informazioni di autenticazione.

Si noti anche che questo non funziona con il server IIS di Microsoft e con la versione CGI di PHP a causa di una limitazione di IIS.

Capitolo 18. Cookies

PHP supporta in modo trasparente i cookies HTTP. I cookies sono un meccanismo per memorizzare dati nel browser remoto e tenere traccia degli utenti o identificarli al loro ritorno. I cookies possono essere impostati tramite la funzione setcookie(). I cookies sono parte dell'intestazione HTTP, quindi setcookie() deve essere chiamata prima che qualsiasi output sia inviato al browser. Si tratta della stessa limitazione della funzione header().

Ogni cookie inviato dal client sarà automaticamente trasformato in una variabile PHP, come avviene nel caso di dati GET o POST. Se si vogliono assegnare più valori ad un singolo cookie, basta aggiungere *[]* al nome del cookie. Per maggiori dettagli si veda la funzione setcookie().

Capitolo 19. Handling file uploads

POST method uploads

PHP is capable of receiving file uploads from any RFC-1867 compliant browser (which includes Netscape Navigator 3 or later, Microsoft Internet Explorer 3 with a patch from Microsoft, or later without a patch). This feature lets people upload both text and binary files. With PHP's authentication and file manipulation functions, you have full control over who is allowed to upload and what is to be done with the file once it has been uploaded.

Note that PHP also supports PUT-method file uploads as used by Netscape Composer and W3C's Amaya clients. See the PUT Method Support for more details.

A file upload screen can be built by creating a special form which looks something like this:

Esempio 19-1. File Upload Form

```
<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>
```

The _URL_ should point to a PHP file. The MAX_FILE_SIZE hidden field must precede the file input field and its value is the maximum filesize accepted. The value is in bytes.

> **Attenzione**
>
> The MAX_FILE_SIZE is advisory to the browser. It is easy to circumvent this maximum. So don't count on it that the browser obeys you wish! The PHP-settings for maximum-size, however, cannot be fooled.

In PHP, the following variables will be defined within the destination script upon a successful upload, assuming that register_globals is turned on in `php.ini`. If track_vars is turned on, they will also be available in PHP within the global array `$HTTP_POST_VARS`. Note that the following variable names assume the use of the file upload name 'userfile', as used in the example above:

- `$userfile` - The temporary filename in which the uploaded file was stored on the server machine.
- `$userfile_name` - The original name or path of the file on the sender's system.
- `$userfile_size` - The size of the uploaded file in bytes.

- `$userfile_type` - The mime type of the file if the browser provided this information. An example would be "image/gif".

Note that the "$userfile" part of the above variables is whatever the name of the INPUT field of TYPE=file is in the upload form. In the above upload form example, we chose to call it "userfile"

In PHP 4, the behaviour is slightly different, in that the new global array `$HTTP_POST_FILES` is provided to contain the uploaded file information. This is still only available if track_vars is turned on, but track_vars is always turned on in versions of PHP after PHP 4.0.2.

The contents of `$HTTP_POST_FILES` are as follows. Note that this assumes the use of the file upload name 'userfile', as used in the example above:

`$HTTP_POST_FILES['userfile']['name']`

The original name of the file on the client machine.

`$HTTP_POST_FILES['userfile']['type']`

The mime type of the file, if the browser provided this information. An example would be `"image/gif"`.

`$HTTP_POST_FILES['userfile']['size']`

The size, in bytes, of the uploaded file.

`$HTTP_POST_FILES['userfile']['tmp_name']`

The temporary filename of the file in which the uploaded file was stored on the server.

Files will by default be stored in the server's default temporary directory, unless another location has been given with the upload_tmp_dir directive in `php.ini`. The server's default directory can be changed by setting the environment variable TMPDIR in the environment in which PHP runs. Setting it using putenv() from within a PHP script will not work. This environment variable can also be used to make sure that other operations are working on uploaded files, as well.

Esempio 19-2. Validating file uploads

The following examples are for versions of PHP 3 greater than 3.0.16, and versions of PHP 4 greater than 4.0.2. See the function entries for is_uploaded_file() and move_uploaded_file().

```
<?php
if (is_uploaded_file($userfile)) {
    copy($userfile, "/place/to/put/uploaded/file");
} else {
    echo "Possible file upload attack: filename '$userfile'.";
}
/* ...or... */
move_uploaded_file($userfile, "/place/to/put/uploaded/file");
?>
```

For earlier versions of PHP, you'll need to do something like the following.

Nota: This will *not* work in versions of PHP 4 after 4.0.2. It depends on internal functionality of PHP which changed after that version.

```
<?php
/* Userland test for uploaded file. */
function is_uploaded_file($filename) {
    if (!$tmp_file = get_cfg_var('upload_tmp_dir')) {
        $tmp_file = dirname(tempnam('', ''));
    }
    $tmp_file .= '/' . basename($filename);
    /* User might have trailing slash in php.ini... */
    return (ereg_replace('/+', '/', $tmp_file) == $filename);
}

if (is_uploaded_file($userfile)) {
    copy($userfile, "/place/to/put/uploaded/file");
} else {
    echo "Possible file upload attack: filename '$userfile'.";
}
?>
```

The PHP script which receives the uploaded file should implement whatever logic is necessary for determining what should be done with the uploaded file. You can for example use the `$file_size` variable to throw away any files that are either too small or too big. You could use the `$file_type` variable to throw away any files that didn't match a certain type criteria. Whatever the logic, you should either delete the file from the temporary directory or move it elsewhere.

The file will be deleted from the temporary directory at the end of the request if it has not been moved away or renamed.

Common Pitfalls

The `MAX_FILE_SIZE` item cannot specify a file size greater than the file size that has been set in the upload_max_filesize ini-setting. The default is 2 Megabytes.

Not validating which file you operate on may mean that users can access sensitive information in other directories.

Please note that the CERN httpd seems to strip off everything starting at the first whitespace in the content-type mime header it gets from the client. As long as this is the case, CERN httpd will not support the file upload feature.

Uploading multiple files

It is possible to upload multiple files simultaneously and have the information organized automatically in arrays for you. To do so, you need to use the same array submission syntax in the HTML form as you do with multiple selects and checkboxes:

> **Nota:** Support for multiple file uploads was added in version 3.0.10.

Esempio 19-3. Uploading multiple files

```
<form action="file-upload.php" 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>
```

When the above form is submitted, the arrays `$userfile`, `$userfile_name`, and `$userfile_size` will be formed in the global scope (as well as in $HTTP_POST_FILES ($HTTP_POST_VARS in PHP 3)). Each of these will be a numerically indexed array of the appropriate values for the submitted files.

For instance, assume that the filenames `/home/test/review.html` and `/home/test/xwp.out` are submitted. In this case, `$userfile_name[0]` would contain the value `review.html`, and `$userfile_name[1]` would contain the value `xwp.out`. Similarly, `$userfile_size[0]` would contain `review.html`'s filesize, and so forth.

`$userfile['name'][0]`, `$userfile['tmp_name'][0]`, `$userfile['size'][0]`, and `$userfile['type'][0]` are also set.

PUT method support

PHP provides support for the HTTP PUT method used by clients such as Netscape Composer and W3C Amaya. PUT requests are much simpler than a file upload and they look something like this:

```
PUT /path/filename.html HTTP/1.1
```

This would normally mean that the remote client would like to save the content that follows as: /path/filename.html in your web tree. It is obviously not a good idea for Apache or PHP to automatically let everybody overwrite any files in your web tree. So, to handle such a request you have to first tell your web server that you want a certain PHP script to handle the request. In Apache you do this with the *Script* directive. It can be placed almost anywhere in your Apache configuration file. A common place is inside a <Directory> block or perhaps inside a <Virtualhost> block. A line like this would do the trick:

```
Script PUT /put.php
```

This tells Apache to send all PUT requests for URIs that match the context in which you put this line to the put.php script. This assumes, of course, that you have PHP enabled for the .php extension and PHP is active.

Inside your put.php file you would then do something like this:

```
<?php copy($PHP_UPLOADED_FILE_NAME,$DOCUMENT_ROOT.$REQUEST_URI); ?>
```

This would copy the file to the location requested by the remote client. You would probably want to perform some checks and/or authenticate the user before performing this file copy. The only trick here is that when PHP sees a PUT-method request it stores the uploaded file in a temporary file just like those handled but the POST-method. When the request ends, this temporary file is deleted. So, your PUT handling PHP script has to copy that file somewhere. The filename of this temporary file is in the $PHP_PUT_FILENAME variable, and you can see the suggested destination filename in the $REQUEST_URI (may vary on non-Apache web servers). This destination filename is the one that the remote client specified. You do not have to listen to this client. You could, for example, copy all uploaded files to a special uploads directory.

Capitolo 20. Utilizzo di file remoti

Quando viene abilitato il supporto per l "URL fopen wrapper" durante la configurazione di PHP (avviene automaticamente a meno che si specifichi espressamente il flag `--disable-url-fopen-wrapper`), si possono usare URL FTP e HTTP con la maggior parte delle funzioni che richiedono nomi di file come parametri, incluse le funzioni require() e include().

Nota: Non si possono usare i file remoti con include() e require() sotto Windows.

Per esempio, si può usare per aprire un file da un web server remoto, elaborare i dati presi da remoto, e usarli per effetuare delle query, o semplicemente visualizzarli con lo stile del proprio sito web.

Esempio 20-1. Legere il titolo di una pagina web remota

```
<?php
  $file = fopen("http://www.php.net/", "r");
  if (!$file) {
    echo "<p>Errore nell'apertura del file remoto.\n";
    exit;
  }
  while (!feof($file)) {
    $line = fgets($file, 1024);
    /* Funziona solo se i tag del titolo sono sulla stessa linea. */
    if (eregi("<title>(.*)</title>", $line, $out)) {
      $title = $out[1];
      break;
    }
  }
  fclose($file);
?>
```

Si può anche scrivere in un file remoto via FTP se l'utente con cui ci si connette ha le autorizzazioni necessarie, e il file non è già presente. Per connettersi con un utenti specifico si ha bisogno di specificare lo username (e la relativa password) dentro l'URL in questo modo: 'ftp://user:password@ftp.test.com/dir/del/file'. (Si può usare lo stesso tipo di sintassi per accedere a file via HTTP quando richiedono autenticazione).

Esempio 20-2. Salvataggio di dati su server remoto

```
<?php
  $file = fopen("ftp://ftp.php.net/incoming/outputfile", "w");
  if (!$file) {
```

```
    echo "<p>Errore nell'apertura del file remoto in scrittura.\n";
    exit;
  }
  /* Scrittura del file. */
  fputs($file, "$HTTP_USER_AGENT\n");
  fclose($file);
?>
```

Nota: Dall'esempio precedente ci si può fare un'idea di come usare questa tecnica per effettuare dei log in remoto, ma come già accennato non è possibile scrivere su file già esistenti con questo sitema. Per fare una procedura di log distribuito è più indicata la funzione syslog().

Capitolo 21. Connection handling

Nota: The following applies to 3.0.7 and later.

Internally in PHP a connection status is maintained. There are 3 possible states:

- 0 - NORMAL
- 1 - ABORTED
- 2 - TIMEOUT

When a PHP script is running normally the NORMAL state, is active. If the remote client disconnects the ABORTED state flag is turned on. A remote client disconnect is usually caused by the user hitting his STOP button. If the PHP-imposed time limit (see set_time_limit()) is hit, the TIMEOUT state flag is turned on.

You can decide whether or not you want a client disconnect to cause your script to be aborted. Sometimes it is handy to always have your scripts run to completion even if there is no remote browser receiving the output. The default behaviour is however for your script to be aborted when the remote client disconnects. This behaviour can be set via the ignore_user_abort php3.ini directive as well as through the corresponding php3_ignore_user_abort Apache .conf directive or with the ignore_user_abort() function. If you do not tell PHP to ignore a user abort and the user aborts, your script will terminate. The one exception is if you have registered a shutdown function using register_shutdown_function(). With a shutdown function, when the remote user hits his STOP button, the next time your script tries to output something PHP will detect that the connection has been aborted and the shutdown function is called. This shutdown function will also get called at the end of your script terminating normally, so to do something different in case of a client diconnect you can use the connection_aborted() function. This function will return `TRUE` if the connection was aborted.

Your script can also be terminated by the built-in script timer. The default timeout is 30 seconds. It can be changed using the max_execution_time php3.ini directive or the corresponding php3_max_execution_time Apache .conf directive as well as with the set_time_limit() function. When the timer expires the script will be aborted and as with the above client disconnect case, if a shutdown function has been registered it will be called. Within this shutdown function you can check to see if a timeout caused the shutdown function to be called by calling the connection_timeout() function. This function will return `TRUE` if a timeout caused the shutdown function to be called.

One thing to note is that both the ABORTED and the TIMEOUT states can be active at the same time. This is possible if you tell PHP to ignore user aborts. PHP will still note the fact that a user may have broken the connection, but the script will keep running. If it then hits the time limit it will be aborted and your shutdown function, if any, will be called. At this point you will find that connection_timeout() and connection_aborted() return `TRUE`. You can also check both states in a

single call by using the connection_status(). This function returns a bitfield of the active states. So, if both states are active it would return 3, for example.

Capitolo 22. Connessioni Persistenti ai Database

Connessioni persistenti sono collegamenti SQL che non vengono chiusi quando l'esecusione di uno script viene terminata. Quando è richiesta una connessione persistente, PHP controlla se esiste già una identica connessione persistente (che è rimasta aperta da prima) - e se esiste, la usa. Se non esiste, crea il collegamento. Una connessione 'identica' è una connessione aperta verso lo stesso host, con lo stesso username e la stessa password (dove applicabile).

Chi non ha molta familiarità con il modo in cui lavorano i server web e di come essi distribuiscano il carico potrebbero confondere le connessioni permanenti per ciò che esse non sono. In particolare, *non* danno la possibilità di aprire 'sessioni utente' sullo stesso collegamento SQL, *non* danno la possibilità di gestire una transazione in maniera efficiente e sopratutto non fanno molte altre cose. Infatti, per essere molto chiari su questo punto, le connessioni persistenti non hanno *nessuna* funzionalià che non era disponibile con le loro omonime non-persistenti.

Perché?

Questo ha a che fare con il modo in cui i server web operano. Ci sono tre modi in cui un server web può utilizzare PHP per generare le pagine web.

Il primo metodo ' quello di usare il PHP come un "wrapper" (involucro) CGI. Quando viene eseguito in questa modalità, per ogni pagina che viene richiesta al server web che contenga del codice PHP, viene creata e, alla fine dell'esecuzione, distrutta, una istanza dell'interprete PHP. Poiché viene distrutta alla fine di ogni richiesta, qualunque risorsa abbia acquisito (come ad esempio un collegamento ad un server di database SQL) verrà anch'essa distrutta. In questo caso, non vi è nessun guadagno nell'usare le connessioni persistenti -- semplicemente esse non persistono.

Il secondo, e più popolare, metodo è quello di eseguire il programma PHP come modulo in un server web multiprocesso, cosa che attualmente include solo Apache. Un server multiprocesso ha tipicamente un processo (il padre) che coordina un insieme di processi (i suoi figli) che sono quelli che generano le pagine web. Quando arriva una richiesta da parte di un client, questa è passata ad uno dei figli che in quel momento non è già occupato a servire un altro client. Questo significa che quando lo stesso client effettua una seconda richiesta al server, esso potrà essere servito da un diverso processo figlio rispetto a quello della prima volta. In questa situazione, usare una connessione persistente, permette di stabilire una e una sola connessione al database SQL per ogni processo figlio, poiché ciascun processo figlio necessita di collegarsi al database SQL solo la prima volta che richiama una pagina che ne fa uso. Quando viene richamata un'altra pagina che necessita di una connessione al server SQL, essa può riutilizzare la connessione che lo stesso processo figlio aveva stabilito in precedenza.

L'ultimo metodo è quello di usare il PHP come una plug-in per un server web multithread. Attualmente PHP 4 supporta ISAPI, WSAPI e NSAPI (su piattaforma Windows), i quali permettono di usare PHP come una plug-in sui server web multithread come FastTrack di Netscape, Internet Information Server (IIS) di Microsoft e WebSite Pro di O'Reilly. Il comportamento è essenzialmente lo stesso che si ha nel modello multiprocesso descritto prima. Si noti che il supporto SAPI non è disponibile in PHP 3.

Se le connessioni persistenti non hanno nessuna funzionalità aggiuntiva, perch´ usarle?

La risposta, in questo caso è estremamente semplice: efficienza. Le connessioni persistenti sono utili se il carico di lavoro necessario per aprire una connessione al server SQL è alto. Che il carico sia molto alto o meno dipende da molti fattori. Quali, il tipo di database che si utilizza, il fatto che esso si trovi sulla stessa macchina sulla quale si trova il server web, quanto carico di lavoro ha la macchina sulla quale si trova il server SQL, e molte altre ragioni. Il fatto importante è che se il carico di lavoro necessario per aprire le connessioni è alto, le connessioni persistenti possono aiutare in maniera considerevole. Fanno in modo che il processo figlio si colleghi semplicemente una volta durante la sua intera vita, invece di collegarsi ogni volta che processa una pagina che richiede una connessione al server SQL. Questo significa che per ogni figlio che ha aperto una connessione persistente, si avrà una nuova connessione persistente aperta verso il server. Per esempio, se si hanno 20 diversi processi figlio che eseguono uno script che crea una connessione persistente al server SQL server, si avranno 20 diverse connessioni al server SQL, una per ogni figlio.

Si fa notare, tuttavia, che questo può avere degli svantaggi se si fa uso di un database che ha un limite al numero di connessioni, minore rispetto al numero delle connessioni persistenti dei procesi figlio. Se per esempio di usa un database con 16 connessioni simultanee, e durante un periodo di intensa attività del web server, 17 processi figlio cercano di collegarsi, uno non sarà in grado di farlo. Se nei vostri script ci sono bug che non permettono alle connessioni di chiudersi in maniera regolare (come ad esempio un loop infinito), un database con sole 32 connessioni diventerà rapidamente saturo. Fare riferimento alla documentazione del database per informazioni su come gestire connessioni abbandonate o inattive.

Sommario importante. Le connessioni persistenti sono state pensate per avere una mappatura uno-a-uno con le connessioni di tipo normale. Ciò significa che si dovrebbe *sempre* essere in grado di cambiare una connessione persistente con una connessione non-persistente, e che questo non dovrebbe cambiare il modo in cui lo script si comporta. *Può* (e probabilmente succederà) cambiare l’efficienza dello script, ma non il suo comportamento!

Capitolo 23. Modalità sicura (Safe mode)

La modalità Safe Mode è un tentativo di risolvere il problema di sicurezza derivante dalla condivisione del server. Dal punto di vista architetturale non è corretto cercare di risolvere questo problema al livello del PHP, ma poiché le alternative al livello del web server e del SO (Sistema Operativo) non sono realistiche, in molti, specialmente ISP (Internet Service Provider), utilizzano la modalità sicura.

Le direttive di configurazione che controllano la modalità sicura sono:

```
safe_mode = Off
open_basedir =
safe_mode_exec_dir =
safe_mode_allowed_env_vars = PHP_
safe_mode_protected_env_vars = LD_LIBRARY_PATH
disable_functions =
```

Quando safe_mode è attiva (on), il PHP verifica se il proprietario dello script in esecuzione e il proprietario del file su cui si sta operando con una funzione sui file, coincidono. Per esempio:

```
-rw-rw-r--    1 rasmus   rasmus       33 Jul  1 19:20 script.php
-rw-r--r--    1 root     root       1116 May 26 18:01 /etc/passwd
```

Eseguendo questo script.php

```
<?php
 readfile('/etc/passwd');
?>
```

con la modalità sicura attiva si ottiene il seguente errore:

```
Warning: SAFE MODE Restriction in ef-
fect. The script whose uid is 500 is not
allowed to access /etc/passwd owned by uid 0 in /doc-
root/script.php on line 2
```

Ovvero non è possibile accedere al file /etc/passwd in quanto l'utente che esegue lo script non coindice con l'utente proprietario del file.

Se, invece di safe_mode, viene definita una directory open_basedir allora tutte le operazioni sui file saranno limitate ai file sottostanti la directory specificata. Per esempio (nel file httpd.conf di Apache):

```
<Directory /docroot>
php_admin_value open_basedir /docroot
</Directory>
```

Se si esegue lo stesso script.php con questa impostazione di open_basedir si ottiene il seguente risultato:

```
Warning: open_basedir restriction in effect. File is in wrong directory in
/docroot/script.php on line 2
```

Ovvero il file è in una directory non accessibile tramite script in PHP.

È possibile inoltre disabilitare le singole funzioni. Se si aggiunge la seguente riga al file php.ini:

```
disable_functions readfile,system
```

Si ottiene il seguente risultato:

```
Warning: readfile() has been disabled for security reasons in
/docroot/script.php on line 2
```

Ovvero la funzione readfile() è stata disabilitata per motivi di sicurezza.

Funzioni limitate/disabilitate dalla modalità sicura (safe-mode)

Questo è un elenco probabilmente ancora incompleto e forse non esatto delle funzioni limitate da safe-mode.

Tabella 23-1. Funzioni limitate dalla modalità sicura

Funzioni	Limitazioni

IV. Guida Funzioni

I. Apache-specific Functions

apache_lookup_uri (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Perform a partial request for the specified URI and return all info about it

```
object apache_lookup_uri (string filename)
```

This performs a partial request for a URI. It goes just far enough to obtain all the important information about the given resource and returns this information in a class. The properties of the returned class are:

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

Nota: apache_lookup_uri() only works when PHP is installed as an Apache module.

apache_note (PHP 3>= 3.0.2, PHP 4 >= 4.0.0)

Get and set apache request notes

```
string apache_note (string note_name [, string note_value])
```

apache_note() is an Apache-specific function which gets and sets values in a request's `notes` table. If called with one argument, it returns the current value of note `note_name`. If called with two arguments, it sets the value of note `note_name` to `note_value` and returns the previous value of note `note_name`.

ascii2ebcdic (PHP 3>= 3.0.17)

Translate string from ASCII to EBCDIC

```
int ascii2ebcdic (string ascii_str)
```

ascii2ebcdic() is an Apache-specific function which is available only on EBCDIC based operating systems (OS/390, BS2000). It translates the ASCII encoded string *`ascii_str`* to its equivalent EBCDIC representation (binary safe), and returns the result.

See also the reverse function ebcdic2ascii()

ebcdic2ascii (PHP 3>= 3.0.17)

Translate string from EBCDIC to ASCII

```
int ebcdic2ascii (string ebcdic_str)
```

ebcdic2ascii() is an Apache-specific function which is available only on EBCDIC based operating systems (OS/390, BS2000). It translates the EBCDIC encoded string *`ebcdic_str`* to its equivalent ASCII representation (binary safe), and returns the result.

See also the reverse function ascii2ebcdic()

getallheaders (PHP 3, PHP 4 >= 4.0.0)

Fetch all HTTP request headers

```
array getallheaders ()
```

This function returns an associative array of all the HTTP headers in the current request.

> **Nota:** You can also get at the value of the common CGI variables by reading them from the environment, which works whether or not you are using PHP as an Apache module. Use phpinfo() to see a list of all of the environment variables defined this way.

Esempio 1. getallheaders() Example

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

This example will display all the request headers for the current request.

> **Nota: getallheaders()** is currently only supported when PHP runs as an Apache module.

virtual (PHP 3, PHP 4 >= 4.0.0)

Perform an Apache sub-request

```
int virtual (string filename)
```

virtual() is an Apache-specific function which is equivalent to <!--#include virtual...--> in mod_include. It performs an Apache sub-request. It is useful for including CGI scripts or .shtml files, or anything else that you would parse through Apache. Note that for a CGI script, the script must generate valid CGI headers. At the minimum that means it must generate a Content-type header. For PHP files, you need to use include() or require(); **virtual()** cannot be used to include a document which is itself a PHP file.

II. Funzioni di Array

Queste funzioni permettono di manupolare e interagire con gli array in vari modi. Gli Array sono indispensabili per immagazzinare, mantenere, e operare su gruppi di variabili.

Sono supportati sia array semplici che multi-dimensionali, che possono essere sia creati dall'utente che da funzioni. Ci sono specifiche funzioni di database per riempire gli array a partire da interrogazioni sui dati, e parecchie funzioni restituiscono array.

Vedere la sezione Array del manuale per una spiegazione dettagliata di come gli array siano implementati ed usati in PHP.

Vedere anche is_array(), explode(), implode(), split() e join().

array (unknown)

Crea un array

```
array array ([mixed ...])
```

Restituisce un array contenente i parametri. Ai parametri si può dare un indice con l'operatore =>.

> **Nota: array()** è un costrutto del linguaggio usato per rappresentare array letterali, e non una normale funzione.

La sintassi "indice => valori", separati da virgole, definisce indici e valori. indice può essere di tipo string o numerico. Quando l'indice è omesso, viene generato automaticamente un indice intero, a partire da 0. Se l'indice è un intero, il successivo indice generato sarà l'indice intero più grande + 1. Si noti che quando due indici identici vengono definiti, l'ultimo sovrascrive il primo.

L'esempio seguente dimostra come creare un array bidimensionale, come specificare le chiavi per gli array associativi, e come modificare la serie degli indici numerici negli array normali.

Esempio 1. Esempio di array()

```
$frutta = array (
    "frutta"  => array ("a"=>"arancia", "b"=>"banana", "c"=>"mela"),
    "numeri" => array (1, 2, 3, 4, 5, 6),
    "buche"   => array ("prima", 5 => "seconda", "terza")
);
```

Esempio 2. Indice automatico con array()

```
$array = array( 1, 1, 1, 1,  1, 8=>1,  4=>1, 19, 3=>13);
print_r($array);
```

che stamperà :

```
Array
(
    [0] => 1
    [1] => 1
    [2] => 1
    [3] => 13
    [4] => 1
```

```
    [8] => 1
    [9] => 19
)
```

Si noti che l'indice '3' è definito due volte, e che mantiene il valore finale 13. L'indice 4 è definito dopo l'indice 8, e il successivo indice generato (valore 19) è 9, dal momento che l'indice più grande era 8.

Questo esempio crea un array che parte da 1 (1-based).

Esempio 3. Indice 1-based con array()

```
$primotrimestre  = array(1 => 'Gennaio', 'Febbraio', 'Marzo');
print_r($primotrimestre);
```

che stamperà :

```
Array
(
    [1] => 'Gennaio'
    [2] => 'Febbraio'
    [3] => 'Marzo'
)
```

Vedere anche: list().

array_count_values (PHP 4 >= 4.0.0)

Conta tutti i valori di un array

```
array array_count_values (array input)
```

array_count_values() restituisce un array che ha i valori dell'array *input* per chiavi e la loro frequenza in *input* come valori.

Esempio 1. Esempio di array_count_values()

```
$array = array (1, "ciao", 1, "mondo", "ciao");
array_count_values ($array); // restitu-
isce l'array (1=>2, "ciao"=>2, "mondo"=>1)
```

array_diff (PHP 4)

Calcola la differenza di due o più array

```
array array_diff (array array1, array array2 [, array ...])
```

array_diff() restituisce un array contenente tutti i valori di *array1* che non sono presenti in alcuno degli altri array. Si noti che le associazioni con le chiavi vengono mantenute.

Esempio 1. Esempio di array_diff()

```
$array1 = array ("a" => "verde", "rosso", "blu", "rosso");
$array2 = array ("b" => "verde", "giallo", "rosso");
$risultato = array_diff ($array1, $array2);
```

In questo modo `$risultato` sarà `array ("blue");`. Occorrenze multiple in $array1 sono tutte trattate nello stesso modo.

> **Nota:** Due elementi sono considerati uguali se e solo se `(string) $elem1 === (string) $elem2`. Ovvero: quando la rappresentazione sotto forma di stringa è la stessa.

> **Attenzione**
>
> Questa funzione era errata nel PHP 4.0.4!

Vedere anche array_intersect().

array_filter (PHP 4 >= 4.0.6)

Filtra gli elementi di un array usando una funzione callback

```
array array_filter (array input [, mixed callback])
```

array_filter() restituisce un array contenente tutti gli elementi di *input* filtrati attraverso una funzione callback. Se *input* è un array associativo le chiavi sono mantenute.

Esempio 1. Esempio di array_filter()

```
function dispari($var) {
```

```
   return ($var % 2 == 1);
}

function pari($var) {
   return ($var % 2 == 0);
}

$array1 = array ("a"=>1, "b"=>2, "c"=>3, "d"=>4, "e"=>5);
$array2 = array (6, 7, 8, 9, 10, 11, 12);

$arr_dispari = array_filter($array1, "dispari");
$arr_pari = array_filter($array2, "pari");
```

In questo modo `$arr_dispari` sarà `array ("a"=>1, "c"=>3, "e"=>5);`, e `$arr_pari` sarà `array (6, 8, 10, 12);`,

Vedere anche array_map(), array_reduce().

array_flip (PHP 4 >= 4.0.0)

Scambia tutti i valori di un array

```
array array_flip (array trans)
```

array_flip() restituisce un array scambiato, ovvero le chiavi di *`trans`* diventano valori e i valori di *`trans`* diventano chiavi.

Si noti che i valori di *`trans`* devono poter diventare chiavi valide, ovvero devo essere di tipo integer o string. Un errore verrà segnalato se un valore ha il tipo errato, e la coppia chiave/valore in questione *non verrà scambiata.*

Se un valore ha più di una occorrenza, L'ultima chiave verrà usata come valore, e tutte le altre verranno perse.

array_flip() restituisce `FALSE` se fallisce.

Esempio 1. Esempio di array_flip()

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

Esempio 2. Esempio di array_flip(): collisione

```
$trans = array ("a" => 1, "b" => 1, "c" => 2);
$trans = array_flip ($trans);
// ora $trans è : array(1 => "b", 2 => "c");
```

array_intersect (PHP 4)

Calcola l'intersezione degli arrays

```
array array_intersect (array array1, array array2 [, array ...])
```

array_intersect() restituisce un array contenente tutti i valori di *array1* che siano presenti in tutti gli array passati come argomento. Si noti che le associazioni con le chiavi sono mantenute.

Esempio 1. Esempio di array_intersect()

```
$array1 = array ("a" => "verde", "rosso", "blu");
$array2 = array ("b" => "verde", "giallo", "rosso");
$risultato = array_intersect ($array1, $array2);
```

In questo modo `$result` sarà `array ("a" => "verde", "rosso");`

> **Nota:** Due elementi sono considerati uguali solo e solo se `(string) $elem1 === (string) $elem2`. Ovvero: quando la rappresentazione sotto forma di stringa è la stessa.

> **Attenzione**
>
> Questa funzione era errata nel PHP 4.0.4!

Vedere anche array_diff().

array_keys (PHP 4 >= 4.0.0)

Restituisce tutte le chiavi di un array

```
array array_keys (array input [, mixed search_value])
```

array_keys() rstituisce le chiavi, numeriche e stringa, dell'array *input*.

Se il parametro opzionale *search_value* è specificato, solo le chiavi con che corrispondono a quel valore vengono restituite. Altrimenti, vengono restituite tutte le chiavi dell'array *input*.

Esempio 1. Esempio di array_keys()

```
$array = array (0 => 100, "colore" => "rosso");
array_keys ($array);       // restituisce array (0, "colore")

$array = array ("blu", "rosso", "verde", "blu", "blu");
array_keys ($array, "blu");  //  restituisce array (0, 3, 4)

$array = array ("colore" => ar-
ray("blu", "rosso", "verde"), "misura" => array("piccola", "me-
dia", "grande"));
array_keys ($array);  //  restituisce array ("colore", "misura")
```

Nota: Questa funzione è stata aggiunta in PHP 4, qui sotto c'è una implementazione per coloro che usano ancora PHP 3.

Esempio 2. Implementazione di array_keys() per utenti PHP 3

```
function array_keys ($arr, $term="") {
    $t = array();
    while (list($k,$v) = each($arr)) {
        if ($term && $v != $term) {
            continue;
            $t[] = $k;
        }
        return $t;
    }
}
```

Vedere anche array_values().

array_map (PHP 4 >= 4.0.6)

Applica la funzione callback a tutti gli elementi dell'array dato

```
array array_map (mixed callback, array arr1 [, array arr2...])
```

array_map() restituisce un array contenente tutti gli elementi di *arr1* dopo che è stata loro applicata la funzione callback. Il numero di parametri che la funzione callback accetta deve corrispondere al numero di array passati alla funzione **array_map()**

Esempio 1. Esempio di array_map()

```
function cubo($n) {
   return $n*$n*$n;
}

$a = array(1, 2, 3, 4, 5);
$b = array_map("cubo", $a);
```

L'array `$b` conterrà `array (1, 8, 27, 64, 125);`

Esempio 2. array_map() - usare più array

```
function mostra_Spagnolo($n, $m) {
   return "Il numero $n si dice $m in Spagnolo";
}

function mappa_Spagnolo($n, $m) {
   return array ($n => $m);
}

$a = array(1, 2, 3, 4, 5);
$b = array("uno", "dos", "tres", "cuatro", "cinco");

$c = array_map("mostra_Spagnolo", $a, $b);

print_r($c);

// darà:
// Array
// (
//     [0] => Il numero 1 si dice uno in Spagnolo
//     [1] => Il numero 2 si dice dos in Spagnolo
//     [2] => Il numero 3 si dice tres in Spagnolo
//     [3] => Il numero 4 si dice cuatro in Spagnolo
//     [4] => Il numero 5 si dice cinco in Spagnolo
// )

$d = array_map("mappa_Spagnolo", $a , $b);

print_r($d);
```

```
// darà:
// Array
// (
//     [0] => Array
//         (
//             [1] => uno
//         )
//
//     [1] => Array
//         (
//             [2] => dos
//         )
//
//     [2] => Array
//         (
//             [3] => tres
//         )
//
//     [3] => Array
//         (
//             [4] => cuatro
//         )
//
//     [4] => Array
//         (
//             [5] => cinco
//         )
//
// )
```

Generalmente, quando si usano due o più array, questi devono avere eguale lunghezza in quanto la funzione callback viene applicata in parallelo agli elementi corrispondenti. Se gli array sono di lunghezza diversa, il più corto verrà esteso con elementi vuoti.

Un uso interessante di questa funzione è quello di costruire un array di array, cosa che pouò essere facilmente ottenuta usando `null` come nome della funzione callback

Esempio 3. Array_map() - creare un array di array

```
$a = array(1, 2, 3, 4, 5);
$b = array("uno", "due", "tre", "quattro", "cinque");
$c = array("uno", "dos", "tres", "cuatro", "cinco");

$d = array_map(null, $a, $b, $c);
print_r($d);

// darà:
```

```
// Array
// (
//     [0] => Array
//         (
//             [0] => 1
//             [1] => uno
//             [2] => uno
//         )
//
//     [1] => Array
//         (
//             [0] => 2
//             [1] => due
//             [2] => dos
//         )
//
//     [2] => Array
//         (
//             [0] => 3
//             [1] => tre
//             [2] => tres
//         )
//
//     [3] => Array
//         (
//             [0] => 4
//             [1] => quattro
//             [2] => cuatro
//         )
//
//     [4] => Array
//         (
//             [0] => 5
//             [1] => cinque
//             [2] => cinco
//         )
//
// )
```

Vedere anche array_filter(), array_reduce().

array_merge (PHP 4 >= 4.0.0)

Fonde due o più array

```
array array_merge (array array1, array array2 [, array ...])
```

array_merge() fonde gli elementi di due o più array in modo che i valori di un array siano accodati a quelli dell'array precedente. Restituisce l'array risultante.

Se gli array in input hanno le stesse chiavi stringa, l'ultimo valore di quella chiave sovrascriverà i precedenti. Se gli array hanno le stess chiavi numeriche, l'ultimo valore non sovrascriverà quello originale, bensì sarà accodato.

Esempio 1. Esempio di array_merge()

```
$array1 = array ("colore" => "rosso", 2, 4);
$array2 = array ("a", "b", "colore" => "verde", "forma" => "trapezio", 4);
array_merge ($array1, $array2);
```

L'array risultante sarà `array("colore" => "verde", 2, 4, "a", "b", "forma" => "trapezio", 4)`.

Vedere anche array_merge_recursive().

array_merge_recursive (PHP 4)

Fonde due o più array in modo ricorsivo

```
array array_merge_recursive (array array1, array array2 [, array ...])
```

Array_merge_recursive() fonde gli elementi di due o più array in modo tale che i valori di un array siano accodati all'array precedente. Restituisce l'array risultante.

Se gli array in input hanno le stesse chiavi stringa, i valori di queste chiavi vengono fusi in un array, e questo è fatto in modo ricorsivo, cio‘ se uno dei valori è un array, la funzione lo fonder%agrave; con una voce corrispondente in un altro array Comunque, se gli array hanno la stessa chiave numerica, l'ultimo valore non sovrascriver‘ il valore originale, bensì verrà accodato.

Esempio 1. Esempio di array_merge_recursive()

```
$ar1 = array ("colore" => array ("preferito" => "rosso"), 5);
$ar2 = array (10, "colore" => array ("preferito" => "verde", "blu"));
$result = array_merge_recursive ($ar1, $ar2);
```

L’array risultante sarà `array ("colore" => array ("preferito" => array ("rosso", "verde"), "blu"), 5, 10)`.

Vedere anche array_merge().

array_multisort (PHP 4 >= 4.0.0)

Ordina array multipli o multidimensionali

```
bool array_multisort (array ar1 [, mixed arg [, mixed ... [, array ...]]])
```

Array_multisort() Può essere usata per ordinare parecchi array allo stesso tempo, oppure un array multidimensionale, rispetto a una o più dimensioni. Mantiene le associazioni delle chiavi durante l’ordinamento.

Gli array in input sono trattati come campi di una tabella che vengano ordinati per righe - questo assomiglia alla funzionalità della clausola SQL ORDER BY Il primo array è quello primario, rispetto a cui aordinare. Le righe (valori) in questo array that siano uguali vengono ordinate secondo l’array successivo, e così via.

La struttura degli argomenti di questa funzione è un po’ inusuale, ma flessibile. Il primo argomento deve essere un array. In seguito, ogni argomento può essere sia un aray che un flag di ordinamento, selezionabile dalla seguente lista.

Flag di ordinamento:

- SORT_ASC - ordinamento crescente
- SORT_DESC - ordinamento decrescente

Sorting type flags:

- SORT_REGULAR - confronta gli elementi in modo normale
- SORT_NUMERIC - confronta gli elementi numericamente
- SORT_STRING - confronta gli elementi come stringhe

Dopo ogni array, non si possono specificare due flag dello stesso tipo. I flag specificati dopo un array si applicano solo a quell’array - sono reimpostati ai default SORT_ASC e SORT_REGULAR dopo ogni nuovo array passato come argomento.

Restituisce `TRUE` in caso di successo, `FALSE` altrimenti.

Esempio 1. Ordinamre più array

```
$ar1 = array ("10", 100, 100, "a");
```

```
$ar2 = array (1, 3, "2", 1);
array_multisort ($ar1, $ar2);
```

In questo esempio, dopo l'ordinamento, il primo array conterrà "10", "a", 100, 100. Il secondo array conterrà 1, 1, "2", 3. Gli elementi nel secondo array che corrispondono agli elementi identici nel primo array (100 e 100) vengono pure ordinati.

Esempio 2. Ordinare un array multi-dimensionale

```
$ar = array (array ("10", 100, 100, "a"), array (1, 3, "2", 1));
array_multisort ($ar[0], SORT_ASC, SORT_STRING,
                 $ar[1], SORT_NUMERIC, SORT_DESC);
```

In questo esempio, dopo l'ordinamento, il primo array conterrà 10, 100, 100, "a" (ordinato come stringhe ordine crescente), e il secondo conterrà 1, 3, "2", 1 (ordinati come numeri, in ordine decrescente).

array_pad (PHP 4 >= 4.0.0)

Riempie con un valore un array fino alla lunghezza specificata

```
array array_pad (array input, int pad_size, mixed pad_value)
```

array_pad() restituisce una copia di *input* allungato alla dimensione sepcificata da *pad_size* con il valore *pad_value*. Se *pad_size* è positivo l'array è riempito sulla destra, se è negativo sulla sinistra. Se il valore assoluto di *pad_size* è minore o uguale alla lunghezza di *input* non viene effettuata alcuna modifica.

Esempio 1. esempio di array_pad()

```
$input = array (12, 10, 9);

$risultato = array_pad ($input, 5, 0);
// risultato diventa array (12, 10, 9, 0, 0)

$risultato = array_pad ($input, -7, -1);
// risultato diventa array (-1, -1, -1, -1, 12, 10, 9)

$risultato = array_pad ($input, 2, "noop");
// ridimensionamento non efettuato
```

array_pop (PHP 4 >= 4.0.0)

Estrae l'elemento alla fine dell'array

```
mixed array_pop (array array)
```

array_pop() estrae e restituisce l'ultimo valore di *array*, accorciando *array* di un elemento. Se *array* è vuoto (o non è un array), viene restituito NULL.

Esempio 1. esempio di array_pop()

```
$pila = array ("arancia", "mela", "lampone");
$frutto = array_pop ($pila);
```

Dopo questa operazione, $pila ha solo 2 elementi: "arancia" e "mela", e $frutto contiene "lampone".

Vedere anche array_push(), array_shift(), e array_unshift().

array_push (PHP 4 >= 4.0.0)

Accoda uno o più elementi ad un array

```
int array_push (array array, mixed var [, mixed ...])
```

array_push() tratta *array* come una pila, e accoda le variabili date alla fine di *array*. La lunghezza di *array* aumenta del numero di variabili accodate. Ha lo steso effetto di :

```
$array[] = $var;
```

ripetuto per ogni *var*.

Restituisce il nuovo numero di elementi nell'array.

Esempio 1. esempio di array_push()

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

Questo esempio fa sì che `$pila` abbia 4 elementi: 1, 2, "+", e 3.

Vedere anche: array_pop(), array_shift(), e array_unshift().

array_rand (PHP 4 >= 4.0.0)

Estrae a caso uno o più elementi da un array

```
mixed array_rand (array input [, int num_req])
```

array_rand() è piuttosto utile quando si vuole estrarre a caso uno o più elementi da un array. Prende un array (*`input`*) e un argomento ozpionale (*`num_req`*) che specifica quanti elementi estrarre - se non è specificato, è 1 per default.

Se si sta estraendo solo un elemento, **array_rand()** restituisce la chiave di un elemento. Altrimenti, restituisce un array di chiavi. Questo viene fatto in modo da permettere di estrarre dall'array sia le chiavi che i valori.

Non dimenticare di chiamare srand() per perturbare il generatore di numri casuali.

Esempio 1. esempio di array_rand()

```
srand ((float) microtime() * 10000000);
$input = array ("Neo", "Morpheus", "Trinity", "Cypher", "Tank");
$chiavi = array_rand ($input, 2);
print $input[$chiavi[0]]."\n";
print $input[$chiavi[1]]."\n";
```

array_reverse (PHP 4 >= 4.0.0)

Restituisce un array con gli elementi in ordine invertito

```
array array_reverse (array array [, bool mantieni_chiavi])
```

array_reverse() prende *`array`* e restituisce un nuovo array con l'ordine degli elementi invertito, mantenendo le chiavi sie *`mantieni_chiavi`* è `TRUE`.

Esempio 1. esempio di array_reverse()

```
$input = array ("php", 4.0, array ("verde", "rosso"));
```

```
$risultato = array_reverse ($input);
$resultato_chiavi = array_reverse ($input, TRUE);
```

In questo modo sia `$result` che `$result_keyed` sono `array(array ("verde", "rosso"), 4.0, "php")`. Ma `$risultato_chiavi[0]` è sempre `"php"`.

> **Nota:** Il secondo parametro è stato aggiunto in PHP 4.0.3.

array_reduce (PHP 4 >= 4.0.5)

Riduce iterativamente l'array a un singolo valore utilizzando una funzione callback

```
mixed array_reduce (array input, mixed callback [, int initial])
```

array_reduce() applica iterativamente la funzione *`callback`* agli elementi dell'array *`input`*, riducendo l'array a un singolo valore. Seil parametro opzionale *`intial`* è specificato, viene usato come valore iniziale all'inizio del processo, o come risultato finale nel caso l'array sia vuoto.

Esempio 1. esempio di array_reduce()

```
function rsum($v, $w) {
   $v += $w;
  return $v;
}

function rmul($v, $w) {
   $v *= $w;
  return $v;
}

$a = array(1, 2, 3, 4, 5);
$x = array();
$b = array_reduce($a, "rsum");
$c = array_reduce($a, "rmul", 10);
$d = array_reduce($x, "rsum", 1);
```

In questo modo `$b` conterrà `15`, `$c` conterrà `1200` (= 1*2*3*4*5*10), e `$d` conterrà `1`.

Vedere anche array_filter(), array_map().

array_shift (PHP 4 >= 4.0.0)

Estrae l'elemento alla testa dell'array

```
mixed array_shift (array array)
```

array_shift() estrae il primo elemento di *array* e lo restituisce, accorciando *array* di un elemento e spostando tutti gli altri all'indietro. Se *array* è vuoto (o non è un array), viene restituito NULL.

Esempio 1. esempio di array_shift()

```
$args = array ("-v", "-f");
$opt = array_shift ($args);
```

Il risultato sarà $args contenente un elemento "-f", e $opt uguale a "-v".

Vedere anche array_unshift(), array_push(), e array_pop().

array_slice (PHP 4 >= 4.0.0)

Estrae un sottoinsieme da un array

```
array array_slice (array array, int offset [, int length])
```

array_slice() restituisce una sequenza di elementi di *array* specificati dai parametri *offset* e *length* .

Se *offset* è positivo, la sequenza comincerà da quell'offset in *array*. Se *offset* è negativo, la sequenza comincerà alla distanza *offset* dalla fine di *array*.

Se *length* è specificata ed è positiva, la sequenza conterrà quel numero di elementi. Se *length* è specificata ed è negativa la sequenza si fermerà a quel numero di elementi dalla fine dell'array. Se viene omessa, la sequenza conterrà tutto da *offset* fino alla fine di *array*.

Esempio 1. esempi di array_slice()

```
$input = array ("a", "b", "c", "d", "e");

$output = array_slice ($input, 2);      // restituisce "c", "d", e "e"
$output = array_slice ($input, 2, -1);  // restituisce "c", "d"
$output = array_slice ($input, -2, 1);  // restituisce "d"
```

```
$output = array_slice ($input, 0, 3);   // restituisce "a", "b", e "c"
```

Vedere anche array_splice().

array_splice (PHP 4 >= 4.0.0)

Rimuove una porzione dell'array e la sostituisce con altro

```
array array_splice (array input, int offset [, int length [, array
replacement]])
```

array_splice() rimuove gli elementi specificati da *offset* e *length* dall'array *input*, e li sostituisce con gli elementi dell'array *replacement*, se fornito.

Se *offset* è positivo l'inizio della porzione rimossa è a quella distanza dall'inizio dell'array *input*. Se *offset* è negativo inizia a quella distanza dalla fine dell'array *input*.

Se *length* è omessa, rimuove tutti gli elementi da *offset* alla fine dell'array. Se *length* è specificata a positiva, quel numero di elementi vengono rimossi. Se *length* è specificata e negativa la porzione da rimuovere terminerà a *length* elementi dalla fine dell'array. Suggerimento: per rimuovere tutti gli elementi tra *offset* e la fine dell'array quando è specificato pure *replacement*, usare `count($input)` nel parametro *length*.

Se l'array *replacement* è specificato, gli elementi rimossi sono sostituiti dagli elementi di questo array. Se *offset* e *length* sono tali per cui niente viene rimosso, gli elementi dell'array *replacement* sono inseriti nella posizione specificata da *offset*. Suggerimento: se *replacement* è composto solo da un elemento non è necessario porlo nel costrutto `array()`, a meno che l'elemento stesso non sia un array.

Valgono le seguenti equivalenze:

```
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, ar-
ray ($x, $y))
$a[$x] = $y                     array_splice ($input, $x, 1, $y)
```

Restituisce un array contenente gli elementi rimossi.

Esempio 1. esempi di array_splice()

```
$input = array ("rosso", "verde", "blu", "giallo");
array_splice ($input, 2);
// $input è ora array ("rosso", "verde")

$input = array ("rosso", "verde", "blu", "giallo");
array_splice ($input, 1, -1);
// $input è ora array ("rosso", "giallo")

$input = array ("rosso", "verde", "blu", "giallo");
array_splice ($input, 1, count($input), "arancio");
// $input è ora array ("rosso", "arancio")

$input = array ("rosso", "verde", "blu", "giallo");
array_splice ($input, -1, 1, array("nero", "marrone"));
// $input è ora array ("rosso", "verde",
//          "blu", "nero", "marrone")
```

Vedere anche array_slice().

array_sum (PHP 4 >= 4.0.4)

Calcola la somma dei valori di un array.

```
mixed array_sum (array arr)
```

array_sum() restituisce la somma dei valori dell'array sotto forma di integer o float.

Esempio 1. esempi di array_sum()

```
$a = array(2,4,6,8);
echo "sum(a) = ".array_sum($a)."\n";
// stampa: sum(a) = 20

$b = array("a"=>1.2,"b"=>2.3,"c"=>3.4);
echo "sum(b) = ".array_sum($b)."\n";
// stampa: sum(b) = 6.9
```

array_unique (PHP 4)

Rimuove i valori duplicati di un array

```
array array_unique (array array)
```

array_unique() prende *`array`* e restituisce un nuovo array senza i valori duplicati.

Si noti che le chiavi sono mantenute. **array_unique()** manterrà la prima chiave trovata per ogni valore, e ignorerà tutte le chiavi successive.

> **Nota:** Due elementi sono considerati uguali se e solo se `(string) $elem1 === (string) $elem2`. Ovvero: quando la rappresentazione sotto forma di stringa è la stessa.
>
> Verrà usato il primo elemento.

> **Attenzione**
>
> Questa funzione era errata nel PHP 4.0.4!

Esempio 1. esempio di array_unique()

```
$input = array ("a" => "verde", "rosso", "b" => "verde", "blu", "rosso");
$risultato = array_unique ($input);
print_r($result);
// il risultato sarà :
//Array
//(
//    [a] => verde
//    [0] => rosso
//    [1] => blu
//)
```

Esempio 2. array_unique() e i tipi

```
$input = array (4,"4","3",4,3,"3");
$risultato = array_unique ($input);
var_dump($risultato);

/* risultato:
array(2) {
   [0]=>
   int(4)
   [1]=>
   string(1) "3"
}
*/
```

array_unshift (PHP 4 >= 4.0.0)

Inserisce uno o più elementi all'inizio dell'array

```
int array_unshift (array array, mixed var [, mixed ...])
```

array_unshift() aggiunge gli elementi specificati in testa ad *array*. Si noti che la lista di elementi è aggiunta in blocco, in modo tale che gli elementi rimangano nello stesso ordine.

Restituisce il nuovo numero di elementi in *array*.

Esempio 1. esempio di array_unshift()

```
$lista = array ("p1", "p3");
array_unshift ($lista, "p4", "p5", "p6");
```

`$lista` conterrà 5 elementi: "p4", "p5", "p6", "p1", e "p3".

Vedere anche array_shift(), array_push(), e array_pop().

array_values (PHP 4 >= 4.0.0)

Restituisce tutti i valori di un array

```
array array_values (array input)
```

array_values() restituisce tutti i valori dell'array *input*.

Esempio 1. esempio di array_values()

```
$array = array ("taglia" => "XL", "colore" => "oro");
array_values ($array);    // restituisce l'array ("XL", "oro")
```

Nota: Questa funzione è stata aggiunta in PHP 4, qui sotto si trova una implementazione per chi usa ancora PHP 3.

Esempio 2. Implementazone di array_values() per gli utenti PHP 3

```
function array_values ($arr) {
    $t = array();
```

```
    while (list($k, $v) = each ($arr)) {
        $t[] = $v;
    }
    return $t;
}
```

array_walk (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Esegue una funzione su ogni elemento dell'array

```
int array_walk (array arr, string funzione [, mixed datiutente])
```

Esegue la funzione definita dall'utente identificata da *funzione* su ogni elemento di *arr*. A *funzione* verrà passato il valore dell'elemento come primo parametro e la chiave come secondo parametro. Se *datiutente* è specificato, verrà passato come terzo parametro alla funzione. *funzione* deve essere una funzione definita dall'utente, e non può essere una funzione nativa PHP. Quindi, non si può usare **array_walk()** direttamente con **str2lower()**, bensì occorre costruire una funzione utente con tale istruzione, e passarla come argomento.

Se *funzione* richiede più di due o tre argomenti, a seconda di *datiutente*, un warning verrà generato ogni qualvolta **array_walk()** chiami *funzione*. Questi warning possono essere soppressi apponendo il simbolo '@' alla chiamata di **array_walk()**, oppure usando error_reporting().

> **Nota:** Se *funzione* deve lavorare con i reali valori dell'array, specificare che il primo parametro di *funzione* deve essere passato come riferimento. A qesto punto ogni modifica a questi elementi verrà effettuata sull'array stesso.

> **Nota:** Il passaggio della chiave e di *datiutente* a *func* è stato aggiunto nella versione 4.0.
>
> In PHP 4 la funzione reset() deve essere chiamata obbligatoriamente, in quanto **array_walk()** non reinizializza automaticamente l'array.

Esempio 1. esempio di array_walk()

```
$frutta = ar-
ray ("d"=>"limone", "a"=>"arancia", "b"=>"banana", "c"=>"mela");

function modifica (&$elemento1, $chiave, $prefisso) {
    $elemento1 = "$prefisso: $elemento1";
}
```

```
function stampa ($elemento2, $chiave) {
    echo "$chiave. $elemento2<br>\n";
}

array_walk ($frutta, 'stampa');
reset ($frutta);
array_walk ($frutta, 'modifica', 'frutto');
reset ($frutta);
array_walk ($frutta, 'stampa');
```

Vedere anche each() e list().

arsort (PHP 3, PHP 4 >= 4.0.0)

Ordina un array in ordine decrescente e mantiene le associazioni degli indici

```
void arsort (array array [, int sort_flags])
```

Questa funzione ordina un array in modo tale che i suoi indici mantengano la loro correlazione con gli elementi ai quali sono associati. Viene usata principalmente nell'ordinamento degli array associativi, quando la disposizione originaria degli elementi è importante.

Esempio 1. esempio di arsort()

```
$frutta = ar-
ray ("d"=>"limone", "a"=>"arancia", "b"=>"banana", "c"=>"mela");
arsort ($frutta);
reset ($frutta);
while (list ($chiave, $valore) = each ($frutta)) {
    echo "$chiave = $valore\n";
}
```

Questo esempio mostrerà:

```
frutta[c] = mela
frutta[d] = limone
frutta[b] = banana
frutta[a] = arancia
```

I frutti sono ordinati in ordine alfabetico decrescente, e l'indice associato a ogni elemento è stato mantenuto.

È possibile modificare il comportamento dell'ordinamento usando il parametro opzionale *`sort_flags`*, per maggiori dettagli vedere sort().

vedere anche: asort(), rsort(), ksort(), e sort().

asort (PHP 3, PHP 4 >= 4.0.0)

Ordina un array e mantiene le associazioni degli indici

```
void asort (array array [, int sort_flags])
```

Questa funzione ordina un array in modo tale che i suoi indici mantengano la loro correlazione con gli elementi ai quali sono associati. Viene usata principalmente nell'ordinamento degli array associativi, quando la disposizione originaria degli elementi è importante .

Esempio 1. esempio di asort()

```
$frutta = ar-
ray ("d"=>"limone", "a"=>"arancia", "b"=>"banana", "c"=>"mela");
asort ($frutta);
reset ($frutta);
while (list ($chiave, $valore) = each ($frutta)) {
    echo "$chiave = $valore\n";
}
```

Questo esempio mostrerà:

```
fruits[a] = arancia
fruits[b] = banana
fruits[d] = limone
fruits[c] = mela
```

I frutti sono ordinati in ordine alfabetico , e l'indice associato ad ogni elemento è stato mantenuto.

È possibile modificare il comportamento dell'ordinamento usando il parametro opzionale *`sort_flags`*, per maggiori dettagli vedere sort().

Vedere anche arsort(), rsort(), ksort(), e sort().

compact (PHP 4 >= 4.0.0)

Crea un array contenente variabili e il loro valore

```
array compact (mixed varname [, mixed ...])
```

compact() accetta un numero variabile di parametri. Ogni parametro può essere una stringa contenente il nome della variabile, o un array di nomi di variabile. L'array può contenere altri array di nomi di variabile; **compact()** se ne occupa in modo ricorsivo.

Per ognuno di questi, **compact()** cerca la variabile con quel nome nella tabella dei simboli corrente, e la aggiunge all'array di output in modo tale che il nome della variabile diventi la chiave e i contenuti della variabile diventino il valore associato a quella chiave. In breve, **compact()** è l'opposto di extract(). Restituisce l'array di output con tutte le variabili aggiunte a quest'ultimo.

Qualsiasi stringa non valorizzata verrà semplicemente ignorata.

Esempio 1. esempio di compact()

```
$citta = "Milano";
$provincia = "MI";
$evento = "SMAU";

$var_luoghi = array ("citta", "provincia");

$risultato = compact ("evento", "niente", $var_luoghi);
```

In questo modo, `$risultato` sarà `array ("evento" => "SMAU", "citta" => "Milano", "provincia" => "MI")`.

Vedere anche extract().

count (PHP 3, PHP 4 >= 4.0.0)

Conta gli elementi in una variabile

```
int count (mixed var)
```

Restituisce il numero di elementi in *`var`*, la quale è di norma un array (dal momento che qualsiasi altro oggetto avrà un elemento).

Se *`var`* non è un array, verrà restituito `1` (eccezione: `count(NULL)` restituisce `0`).

> **Attenzione**
>
> **count()** può restituire 0 per una variabile che non è impostata, ma può anche restituire 0 per una variabile che è stata inizializzata con un array vuoto. Usare isset() per verificare se una variabile è impostata.

Vedere la sezione Arrays nel manuale per una spiegazione dettagliata di come gli array siano implementati ed usati in PHP.

Esempio 1. esempio di count()

```
$a[0] = 1;
$a[1] = 3;
$a[2] = 5;
$risultato = count ($a);
//$risultato == 3

$b[0] = 7;
$b[5] = 9;
$b[10] = 11;
$risultato = count ($b);
// $result == 3;
```

Nota: La funzione sizeof() è un alias per **count()**.

Vedere anche: sizeof(), isset(), e is_array().

current (PHP 3, PHP 4 >= 4.0.0)

Restituisce l’elemento corrente di un array

```
mixed current (array array)
```

Ogni array ha un puntatore interno all’elemento "corrente", che è inizializzato al primo elemento inserito nell’array.

La funzione **current()** restituisce l’elemento che è attualmente puntato dal puntatore interno. In ogni caso non muove il puntatore. Se il puntatore interno punta oltre la fine della lista di elementi, **current()** restituisce FALSE.

> **Attenzione**
>
> Se l'array contiene elementi vuoti (0 o "", la stringa vuota) la funzione restituirà FALSE pure per questi elementi. Questo rende impossibile stabilire se si è veramente alla fine della lista in un array di questo tipo usando **current()**. Per attraversare in modo corretto un array che può contenere elementi vuoti, usare la funzione each().

Vedere anche: end(), next(), prev(), e reset().

each (PHP 3, PHP 4 >= 4.0.0)

Restituisce la successiva coppia chiave/valore di un array e incrementa il puntatore dell'array

```
array each (array array)
```

Restituisce la corrente coppia chiave/valore corrente di *array* e incrementa il puntatore interno dell'array. Questa coppia è restituita in un array di quattro elementi, con le chiavi *0*, *1*, *key*, and *value*. Gli elementi *0* e *key* contengono il nome della chiave dell'elemento dell'array, mentre *1* e *value* contengono i dati.

Se il puntatore interno dell'array punta oltre la fine dei contenuti dell'array, **each()** restituisce FALSE.

Esempio 1. esempi dieach()

```
$foo = array ("bob", "fred", "jussi", "jouni", "egon", "marliese");
$bar = each ($foo);
```

$bar ora contiene la seguente coppia chiave/valore:

- 0 => 0
- 1 => 'bob'
- key => 0
- value => 'bob'

```
$foo = array ("Robert" => "Bob", "Seppo" => "Sepi");
$bar = each ($foo);
```

$bar ora contiene la seguente coppia chiave/valore:

- 0 => 'Robert'
- 1 => 'Bob'
- key => 'Robert'

- value => ’Bob’

each() viene normalmente usata in congiunzione con list() nell’attraversamento di un array; per esempio, `$HTTP_POST_VARS`:

Esempio 2. Attraversamento di `$HTTP_POST_VARS` con each()

```
echo "Valori inviati con il metodo POST:<br>";
reset ($HTTP_POST_VARS);
while (list ($chiave, $valore) = each ($HTTP_POST_VARS)) {
    echo "$chiave => $valore<br>";
}
```

Dopo l’esecuzione di **each()**, il puntatore dell’array viene lasciato sull’elemento successivo, o sull’ultimo elemento se si è alla fine dell’array.

Vedere anche key(), list(), current(), reset(), next(), e prev().

end (PHP 3, PHP 4 >= 4.0.0)

Sposta il puntatore interno dell’array all’ultimo elemento

```
mixed end (array array)
```

end() fa avanzare il puntatore di *`array`* all’ultimo elemento, e restituisce l’elemento.

Vedere anche: current(), each(), **end()**, next(), e reset().

extract (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Importa le variabili nella tabella dei simboli

```
int extract (array var_array [, int extract_type [, string prefix]])
```

Questa funzione viene usata per importare delle variabili da un array nella tabella dei simbloi corrente. Riceve un array associativo *`var_array`* e interpreta le chiavi come nomi di variabile e i valori come valori di variabile. Per ogni coppia chiave/valore verrà creata una variabile nella tabella dei simboli corrente, coerentemente con i parametri *`extract_type`* e *`prefix`*.

Nota: Since version 4.0.5 this function returns the number of variables extracted.

extract() controlla ogni chiave per stabilire se costituisce un nome valido di variabile e se ci sono collisioni con variabili già esistenti nella tabella dei simboli. Il modo in cui vengono trattate le chiavi invalide/numeriche e le collisioni è determinato da *`extract_type`*. Può essere uno dei seguenti valori:

EXTR_OVERWRITE

Se avviene una collisione, sovrascrive la variabile esistente.

EXTR_SKIP

Se avviene una collisione, non sovrascrive la variabile esistente.

EXTR_PREFIX_SAME

Se avviene una collisione, mette come prefisso al nome della variabile il parametro *`prefix`*.

EXTR_PREFIX_ALL

Mette come prefisso di tutte le variabili il parametro *`prefix`*. Dal PHP 4.0.5 questo avviene anche per i valori numerici.

EXTR_PREFIX_INVALID

Mette come prefisso, solo per i nomi di variabili invalidi/numerici, il paramentro *`prefix`*. Questa opzione è stata aggiunta in PHP 4.0.5.

Se *`extract_type`* non è specificato, si assume che sia EXTR_OVERWRITE.

Si noti che *`prefix`* è richiesto solo se *`extract_type`* è EXTR_PREFIX_SAME, EXTR_PREFIX_ALL, o EXTR_PREFIX_INVALID. Se il risultato non è un nome di variabile valido, non viene importato nella tabella dei simboli.

extract() restituisce il numero di variabili importate con successo nella tabella dei simboli.

Un possibile uso di **extract()** è quello di importare nella tabella dei simboli variabili contenute in un array associativo restituito da wddx_deserialize().

Esempio 1. esempio diextract()

```
<?php

/* Si supponga che $array_variabili sia un array restituito da
   wddx_deserialize */

$dimensione = "grande";
$array_variabili = array ("colore" => "blu",
                    "dimensione"  => "media",
```

```
                    "forma" => "sfera");
extract ($array_variabili, EXTR_PREFIX_SAME, "wddx");

print "$colore, $dimensione, $forma, $wddx_dimensione\n";

?>
```

Questo esempio mostrerà:

```
blu, grande, sfera, media
```

La variabile `$dimensione` non è stata sovrascritta, in quanto è specificato EXTR_PREFIX_SAME, che ha portato alla creazione di `$wddx_dimensione`. Se fosse stato specificato EXTR_SKIP, $wddx_dimensione non sarebbe stata creata. EXTR_OVERWRITE avrebbe portato `$dimensione` ad assumere il valore "medio", e EXTR_PREFIX_ALL avrebbe fatto creare nuove variabili chiamate `$wddx_colore`, `$wddx_dimensione`, e `$wddx_forma`.

Si deve usare un array associativo, un array indicizzato numericamente non produce risultati.

Vedere anche: compact().

in_array (PHP 4 >= 4.0.0)

Restituisce `TRUE` se un valore è presente in un array

```
bool in_array (mixed ago, array pagliaio [, bool strict])
```

Cerca in *`pagliaio`* per trovare *`ago`* e restituisce `TRUE` se viene trovato nell'array, `FALSE` altrimenti.

Se il terzo parametri *`strict`* è `TRUE` la funzione **in_array()** controllerà anche il tipo di *`ago`* nell'array *`haystack`*.

Esempio 1. esempio di in_array()

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

Esempio 2. esempio di in_array() con strict

```
<?php
$a = array('1.10', 12.4, 1.13);

if (in_array('12.4', $a, TRUE))
    echo "'12.4' trovato con controllo strict\n";
if (in_array(1.13, $a, TRUE))
    echo "1.13 trovato con controllo strict\n";
?>

// Questo ritornerà:

1.13 trovato con controllo strict
```

Vedere anche array_search().

array_search (PHP 4 >= 4.0.5)

Ricerca un dato valore in un array e ne restituisce la chiave corrispondente, se la ricerca ha successo.

```
mixed array_search (mixed ago, array pagliaio [, bool strict])
```

Cerca in *pagliaio* per trovare *ago* e restituisce la chiave se viene trovato nell'array, FALSE altrimenti.

Se il terzo parametro opzionale *strict* è impostato a TRUE la funzione **array_search()** controllerè anche il tipo di *ago* nell'array *pagliaio*.

Vedere anche in_array().

key (PHP 3, PHP 4 >= 4.0.0)

Estrae la chiave corrente da un array associativo

```
mixed key (array array)
```

key() restituisce la chiave corrispondente all'attuale posizione del puntatore interno all'array.

Vedere anche current() e next().

krsort (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Ordina rispetto alle chiavi di un array in ordine inverso

```
int krsort (array array [, int sort_flags])
```

Ordina un array rispetto alle sue chiavi, in ordine inverso, mantenendo le associazioni. Questa funzione è utile con gli array associativi.

Esempio 1. Esempio di krsort()

```
$frutti = ar-
ray ("d"=>"limone", "a"=>"arancio", "b"=>"banana", "c"=>"mela");
krsort ($frutti);
reset ($frutti);
while (list ($chiave, $valore) = each ($frutti)) {
    echo "$chiave -> $valore\n";
}
```

Questo esempio mostrerà:

```
d = limone
c = mela
b = banana
a = arancio
```

Si può modificare il comportamento dell'ordinamento usando il parametro opzionale *sort_flags*, per ulteriori dettagli vedere sort().

Vedere anche asort(), arsort(), ksort() sort(), natsort() e rsort().

ksort (PHP 3, PHP 4 >= 4.0.0)

Ordina rispetto alle chiavi di un array

```
int ksort (array array [, int sort_flags])
```

Ordina un array rispetto alle sue chiavi, mantenendo le associazioni. Questa funzione è utile con gli array associativi.

Esempio 1. esempio di ksort()

```
$frutti = ar-
ray ("d"=>"limone", "a"=>"arancia", "b"=>"banana", "c"=>"mela");
ksort ($frutti);
reset ($frutti);
while (list ($chiave, $valore) = each ($frutti)) {
    echo "$chiave = $valore\n";
}
```

Questo esempio mostrerà:

```
a = arancia
b = banana
c = mela
d = limone
```

Si può modificare il comportamento dell'ordinamento usando il parametro opzionale *`sort_flags`*, per ulteriori dettagli vedere sort().

Vedere anche asort(), arsort(), sort(), natsort(), e rsort().

> **Nota:** Il secondo parametro è stato aggiunto in PHP 4.

list (unknown)

Assegna valori a delle variabili come se fossero un array

```
void list (...)
```

Come array(), questa non è in realtà una funzione, bensì un costrutto del linguaggio. **list()** è usata per assegnare valori ad una lista di variabili in una sola operazione.

Esempio 1. esempio di list()

```
<table>
 <tr>
  <th>Nome dell'impiegato</th>
  <th>Stipendio</th>
 </tr>

<?php
```

```
$risultato = mysql_query ($conn, "SELECT id, nome, stipen-
dio FROM impiegati");
while (list ($id, $nome, $stipendio) = mysql_fetch_row ($risultato)) {
    print (" <tr>\n".
           "  <td><a href=\"info.php3?id=$id\">$nome</a></td>\n".
           "  <td>$stipendio</td>\n".
           " </tr>\n");
}

?>

</table>
```

Vedere anche each() e array().

natsort (PHP 4 >= 4.0.0)

Ordina un array usando un algoritmo di "ordine naturale"

```
void natsort (array array)
```

Questa funzione implementa un algoritmo di ordinamento che ordina le stringhe alfanumeriche come lo farebbe un essere umano. Questo è chiamato "ordine naturale". un esempio della differenza tra questo algoritmo e quello normalmente usato dai computer (usato in sort()) è dato qui sotto:

Esempio 1. esempio di natsort()

```
$array1 = $array2 = array ("img12.png","img10.png","img2.png","img1.png");

sort($array1);
echo "Ordinamento standard\n";
print_r($array1);

natsort($array2);
echo "\nOrdinamento naturale\n";
print_r($array2);
```

Questo codice genererà il seguente risultato:

```
Ordinamento standard
```

```
Array
(
    [0] => img1.png
    [1] => img10.png
    [2] => img12.png
    [3] => img2.png
)

Ordinamento naturale
Array
(
    [3] => img1.png
    [2] => img2.png
    [1] => img10.png
    [0] => img12.png
)
```

Per ulteriori informazioni vedere la pagina di Martin Pool Natural Order String Comparison (http://www.linuxcare.com.au/projects/natsort/) .

Vedere anche natcasesort(), strnatcmp() e strnatcasecmp().

natcasesort (PHP 4 >= 4.0.0)

Ordina un array usando un algoritmo di "ordine naturale" non sensibile alle maiuscole/minuscole

```
void natcasesort (array array)
```

Questa funziona implementa un algoritmo di ordinamento che ordina le stringhe alfanumeriche come lo farebbe un essere umano. Questo è chiamato "ordine naturale".

natcasesort() è una versione, non sensibile alle maiuscole/minuscole, di natsort(). Vedere natsort() per un esempio della differenza tra questo algoritmo e quello normalmente usato dai computer.

Per maggiori informazioni vedere la pagina di Martin Pool Natural Order String Comparison (http://www.linuxcare.com.au/projects/natsort/) .

Vedere anche sort(), natsort(), strnatcmp() e strnatcasecmp().

next (PHP 3, PHP 4 >= 4.0.0)

Incrementa il puntatore interno dell’array

```
mixed next (array array)
```

Restituisce l'elemento dell'array che sta nella posizione successiva a quella attuale indicata dal puntatore interno, oppure FALSE se non ci sono altri elementi.

next() si comporta come current(), con una differenza. Incrementa il puntatore interno dell'array di una posizione, prima di restituire l'elemento. Ciò significa che restituisce l'elemento successivo e incrementa il puntatore di una posizione. Se l'incremento fa sì che il puntatore vada oltre la fine della lista di elementi, **next()** restituisce FALSE.

> **Attenzione**
>
> Se l'array contiene elementi vuoti, o elementi che hanno il valore chiave uguale a 0 allora questa funzione restituisce FALSE anche per questi elementi. Per esplorare correttamente un array che può contenere elementi vuoti o con chiave uguale a 0 vedere la funzione each().

Vedere anche: current(), end(), prev(), e reset().

pos (PHP 3, PHP 4 >= 4.0.0)

Restituisce l'elemento corrente di un array

```
mixed pos (array array)
```

Questo è un alias di current().

Vedere anche: end(), next(), prev() e reset().

prev (PHP 3, PHP 4 >= 4.0.0)

Decrementa il puntatore interno dell'array

```
mixed prev (array array)
```

Restituisce l'elemento dell'array che sta nella posizione precedente a quella attuale indicata dal puntatore interno, oppure FALSE se non ci sono altri elementi.

> **Attenzione**
>
> Se l'array contiene degli elementi vuoti la funzione restituirà FALSE per questi valori. Per esplorare correttamente un array che può contenere elementi vuoti vedere la funzione each().

prev() si comporta come next(), tranne per il fatto di decrementare il puntatore interno di una posizione, invece che incrementarlo.

Vedere anche: current(), end(), next(), e reset().

range (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Crea un array contenente una serie di elementi

```
array range (mixed min, mixed max)
```

range() restituisce una serie di elementi da *min* a *max*, inclusiva. Se *min* > *max*, la sequenza sarà decrescente.

Esempio 1. esempi di range()

```
foreach(range(0,9) as $numero) {
echo $numero;
}
foreach(range('a','z') as $lettera) {
echo $lettera;
}
foreach(range('z','a') as $lettera) {
echo $lettera;
}
```

> **Nota:** Prima della versione 4.0.7 la funzione range() generava solo array di interi crescenti. Il supporto per le sequenze di caratteri e per le sequenza decrescenti è stato aggiunto nella 4.0.7.

Vedere shuffle() per un altro esempio d'uso.

reset (PHP 3, PHP 4 >= 4.0.0)

Reimposta il puntatore interno di un array sulla posizione iniziale

```
mixed reset (array array)
```

reset() riporta il puntatore di *array* sul primo elemento.

reset() restituisce il volore del primo elemento dell'array.

Vedere anche: current(), each(), next(), e prev().

rsort (PHP 3, PHP 4 >= 4.0.0)

Odina un array in ordine decrescente

```
void rsort (array array [, int sort_flags])
```

Questa funzione ordina un array in ordine decrescente.

Esempio 1. esempio di rsort()

```
$fruitti = array ("limone", "arancia", "banana", "mela");
rsort ($frutti);
reset ($frutti);
while (list ($chiave, $valore) = each ($frutti)) {
    echo "$chiave = $valore\n";
}
```

Questo esempio mostrerà:

```
0 = mela
1 = limone
2 = banana
3 = arancia
```

I frutti sono stati ordinati in ordine alfabetico decrescente.

Si può modificare il comportamento dell'ordinamento usando il parametro opzionale *sort_flags*, per maggiori dettagli vedere sort().

Vedere anche: arsort(), asort(), ksort(), sort(), e usort().

shuffle (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Mescola un array

```
void shuffle (array array)
```

Questa funzione mescola un array (rende casuale l'ordine degli elementi). Si deve usare srand() per inizializzare il generatore di numeri casuali.

Esempio 1. esempio di shuffle()

```
$numeri = range (1,20);
srand ((float)microtime()*1000000);
shuffle ($numeri);
while (list (, $numeero) = each ($numeri)) {
    echo "$numero ";
}
```

Vedere anche arsort(), asort(), ksort(), rsort(), sort() e usort().

sizeof (PHP 3, PHP 4 >= 4.0.0)

Conta gli elementi in una variabile

```
int sizeof (mixed var)
```

La funzione **sizeof()** è un alias di count().

Vedere anche count().

sort (PHP 3, PHP 4 >= 4.0.0)

Ordina un array

```
void sort (array array [, int sort_flags])
```

Questa funzione ordina un array. Gli elementi vengono disposti dal più piccolo al più grande.

Esempio 1. esempio di sort()

```
<?php
```

```
$frutti = array ("limon", "arancia", "banana", "mela");
sort ($frutti);
reset ($frutti);
while (list ($chiave, $valore) = each ($frutti)) {
    echo "frutti[".$chiave."] = ".$valore;
}

?>
```

Questo esempio mostrerà:

```
frutti[0] = arancia
frutti[1] = banana
frutti[2] = limone
frutti[3] = mela
```

I frutti sono stati ordinati in ordine alfabetico.

Il secondo parametro opzionale *`sort_flags`* può essere usato per modificare il comportamento dell'ordinamento, usando i seguenti valori:

flag d'ordinamento:

- SORT_REGULAR - compara gli elementi in modo normale
- SORT_NUMERIC - compara gli elementi numericamente
- SORT_STRING - compara gli elementi convertiti in stringa

Vedere anche: arsort(), asort(), ksort(), natsort(), natcasesort(), rsort(), usort(), array_multisort(), e uksort().

> **Nota:** Il secondo parametro è stato aggiunto nel PHP 4.

uasort (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Ordina un array mediante una funzione definita dall'utente e mantiene le associazioni

```
void uasort (array array, function cmp_function)
```

Questa funzione ordina un array in modo tale che le chiavi mantengano la loro correlazione con gli elementi dell'array a cui sono associate. Questo è utile quando si ordinano array associativi in cui l'ordine degli elementi è importante. La funzione di comparazione deve essere fornita dall'utente.

> **Nota:** Vedere usort() e uksort() per esempio di funzioni di comparazione.

Vedere anche: usort(), uksort(), sort(), asort(), arsort(), ksort() e rsort().

uksort (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Ordina rispetto alle chiavi di un array mediante una funzione definita dall'utente

```
void uksort (array array, function cmp_function)
```

Ordina rispetto alle chiavi di un array mediante una funzione di comparazione definita dall'utente. Se si vuole ordinare un array con dei criteri non usuali, si deve usare questa funzione.

Esempio 1. esempio di uksort()

```
function cmp ($a, $b) {
    if ($a == $b) return 0;
    return ($a > $b) ? -1 : 1;
}

$a = array (4 => "quattro", 3 => "tre", 20 => "venti", 10 => "dieci");

uksort ($a, "cmp");

while (list ($chiave, $valore) = each ($a)) {
    echo "$chiave: $valore\n";
}
```

Questo esempio mostrerà:

```
20: venti
10: dieci
4: quattro
3: tre
```

Vedere anche: usort(), uasort(), sort(), asort(), arsort(), ksort(), natsort() e rsort().

usort (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Ordina un array mediante una funzione definita dall'utente

```
void usort (array array, string cmp_function)
```

Ordina i valori di un array mediante una funzione di comparazione definita dall'utente. Se si vuole ordinare un array con dei criteri non usuali, si deve usare questa funzione.

La funzione di comparazione deve restituire un intero minore, uguale o maggiore di zero a seconda che il primo argomento sia considerato rispettivamente minore, uguale o maggiore del secondo. Se due elementi risultano uguali, il loro ordine dnell'array ordinato è indefinito.

Esempio 1. esempio di usort()

```
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 ($chiave, $valore) = each ($a)) {
    echo "$chiave: $valore\n";
}
```

Questo esempio mostrerà:

```
0: 6
1: 5
2: 3
3: 2
4: 1
```

> **Nota:** Ovviamente, in questo caso banale di ordinamento decrescente la funzione rsort() sarebbe stata più appropriata.

Esempio 2. esempio di usort() con un array multidimensionale

```
function cmp ($a, $b) {
    return strcmp($a["frutto"],$b["frutto"]);
}

$frutti[0]["frutto"] = "limoni";
$frutti[1]["frutto"] = "arance";
$frutti[2]["frutto"] = "uva";

usort($frutti, "cmp");

while (list ($chiave, $valore) = each ($frutti)) {
    echo "\$frutti[$chiave]: " . $valore["frutto"] . "\n";
}
```

Quando si ordina un array multidimensionale, $a e $b contengono rierimenti al primo indice dell'array.

Questo esempio mostrerà:

```
$fruits[0]: arance
$fruits[1]: limoni
$fruits[2]: uva
```

Attenzione

La sottostante funzione di quicksort pu‘ ausare, inalcune librerie C (per esempio, sui sistemi Solaris) un grash del PHP se la funzione di comparazione non restituisce valori coerenti.

Vedere anche: uasort(), uksort(), sort(), asort(), arsort(),ksort(), natsort(), e rsort().

III. Funzioni Aspell [deprecated]

Le funzioni **aspell()** permettono di controllare la correttezza di una parola e di offrire suggerimenti.

> **Nota:** aspell funziona solo con versioni molto vecchie (più o meno fino alla .27.*) della libreria aspell. Né il presente modulo, né quelle versioni della libreria sono più supportate. Se si vuole usare le funzionalità di controllo grammaticale in php, usare piuttosto pspell. Usa la libreria pspell e funziona con le nuove versioni di aspell.

È necessaria la libreria aspell, disponibile su: http://aspell.sourceforge.net/.

aspell_new (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Carca un nuovo dizionario [deprecated]

```
int aspell_new (string master [, string personal])
```

aspell_new() apre un nuovo dizionario e restituisce un puntatore (link) identificatore del dizionario, da utilizzare in altre funzioni aspell. Restituisce FALSE in caso di errore.

Esempio 1. aspell_new()

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

aspell_check (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Controlla una parola [deprecated]

```
bool aspell_check (int link_dizionario, string parola)
```

aspell_check() controlla la compitazione di una parola e restituisce TRUE se è corretta, FALSE altrimenti.

Esempio 1. aspell_check()

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

if (aspell_check($aspell_link, "provva")) {
    echo "La parola è corretta";
} else {
    echo "Spiacente, parola non corretta";
}
```

aspell_check_raw (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Controlla una parola senza togliere le maiuscole o cercare di eliminare gli spazi inutili [deprecated]

```
bool aspell_check_raw (int link_dizionario, string parola)
```

aspell_check_raw() controlla la correttezza di una parola, senza modificare le maiuscole/minusciole o cercare di eliminare gli spazi inutili e restituisce TRUE se è corretta, FALSE altrimenti.

Esempio 1. aspell_check_raw()

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

if (aspell_check_raw($aspell_link, "prova")) {
    echo "La parola ‘ corretta";
} else {
    echo "Spiacente, parola non corretta";
}
```

aspell_suggest (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Suggerisce correzioni di una parola [deprecated]

```
array aspell_suggest (int link_dizionario, string parola)
```

aspell_suggest() restituisce un array di possibili correzioni per la parola data.

Esempio 1. aspell_suggest()

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

if (!aspell_check($aspell_link, "prova")) {
    $suggerimenti = aspell_suggest($aspell_link, "prova");

    foreach ($suggerimenti as $suggerimento) {
        echo "Possibile parola corretta: $suggerimento<br>\n";
    }
}
```

IV. BCMath Arbitrary Precision Mathematics Functions

In PHP 4, these functions are only available if PHP was configured with `--enable-bcmath`. In PHP 3, these functions are only available if PHP was not configured with `--disable-bcmath`.

> **Nota:** Due to changes in the licensing, the BCMATH library is distributed separate from the standard PHP source distribution. You can download the tar-gzipped archive at the url: http://www.php.net/extra/number4.tar.gz. Read the file `README.BCMATH` in the PHP distribution for more information.

bcadd (PHP 3, PHP 4 >= 4.0.0)

Add two arbitrary precision numbers

```
string bcadd (string left operand, string right operand [, int scale])
```

Adds the *left operand* to the *right operand* and returns the sum in a string. The optional *scale* parameter is used to set the number of digits after the decimal place in the result.

See also bcsub().

bccomp (PHP 3, PHP 4 >= 4.0.0)

Compare two arbitrary precision numbers

```
int bccomp (string left operand, string right operand [, int scale])
```

Compares the *left operand* to the *right operand* and returns the result as an integer. The optional *scale* parameter is used to set the number of digits after the decimal place which will be used in the comparion. The return value is 0 if the two operands are equal. If the *left operand* is larger than the *right operand* the return value is +1 and if the *left operand* is less than the *right operand* the return value is -1.

bcdiv (PHP 3, PHP 4 >= 4.0.0)

Divide two arbitrary precision numbers

```
string bcdiv (string left operand, string right operand [, int scale])
```

Divides the *left operand* by the *right operand* and returns the result. The optional *scale* sets the number of digits after the decimal place in the result.

See also bcmul().

bcmod (PHP 3, PHP 4 >= 4.0.0)

Get modulus of an arbitrary precision number

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

Get the modulus of the *left operand* using *modulus*.

See also bcdiv().

bcmul (PHP 3, PHP 4 >= 4.0.0)

Multiply two arbitrary precision number

```
string bcmul (string left operand, string right operand [, int scale])
```

Multiply the *left operand* by the *right operand* and returns the result. The optional *scale* sets the number of digits after the decimal place in the result.

See also bcdiv().

bcpow (PHP 3, PHP 4 >= 4.0.0)

Raise an arbitrary precision number to another

```
string bcpow (string x, string y [, int scale])
```

Raise *x* to the power *y*. The optional *scale* can be used to set the number of digits after the decimal place in the result.

See also bcsqrt().

bcscale (PHP 3, PHP 4 >= 4.0.0)

Set default scale parameter for all bc math functions

```
string bcscale (int scale)
```

This function sets the default scale parameter for all subsequent bc math functions that do not explicitly specify a scale parameter.

bcsqrt (PHP 3, PHP 4 >= 4.0.0)

Get the square root of an arbitray precision number

```
string bcsqrt (string operand [, int scale])
```

Return the square root of the *operand*. The optional *scale* parameter sets the number of digits after the decimal place in the result.

See also bcpow().

bcsub (PHP 3, PHP 4 >= 4.0.0)

Subtract one arbitrary precision number from another

```
string bcsub (string left operand, string right operand [, int scale])
```

Subtracts the *right operand* from the *left operand* and returns the result in a string. The optional *scale* parameter is used to set the number of digits after the decimal place in the result.

See also bcadd().

V. Bzip2 Compression Functions

This module uses the functions of the bzip2 (http://sources.redhat.com/bzip2/) library by Julian Seward to transparently read and write bzip2 (.bz2) compressed files.

Bzip2 support in PHP is not enabled by default. You will need to use the --with-bz2 configuration option when compiling PHP to enable bzip2 support. This module requires bzip2/libbzip2 version >= 1.0.x.

Small code example

This example opens a temporary file and writes a test string to it, then prints out the contents of the file.

Esempio 1. Small bzip2 Example

```
<?php

$filename = "/tmp/testfile.bz2";
$str = "This is a test string.\n";

// open file for writing
$bz = bzopen($filename, "w");

// write string to file
bzwrite($bz, $str);

// close file
bzclose($bz);

// open file for reading
$bz = bzopen($filename, "r");

// read 10 characters
print bzread($bz, 10);

// output until end of the file (or the next 1024 char) and close it.
print bzread($bz);

bzclose($bz);

?>
```

bzclose (PHP 4 >= 4.0.4)

Close a bzip2 file pointer

```
int bzclose (int bz)
```

Closes the bzip2 file referenced by the pointer *bz*.

Returns TRUE on success and FALSE on failure.

The file pointer must be valid, and must point to a file successfully opened by bzopen().

See also bzopen().

bzcompress (PHP 4 >= 4.0.4)

Compress a string into bzip2 encoded data

```
string bzcompress (string source [, int blocksize [, int workfactor]])
```

bzcompress() compresses the *source* string and returns it as bzip2 encoded data.

The optional parameter *blocksize* specifies the blocksize used during compression and should be a number from 1 to 9 with 9 giving the best compression, but using more resources to do so. *blocksize* defaults to 4.

The optional parameter *workfactor* controls how the compression phase behaves when presented with worst case, highly repetitive, input data. The value can be between 0 and 250 with 0 being a special case and 30 being the default value. Regardless of the *workfactor*, the generated output is the same.

Esempio 1. bzcompress() Example

```
<?php
$str = "sample data";
$bzstr = bzcompress($str, 9);
print( $bzstr );
?>
```

See also bzdecompress().

bzdecompress (PHP 4 >= 4.0.4)

Decompresses bzip2 encoded data

```
string bzdecompress (string source [, int small])
```

bzdecompress() decompresses the *source* string containing bzip2 encoded data and returns it. If the optional parameter *small* is TRUE, an alternative decompression algorithm will be used which uses less memory (the maximum memory requirement drops to around 2300K) but works at roughly half the speed. See the bzip2 documentation (http://sources.redhat.com/bzip2/) for more information about this feature.

Esempio 1. bzdecompress()

```
<?php
$start_str = "This is not an honest face?";
$bzstr = bzcompress($start_str);

print( "Compressed String: " );
print( $bzstr );
print( "\n<br>n" );

$str = bzdecompress($bzstr);
print( "Decompressed String: " );
print( $str );
print( "\n<br>n" );
?>
```

See also bzcompress().

bzerrno (PHP 4 >= 4.0.4)

Returns a bzip2 error number

```
int bzerrno (int bz)
```

Returns the error number of any bzip2 error returned by the file pointer *bz*.

See also bzerror() and bzerrstr().

bzerror (PHP 4 >= 4.0.4)

Returns the bzip2 error number and error string in an array

```
array bzerror (int bz)
```

Returns the error number and error string, in an associative array, of any bzip2 error returned by the file pointer *bz*.

Esempio 1. bzerror() Example

```
<?php
$error = bzerror($bz);

echo $error["errno"];
echo $error["errstr"];
?>
```

See also bzerrno() and bzerrstr().

bzerrstr (PHP 4 >= 4.0.4)

Returns a bzip2 error string

```
string bzerrstr (int bz)
```

Returns the error string of any bzip2 error returned by the file pointer *bz*.

See also bzerrno() and bzerror().

bzflush (PHP 4 >= 4.0.4)

Force a write of all buffered data

```
int bzflush (int bz)
```

Forces a write of all buffered bzip2 data for the file pointer *bz*.

Returns TRUE on success, FALSE on failure.

See also bzread() and bzwrite().

bzopen (PHP 4 >= 4.0.4)

Open a bzip2 compressed file

```
int bzopen (string filename, string mode)
```

Opens a bzip2 (.bz2) file for reading or writing. *filename* is the name of the file to open. *mode* is similar to the fopen() function ('r' for read, 'w' for write, etc.).

If the open fails, the function returns FALSE, otherwise it returns a pointer to the newly opened file.

Esempio 1. bzopen() Example

```
<?php
$bz = bzopen("/tmp/foo.bz2", "r");
$decompressed_file = bzread($bz, filesize("/tmp/foo.bz2"));
bzclose($bz);

print( "The contents of /tmp/foo.bz2 are: " );
print( "\n<br>n" );
print( $decompressed_file );
?>
```

See also bzclose().

bzread (PHP 4 >= 4.0.4)

Binary safe bzip2 file read

```
string bzread (int bz [, int length])
```

bzread() reads up to *length* bytes from the bzip2 file pointer referenced by *bz*. Reading stops when *length* (uncompressed) bytes have been read or EOF is reached, whichever comes first. If the optional parameter *length* is not specified, **bzread()** will read 1024 (uncompressed) bytes at a time.

Esempio 1. bzread() Example

```
<?php
$bz = bzopen("/tmp/foo.bz2", "r");
$str = bzread($bz, 2048);
print( $str );
?>
```

See also bzwrite() and bzopen().

bzwrite (PHP 4 >= 4.0.4)

Binary safe bzip2 file write

```
int bzwrite (int bz, string data [, int length])
```

bzwrite() writes the contents of the string *data* to the bzip2 file stream pointed to by *bz*. If the optional *length* argument is given, writing will stop after length (uncompressed) bytes have been written or the end of string is reached, whichever comes first.

Esempio 1. bzwrite() Example

```
<?php
$str = "uncompressed data";
$bz = bzopen("/tmp/foo.bz2", "w");
bzwrite($bz, $str, strlen($str));
bzclose($bz);
?>
```

See also bzread() and bzopen().

VI. Calendar functions

The calendar functions are only available if you have compiled the calendar extension, found in either the "dl" or "ext" subdirectories of your PHP source code. Please see the README file before using it.

The calendar extension presents a series of functions to simplify converting between different calendar formats. The intermediary or standard it is based on is the Julian Day Count. The Julian Day Count is a count of days starting way earlier than any date most people would need to track (somewhere around 4000bc). To convert between calendar systems, you must first convert to Julian Day Count, then to the calendar system of your choice. Julian Day Count is very different from the Julian Calendar! For more information on calendar systems visit http://genealogy.org/~scottlee/cal-overview.html. Excerpts from this page are included in these instructions, and are in quotes.

JDToGregorian (PHP 3, PHP 4 >= 4.0.0)

Converts Julian Day Count to Gregorian date

```
string jdtogregorian (int julianday)
```

Converts Julian Day Count to a string containing the Gregorian date in the format of "month/day/year".

GregorianToJD (PHP 3, PHP 4 >= 4.0.0)

Converts a Gregorian date to Julian Day Count

```
int gregoriantojd (int month, int day, int year)
```

Valid Range for Gregorian Calendar 4714 B.C. to 9999 A.D.

Although this software can handle dates all the way back to 4714 B.C., such use may not be meaningful. The Gregorian calendar was not instituted until October 15, 1582 (or October 5, 1582 in the Julian calendar). Some countries did not accept it until much later. For example, Britain converted in 1752, The USSR in 1918 and Greece in 1923. Most European countries used the Julian calendar prior to the Gregorian.

Esempio 1. Calendar functions

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

JDToJulian (PHP 3, PHP 4 >= 4.0.0)

Converts a Julian Day Count to a Julian Calendar Date

```
string jdtojulian (int julianday)
```

Converts Julian Day Count to a string containing the Julian Calendar Date in the format of "month/day/year".

JulianToJD (PHP 3, PHP 4 >= 4.0.0)

Converts a Julian Calendar date to Julian Day Count

```
int juliantojd (int month, int day, int year)
```

Valid Range for Julian Calendar 4713 B.C. to 9999 A.D.

Although this software can handle dates all the way back to 4713 B.C., such use may not be meaningful. The calendar was created in 46 B.C., but the details did not stabilize until at least 8 A.D., and perhaps as late at the 4th century. Also, the beginning of a year varied from one culture to another - not all accepted January as the first month.

JDToJewish (PHP 3, PHP 4 >= 4.0.0)

Converts a Julian Day Count to the Jewish Calendar

```
string jdtojewish (int julianday)
```

Converts a Julian Day Count the the Jewish Calendar.

JewishToJD (PHP 3, PHP 4 >= 4.0.0)

Converts a date in the Jewish Calendar to Julian Day Count

```
int jewishtojd (int month, int day, int year)
```

Valid Range Although this software can handle dates all the way back to the year 1 (3761 B.C.), such use may not be meaningful.

The Jewish calendar has been in use for several thousand years, but in the early days there was no formula to determine the start of a month. A new month was started when the new moon was first observed.

JDToFrench (PHP 3, PHP 4 >= 4.0.0)

Converts a Julian Day Count to the French Republican Calendar

```
string jdtofrench (int juliandaycount)
```

Converts a Julian Day Count to the French Republican Calendar.

FrenchToJD (PHP 3, PHP 4 >= 4.0.0)

Converts a date from the French Republican Calendar to a Julian Day Count

```
int frenchtojd (int month, int day, int year)
```

Converts a date from the French Republican Calendar to a Julian Day Count.

These routines only convert dates in years 1 through 14 (Gregorian dates 22 September 1792 through 22 September 1806). This more than covers the period when the calendar was in use.

JDMonthName (PHP 3, PHP 4 >= 4.0.0)

Returns a month name

```
string jdmonthname (int julianday, int mode)
```

Returns a string containing a month name. *mode* tells this function which calendar to convert the Julian Day Count to, and what type of month names are to be returned.

Tabella 1. Calendar modes

Mode	Meaning

JDDayOfWeek (PHP 3, PHP 4 >= 4.0.0)

Returns the day of the week

```
mixed jddayofweek (int julianday, int mode)
```

Returns the day of the week. Can return a string or an int depending on the mode.

Tabella 1. Calendar week modes

Mode	Meaning

easter_date (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Get UNIX timestamp for midnight on Easter of a given year

```
int easter_date (int year)
```

Returns the UNIX timestamp corresponding to midnight on Easter of the given year. If no year is specified, the current year is assumed.

Warning: This function will generate a warning if the year is outside of the range for UNIX timestamps (i.e. before 1970 or after 2037).

Esempio 1. easter_date() example

```
echo date ("M-d-Y", easter_date(1999));        /* "Apr-04-1999" */
echo date ("M-d-Y", easter_date(2000));        /* "Apr-23-2000" */
echo date ("M-d-Y", easter_date(2001));        /* "Apr-15-2001" */
```

The date of Easter Day was defined by the Council of Nicaea in AD325 as the Sunday after the first full moon which falls on or after the Spring Equinox. The Equinox is assumed to always fall on 21st March, so the calculation reduces to determining the date of the full moon and the date of the following Sunday. The algorithm used here was introduced around the year 532 by Dionysius Exiguus. Under the Julian Calendar (for years before 1753) a simple 19-year cycle is used to track the phases of the Moon. Under the Gregorian Calendar (for years after 1753 - devised by Clavius and Lilius, and introduced by Pope Gregory XIII in October 1582, and into Britain and its then colonies in September 1752) two correction factors are added to make the cycle more accurate.

(The code is based on a C program by Simon Kershaw, <webmaster@ely.anglican.org>)

See easter_days() for calculating Easter before 1970 or after 2037.

easter_days (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Get number of days after March 21 on which Easter falls for a given year

```
int easter_days (int year)
```

Returns the number of days after March 21 on which Easter falls for a given year. If no year is specified, the current year is assumed.

This function can be used instead of easter_date() to calculate Easter for years which fall outside the range of UNIX timestamps (i.e. before 1970 or after 2037).

Esempio 1. easter_days() example

```
echo easter_days (1999);        /* 14, i.e. April 4   */
echo easter_days (1492);        /* 32, i.e. April 22  */
echo easter_days (1913);        /*  2, i.e. March 23  */
```

The date of Easter Day was defined by the Council of Nicaea in AD325 as the Sunday after the first full moon which falls on or after the Spring Equinox. The Equinox is assumed to always fall on 21st March, so the calculation reduces to determining the date of the full moon and the date of the following Sunday. The algorithm used here was introduced around the year 532 by Dionysius Exiguus. Under the Julian Calendar (for years before 1753) a simple 19-year cycle is used to track the phases of the Moon. Under the Gregorian Calendar (for years after 1753 - devised by Clavius and Lilius, and introduced by Pope Gregory XIII in October 1582, and into Britain and its then colonies in September 1752) two correction factors are added to make the cycle more accurate.

(The code is based on a C program by Simon Kershaw, <webmaster@ely.anglican.org>)

See also easter_date().

unixtojd (PHP 4 >= 4.0.0)

Convert UNIX timestamp to Julian Day

```
int unixtojd ([int timestamp])
```

Return the Julian Day for a UNIX *`timestamp`* (seconds since 1.1.1970), or for the current day if no *`timestamp`* is given.

See also jdtounix().

> **Nota:** This function is only available in PHP versions after PHP 4RC1.

jdtounix (PHP 4 >= 4.0.0)

Convert Julian Day to UNIX timestamp

```
int jdtounix (int jday)
```

This function will return a UNIX timestamp corresponding to the Julian Day given in *jday* or FALSE if *jday* is not inside the UNIX epoch (Gregorian years between 1970 and 2037 or 2440588 <= *jday* <= 2465342)

See also **jdtounix()**.

Nota: This function is only available in PHP versions after PHP 4RC1.

VII. CCVS API Functions

These functions interface the CCVS API, allowing you to directly work with CCVS from your PHP scripts. CCVS is RedHat's (http://www.redhat.com/) solution to the "middle-man" in credit card processing. It lets you directly address the credit card clearing houses via your *nix box and a modem. Using the CCVS module for PHP, you can process credit cards directly through CCVS via your PHP Scripts. The following references will outline the process.

To enable CCVS Support in PHP, first verify your CCVS installation directory. You will then need to configure PHP with the `--with-ccvs` option. If you use this option without specifying the path to your CCVS installation, PHP Will attempt to look in the default CCVS Install location (/usr/local/ccvs). If CCVS is in a non-standard location, run configure with: `--with-ccvs=$ccvs_path`, where $ccvs_path is the path to your CCVS installation. Please note that CCVS support requires that $ccvs_path/lib and $ccvs_path/include exist, and include cv_api.h under the include directory and libccvs.a under the lib directory.

Additionally, a ccvsd process will need to be running for the configurations you intend to use in your PHP scripts. You will also need to make sure the PHP Processes are running under the same user as your CCVS was installed as (e.g. if you installed CCVS as user 'ccvs', your PHP processes must run as 'ccvs' as well.)

Additional information about CCVS can be found at http://www.redhat.com/products/ccvs.

This documentation section is being worked on. Until then, RedHat maintains slightly outdated but still useful documentation at http://www.redhat.com/products/ccvs/support/CCVS3.3docs/ProgPHP.html.

(unknown)

( )

VIII. COM support functions for Windows

COM functions are only available on the Windows version of PHP. These functions have been added in PHP 4.

COM (unknown)

COM class

```
$obj = new COM("server.object")
```

The COM class provides a framework to integrate (D)COM components into your php scripts.

```
string COM::COM (string module_name [, string server_name [, int
codepage]])
```

COM class constructor. Parameters:

module_name

name or class-id of the requested component.

server_name

name of the DCOM server from which the component should be fetched. If `NULL`, `localhost` is assumed. To allow DCOM `com.allow_dcom` has to be set to `TRUE` in `php.ini`.

codepage

specifies the codepage that is used to convert php-strings to unicode-strings and vice versa. Possible values are `CP_ACP`, `CP_MACCP`, `CP_OEMCP`, `CP_SYMBOL`, `CP_THREAD_ACP`, `CP_UTF7` and `CP_UTF8`.

Esempio 1. COM example (1)

```
// starting word
$word = new COM("word.application") or die("Unable to instanciate Word");
print "Loaded Word, version {$word->Version}\n";

//bring it to front
$word->Visible = 1;

//open an empty document
$word->Documents->Add();

//do some weird stuff
$word->Selection->TypeText("This is a test...");
$word->Documents[1]->SaveAs("Useless test.doc");

//closing word
$word->Quit();
```

```
//free the object
$word->Release();
$word = null;
```

Esempio 2. COM example (2)

```
$conn = new COM("ADODB.Connection") or die("Cannot start ADO");
$conn->Open("Provider=SQLOLEDB; Data Source=localhost;
Initial Catalog=database; User ID=user; Password=password");

$rs = $conn->Execute("SELECT * FROM sometable");    // Recordset

$num_columns = $rs->Fields->Count();
echo $num_columns . "\n";

for ($i=0; $i < $num_columns; $i++)
{
    $fld[$i] = $rs->Fields($i);
}

$rowcount = 0;
while (!$rs->EOF)
{
    for ($i=0; $i < $num_columns; $i++)
    {
        echo $fld[$i]->value . "\t";
    }
    echo "\n";
    $rowcount++;            // increments rowcount
    $rs->MoveNext();
}

$rs->Close();
$conn->Close();

$rs->Release();
$conn->Release();

$rs = null;
$conn = null;
```

VARIANT (unknown)

VARIANT class

```
$vVar = new VARIANT($var)
```

A simple container to wrap variables into VARIANT structures.

```
string VARIANT::VARIANT ([mixed value [, int type [, int codepage]]])
```

VARIANT class constructor. Parameters:

value

initial value. if omitted an VT_EMPTY object is created.

type

specifies the content type of the VARIANT object. Possible values are VT_UI1, VT_UI2, VT_UI4, VT_I1, VT_I2, VT_I4, VT_R4, VT_R8, VT_INT, VT_UINT, VT_BOOL, VT_ERROR, VT_CY, VT_DATE, VT_BSTR, VT_DECIMAL, VT_UNKNOWN, VT_DISPATCH and VT_VARIANT. These values are mutual exclusive, but they can be combined with VT_BYREF to specify beeing a value. If omitted, the type of *value* is used. Consult the msdn library for additional information.

codepage

specifies the codepage that is used to convert php-strings to unicode-strings and vice versa. Possible values are CP_ACP, CP_MACCP, CP_OEMCP, CP_SYMBOL, CP_THREAD_ACP, CP_UTF7 and CP_UTF8.

com_load (PHP 3>= 3.0.3)

Creates a new reference to a COM component

```
string com_load (string module name [, string server name [, int
codepage]])
```

com_load() creates a new COM component and returns a reference to it. Returns FALSE on failiure.Possible values for *codepage* are CP_ACP, CP_MACCP, CP_OEMCP, CP_SYMBOL, CP_THREAD_ACP, CP_UTF7 and CP_UTF8.

com_invoke (PHP 3>= 3.0.3)

Calls a COM component's method.

```
mixed com_invoke (resource com_object, string function_name [, mixed
function parameters, ...])
```

com_invoke() invokes a method of the COM component referenced by *com_object*. Returns FALSE on error, returns the *function_name*'s return value on success.

com_propget (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Gets the value of a COM Component's property

```
mixed com_propget (resource com_object, string property)
```

This function is an alias for com_get().

com_get (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Gets the value of a COM Component's property

```
mixed com_get (resource com_object, string property)
```

Returns the value of the *property* of the COM component referenced by *com_object*. Returns FALSE on error.

com_propput (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Assigns a value to a COM component's property

```
void com_propput (resource com_object, string property, mixed value)
```

This function is an alias for com_set().

com_propset (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Assigns a value to a COM component's property

```
void com_propset (resource com_object, string property, mixed value)
```

This function is an alias for com_set().

com_set (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Assigns a value to a COM component's property

```
void com_set (resource com_object, string property, mixed value)
```

Sets the value of the *property* of the COM component referenced by *com_object*. Returns the newly set value if succeeded, FALSE on error.

com_addref (PHP 4 >= 4.0.7RC1)

Increases the components reference counter.

```
void com_addref ()
```

Increases the components reference counter.

com_release (PHP 4 >= 4.0.7RC1)

Decreases the components reference counter.

```
void com_release ()
```

Decreases the components reference counter.

IX. Class/Object Functions

Introduction

About

These functions allow you to obtain information about classes and instance objects. You can obtain the name of the class to which a object belongs, as well as its member properties and methods. Using these functions, you can find out not only the class membership of an object, but also its parentage (i.e. what class is the object class extending).

An example of use

In this example, we first define a base class and an extension of the class. The base class describes a general vegetable, whether it is edible or not and what is its color. The subclass `Spinach` adds a method to cook it and another to find out if it is cooked.

Esempio 1. classes.inc

```
<?php

// base class with member properties and methods
class Vegetable {

    var $edible;
    var $color;

    function Vegetable( $edible, $color="green" ) {
        $this->edible = $edible;
        $this->color = $color;
    }

    function is_edible() {
        return $this->edible;
    }

    function what_color() {
        return $this->color;
    }

} // end of class Vegetable


// extends the base class
```

```
class Spinach extends Vegetable {

    var $cooked = false;

    function Spinach() {
        $this->Vegetable( true, "green" );
    }

    function cook_it() {
        $this->cooked = true;
    }

    function is_cooked() {
        return $this->cooked;
    }

} // end of class Spinach

?>
```

We then instantiate 2 objects from these classes and print out information about them, including their class parentage. We also define some utility functions, mainly to have a nice printout of the variables.

Esempio 2. test_script.php

```
<pre>
<?php

include "classes.inc";

// utility functions

function print_vars($obj) {
    $arr = get_object_vars($obj);
    while (list($prop, $val) = each($arr))
        echo "\t$prop = $val\n";
}

function print_methods($obj) {
    $arr = get_class_methods(get_class($obj));
    foreach ($arr as $method)
        echo "\tfunction $method()\n";
}

function class_parentage($obj, $class) {
    global $$obj;
    if (is_subclass_of($$obj, $class)) {
```

```
        echo "Object $obj belongs to class ".get_class($$obj);
        echo " a subclass of $class\n";
    } else {
        echo "Object $obj does not belong to a subclass of $class\n";
    }
}

// instantiate 2 objects

$veggie = new Vegetable(true,"blue");
$leafy = new Spinach();

// print out information about objects
echo "veggie: CLASS ".get_class($veggie)."\n";
echo "leafy: CLASS ".get_class($leafy);
echo ", PARENT ".get_parent_class($leafy)."\n";

// show veggie properties
echo "\nveggie: Properties\n";
print_vars($veggie);

// and leafy methods
echo "\nleafy: Methods\n";
print_methods($leafy);

echo "\nParentage:\n";
class_parentage("leafy", "Spinach");
class_parentage("leafy", "Vegetable");
?>
</pre>
```

One important thing to note in the example above is that the object `$leafy` is an instance of the class `Spinach` which is a subclass of `Vegetable`, therefore the last part of the script above will output:

```
       [...]
Parentage:
Object leafy does not belong to a subclass of Spinach
Object leafy belongs to class spinach a subclass of Vegetable
```

call_user_method_array (PHP 4 >= 4.0.5)

Call a user method given with an array of parameters

```
mixed call_user_method_array  (string method_name, object obj [, array
paramarr])
```

Calls a the method referred by *method_name* from the user defined *obj* object, using the paramaters in *paramarr*.

See also: **call_user_func_array()**, call_user_func(), call_user_method().

> **Nota:** This function was added to the CVS code after release of PHP 4.0.4pl1

call_user_method (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Call a user method on an specific object

```
mixed call_user_method  (string method_name, object obj [, mixed parameter
[, mixed ...]])
```

Calls a the method referred by *method_name* from the user defined *obj* object. An example of usage is below, where we define a class, instantiate an object and use **call_user_method()** to call indirectly its `print_info` method.

```
<?php
class Country {
    var $NAME;
    var $TLD;

    function Country($name, $tld) {
        $this->NAME = $name;
        $this->TLD = $tld;
    }

    function print_info($prestr="") {
        echo $prestr."Country: ".$this->NAME."\n";
        echo $prestr."Top Level Domain: ".$this->TLD."\n";
    }
}

$cntry = new Country("Peru","pe");
```

```
echo "* Calling the object method directly\n";
$cntry->print_info();

echo "\n* Calling the same method indirectly\n";
call_user_method ("print_info", $cntry, "\t");
?>
```

See also **call_user_func_array()**, call_user_func(), call_user_method_array().

class_exists (PHP 4 >= 4.0.0)

Checks if the class has been defined

```
bool class_exists (string class_name)
```

This function returns TRUE if the class given by *class_name* has been defined, FALSE otherwise.

get_class (PHP 4 >= 4.0.0)

Returns the name of the class of an object

```
string get_class (object obj)
```

This function returns the name of the class of which the object *obj* is an instance.

> **Nota: get_class()** returns the class name in lowercase form.

See also get_parent_class(), is_subclass_of()

get_class_methods (PHP 4 >= 4.0.0)

Returns an array of class methods' names

```
array get_class_methods (string class_name)
```

This function returns an array of method names defined for the class specified by *class_name*.

Nota: As of PHP 4.0.6, you can specify the object itself instead of *`class_name`*. For example:

```
$class_methods = get_class_methods($my_class);
```

Esempio 1. get_class_methods() example

```
<?php

class myclass {
    // constructor
    function myclass() {
        return(true);
    }

    // method 1
    function myfunc1() {
        return(true);
    }

    // method 2
    function myfunc2() {
        return(true);
    }
}

$my_class = new myclass();

$class_methods = get_class_methods(get_class($my_class));

foreach ($class_methods as $method_name) {
    echo "$method_name\n";
}

?>
```

Will produce:

```
myclass
myfunc1
myfunc2
```

See also get_class_vars(), get_object_vars()

get_class_vars (PHP 4 >= 4.0.0)

Returns an array of default properties of the class

```
array get_class_vars (string class_name)
```

This function will return an associative array of default properties of the class. The resulting array elements are in the form of *varname => value*.

Nota: Uninitialized class variables will not be reported by **get_class_vars()**.

Esempio 1. get_class_vars() example

```
<?php

class myclass {

    var $var1; // this has no default value...
    var $var2 = "xyz";
    var $var3 = 100;

    // constructor
    function myclass() {
        return(true);
    }

}

$my_class = new myclass();

$class_vars = get_class_vars(get_class($my_class));

foreach ($class_vars as $name => $value) {
    echo "$name : $value\n";
}

?>
```

Will produce:

```
var2 : xyz
var3 : 100
```

See also get_class_methods(), get_object_vars()

get_declared_classes (PHP 4 >= 4.0.0)

Returns an array with the name of the defined classes

```
array get_declared_classes ()
```

This function returns an array of the names of the declared classes in the current script.

> **Nota:** In PHP 4.0.1pl2, three extra classes are returned at the beginning of the array: `stdClass` (defined in `Zend/zend.c`), `OverloadedTestClass` (defined in `ext/standard/basic_functions.c`) and `Directory` (defined in `ext/standard/dir.c`).
>
> Also note that depending on what libraries you have compiled into PHP, additional classes could be present.

get_object_vars (PHP 4 >= 4.0.0)

Returns an associative array of object properties

```
array get_object_vars (object obj)
```

This function returns an associative array of defined object properties for the specified object *`obj`*. If variables declared in the class of which the *`obj`* is an instance, have not been assigned a value, those will not be returned in the array.

Esempio 1. Use of get_object_vars()

```
<?php
class Point2D {
    var $x, $y;
    var $label;

    function Point2D($x, $y) {
        $this->x = $x;
        $this->y = $y;
    }

    function setLabel($label) {
        $this->label = $label;
    }
```

```
    function getPoint() {
        return array("x" => $this->x,
                     "y" => $this->y,
                     "label" => $this->label);
    }
}

$p1 = new Point2D(1.233, 3.445);
print_r(get_object_vars($p1));
// "$label" is declared but not defined
// Array
// (
//     [x] => 1.233
//     [y] => 3.445
// )

$p1->setLabel("point #1");
print_r(get_object_vars($p1));
// Array
// (
//     [x] => 1.233
//     [y] => 3.445
//     [label] => point #1
// )

?>
```

See also get_class_methods(), get_class_vars()

get_parent_class (PHP 4 >= 4.0.0)

Returns the name of the parent class of an object

```
string get_parent_class (object obj)
```

This function returns the name of the parent class to the class of which the object *obj* is an instance.

See also get_class(), is_subclass_of()

is_subclass_of (PHP 4 >= 4.0.0)

Determines if an object belongs to a subclass of the specified class

```
bool is_subclass_of (object obj, string superclass)
```

This function returns TRUE if the object *obj*, belongs to a class which is a subclass of *superclass*, FALSE otherwise.

See also get_class(), get_parent_class()

method_exists (PHP 4 >= 4.0.0)

Checks if the class method exists

```
bool method_exists (object object, string method_name)
```

This function returns TRUE if the method given by *method_name* has been defined for the given *object*, FALSE otherwise.

X. ClibPDF functions

ClibPDF lets you create PDF documents with PHP. It is available for download from FastIO (http://www.fastio.com/), but requires that you purchase a license for commercial use. ClibPDF functionality and API is similar to PDFlib.

This documentation should be read alongside the ClibPDF manual since it explains the library in much greater detail.

Many functions in the native ClibPDF and the PHP module, as well as in PDFlib, have the same name. All functions except for cpdf_open() take the handle for the document as their first parameter.

Currently this handle is not used internally since ClibPDF does not support the creation of several PDF documents at the same time. Actually, you should not even try it, the results are unpredictable. I can’t oversee what the consequences in a multi threaded environment are. According to the author of ClibPDF this will change in one of the next releases (current version when this was written is 1.10). If you need this functionality use the pdflib module.

> **Nota:** The function cpdf_set_font() has changed since PHP 3 to support asian fonts. The encoding parameter is no longer an integer but a string.

A nice feature of ClibPDF (and PDFlib) is the ability to create the pdf document completely in memory without using temporary files. It also provides the ability to pass coordinates in a predefined unit length. (This feature can also be simulated by pdf_translate() when using the PDFlib. functions.)

Another nice feature of ClibPDF is the fact that any page can be modified at any time even if a new page has been already opened. The function cpdf_set_current_page() allows to leave the current page and presume modifying an other page.

Most of the functions are fairly easy to use. The most difficult part is probably creating a very simple PDF document at all. The following example should help you to get started. It creates a document with one page. The page contains the text "Times-Roman" in an outlined 30pt font. The text is underlined.

Esempio 1. Simple ClibPDF Example

```
<?php
$cpdf = cpdf_open(0);
cpdf_page_init($cpdf, 1, 0, 595, 842);
cpdf_add_outline($cpdf, 0, 0, 0, 1, "Page 1");
cpdf_begin_text($cpdf);
cpdf_set_font($cpdf, "Times-Roman", 30, "WinAnsiEncoding");
cpdf_set_text_rendering($cpdf, 1);
cpdf_text($cpdf, "Times Roman outlined", 50, 750);
cpdf_end_text($cpdf);
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);
?>
```

The pdflib distribution contains a more complex example which creates a series of pages with an analog clock. Here is that example converted into PHP using the ClibPDF extension:

Esempio 2. pdfclock example from pdflib 2.0 distribution

```
<?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 + $margin), 2 * ($ra-
dius + $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);
?>
```

cpdf_add_annotation (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Adds annotation

```
void cpdf_add_annotation (int pdf document, float llx, float lly, float
urx, float ury, string title, string content [, int mode])
```

The **cpdf_add_annotation()** adds a note with the lower left corner at (*llx*, *lly*) and the upper right corner at (*urx*, *ury*).

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

cpdf_add_outline (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Adds bookmark for current page

```
void cpdf_add_outline (int pdf document, string text)
```

The **cpdf_add_outline()** function adds a bookmark with text *text* that points to the current page.

Esempio 1. Adding a page outline

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

cpdf_arc (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Draws an arc

```
void cpdf_arc (int pdf document, float x-coor, float y-coor, float radius,
float start, float end [, int mode])
```

The **cpdf_arc()** function draws an arc with center at point (*x-coor*, *y-coor*) and radius *radius*, starting at angle *start* and ending at angle *end*.

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

See also cpdf_circle().

cpdf_begin_text (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Starts text section

```
void cpdf_begin_text (int pdf document)
```

The **cpdf_begin_text()** function starts a text section. It must be ended with cpdf_end_text().

Esempio 1. Text output

```
<?php
cpdf_begin_text($pdf);
cpdf_set_font($pdf, 16, "Helvetica", "WinAnsiEncoding");
cpdf_text($pdf, 100, 100, "Some text");
cpdf_end_text($pdf)
?>
```

See also cpdf_end_text().

cpdf_circle (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Draw a circle

```
void cpdf_circle (int pdf document, float x-coor, float y-coor, float
radius [, int mode])
```

The **cpdf_circle()** function draws a circle with center at point (*x-coor*, *y-coor*) and radius *radius*.

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

See also cpdf_arc().

cpdf_clip (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Clips to current path

```
void cpdf_clip (int pdf document)
```

The **cpdf_clip()** function clips all drawing to the current path.

cpdf_close (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Closes the pdf document

```
void cpdf_close (int pdf document)
```

The **cpdf_close()** function closes the pdf document. This should be the last function even after cpdf_finalize(), cpdf_output_buffer() and cpdf_save_to_file().

See also cpdf_open().

cpdf_closepath (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Close path

```
void cpdf_closepath (int pdf document)
```

The **cpdf_closepath()** function closes the current path.

cpdf_closepath_fill_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Close, fill and stroke current path

```
void cpdf_closepath_fill_stroke (int pdf document)
```

The **cpdf_closepath_fill_stroke()** function closes, fills the interior of the current path with the current fill color and draws current path.

See also cpdf_closepath(), cpdf_stroke(), cpdf_fill(), cpdf_setgray_fill(), cpdf_setgray(), cpdf_setrgbcolor_fill(), cpdf_setrgbcolor().

cpdf_closepath_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Close path and draw line along path

```
void cpdf_closepath_stroke (int pdf document)
```

The **cpdf_closepath_stroke()** function is a combination of cpdf_closepath() and cpdf_stroke(). Then clears the path.

See also cpdf_closepath(), cpdf_stroke().

cpdf_continue_text (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Output text in next line

```
void cpdf_continue_text (int pdf document, string text)
```

The **cpdf_continue_text()** function outputs the string in *text* in the next line.

See also cpdf_show_xy(), cpdf_text(), cpdf_set_leading(), cpdf_set_text_pos().

cpdf_curveto (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Draws a curve

```
void cpdf_curveto (int pdf document, float x1, float y1, float x2, float
y2, float x3, float y3 [, int mode])
```

The **cpdf_curveto()** function draws a Bezier curve from the current point to the point (*x3*, *y3*) using (*x1*, *y1*) and (*x2*, *y2*) as control points.

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

See also cpdf_moveto(), cpdf_rmoveto(), cpdf_rlineto(), cpdf_lineto().

cpdf_end_text (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Ends text section

```
void cpdf_end_text (int pdf document)
```

The **cpdf_end_text()** function ends a text section which was started with cpdf_begin_text().

Esempio 1. Text output

```
<?php
cpdf_begin_text($pdf);
cpdf_set_font($pdf, 16, "Helvetica", "WinAnsiEncoding");
cpdf_text($pdf, 100, 100, "Some text");
cpdf_end_text($pdf)
?>
```

See also cpdf_begin_text().

cpdf_fill (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Fill current path

```
void cpdf_fill (int pdf document)
```

The **cpdf_fill()** function fills the interior of the current path with the current fill color.

See also cpdf_closepath(), cpdf_stroke(), cpdf_setgray_fill(), cpdf_setgray(), cpdf_setrgbcolor_fill(), cpdf_setrgbcolor().

cpdf_fill_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Fill and stroke current path

```
void cpdf_fill_stroke (int pdf document)
```

The **cpdf_fill_stroke()** function fills the interior of the current path with the current fill color and draws current path.

See also cpdf_closepath(), cpdf_stroke(), cpdf_fill(), cpdf_setgray_fill(), cpdf_setgray(), cpdf_setrgbcolor_fill(), cpdf_setrgbcolor().

cpdf_finalize (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Ends document

```
void cpdf_finalize (int pdf document)
```

The **cpdf_finalize()** function ends the document. You still have to call cpdf_close()

See also cpdf_close().

cpdf_finalize_page (PHP 3>= 3.0.10, PHP 4 >= 4.0.0)

Ends page

```
void cpdf_finalize_page (int pdf document, int page number)
```

The **cpdf_finalize_page()** function ends the page with page number *page number*.

This function is only for saving memory. A finalized page takes less memory but cannot be modified anymore.

See also cpdf_page_init().

cpdf_global_set_document_limits (PHP 4 >= 4.0.0)

Sets document limits for any pdf document

```
void cpdf_global_set_document_limits  (int maxpages, int maxfonts, int
maximages, int maxannotations, int maxobjects)
```

The **cpdf_global_set_document_limits()** function sets several document limits. This function has to be called before cpdf_open() to take effect. It sets the limits for any document open afterwards.

See also cpdf_open().

cpdf_import_jpeg (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Opens a JPEG image

```
int cpdf_import_jpeg (int pdf document, string file name, float x-coor,
float y-coor, float angle, float width, float height, float x-scale, float
y-scale [, int mode])
```

The **cpdf_import_jpeg()** function opens an image stored in the file with the name *file name*. The format of the image has to be jpeg. The image is placed on the current page at position (*x-coor*, *y-coor*). The image is rotated by *angle* degres.

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

See also cpdf_place_inline_image().

cpdf_lineto (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Draws a line

```
void cpdf_lineto (int pdf document, float x-coor, float y-coor [, int
mode])
```

The **cpdf_lineto()** function draws a line from the current point to the point with coordinates (*x-coor*, *y-coor*).

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

See also cpdf_moveto(), cpdf_rmoveto(), cpdf_curveto().

cpdf_moveto (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets current point

```
void cpdf_moveto (int pdf document, float x-coor, float y-coor [, int
mode])
```

The **cpdf_moveto()** function set the current point to the coordinates *x-coor* and *y-coor*.

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

cpdf_newpath (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Starts a new path

```
void cpdf_newpath  (int pdf document)
```

The **cpdf_newpath()** starts a new path on the document given by the *pdf document* parameter.

cpdf_open (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Opens a new pdf document

```
int cpdf_open (int compression [, string filename])
```

The **cpdf_open()** function opens a new pdf document. The first parameter turns document compression on if it is unequal to 0. The second optional parameter sets the file in which the document is written. If it is omitted the document is created in memory and can either be written into a file with the cpdf_save_to_file() or written to standard output with cpdf_output_buffer().

Nota: The return value will be needed in further versions of ClibPDF as the first parameter in all other functions which are writing to the pdf document.

The ClibPDF library takes the filename "-" as a synonym for stdout. If PHP is compiled as an apache module this will not work because the way ClibPDF outputs to stdout does not work with apache. You can solve this problem by skipping the filename and using cpdf_output_buffer() to output the pdf document.

See also cpdf_close(), cpdf_output_buffer().

cpdf_output_buffer (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Outputs the pdf document in memory buffer

```
void cpdf_output_buffer (int pdf document)
```

The **cpdf_output_buffer()** function outputs the pdf document to stdout. The document has to be created in memory which is the case if cpdf_open() has been called with no filename parameter.

See also cpdf_open().

cpdf_page_init (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Starts new page

```
void cpdf_page_init (int pdf document, int page number, int orientation,
float height, float width [, float unit])
```

The **cpdf_page_init()** function starts a new page with height *height* and width *width*. The page has number *page number* and orientation *orientation*. *orientation* can be 0 for portrait and 1 for landscape. The last optional parameter *unit* sets the unit for the coordinate system. The value should be the number of postscript points per unit. Since one inch is equal to 72 points, a value of 72 would set the unit to one inch. The default is also 72.

See also cpdf_set_current_page().

cpdf_place_inline_image (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Places an image on the page

```
void cpdf_place_inline_image (int pdf document, int image, float x-coor,
float y-coor, float angle, float width, float height [, int mode])
```

The **cpdf_place_inline_image()** function places an image created with the php image functions on the page at postion (*x-coor*, *y-coor*). The image can be scaled at the same time.

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

See also cpdf_import_jpeg().

cpdf_rect (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Draw a rectangle

```
void cpdf_rect (int pdf document, float x-coor, float y-coor, float width,
float height [, int mode])
```

The **cpdf_rect()** function draws a rectangle with its lower left corner at point (*x-coor*, *y-coor*). This width is set to *widgth*. This height is set to *height*.

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

cpdf_restore (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Restores formerly saved enviroment

```
void cpdf_restore (int pdf document)
```

The **cpdf_restore()** function restores the enviroment saved with cpdf_save(). It works like the postscript command grestore. Very useful if you want to translate or rotate an object without effecting other objects.

Esempio 1. Save/Restore

```
<?php
cpdf_save($pdf);
// do all kinds of rotations, transformations, ...
cpdf_restore($pdf)
?>
```

See also cpdf_save().

cpdf_rlineto (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Draws a line

```
void cpdf_rlineto (int pdf document, float x-coor, float y-coor [, int mode])
```

The **cpdf_rlineto()** function draws a line from the current point to the relative point with coordinates (*x-coor*, *y-coor*).

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

See also cpdf_moveto(), cpdf_rmoveto(), cpdf_curveto().

cpdf_rmoveto (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Sets current point

```
void cpdf_rmoveto (int pdf document, float x-coor, float y-coor [, int mode])
```

The **cpdf_rmoveto()** function set the current point relative to the coordinates *x-coor* and *y-coor*.

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

See also cpdf_moveto().

cpdf_rotate (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets rotation

```
void cpdf_rotate (int pdf document, float angle)
```

The **cpdf_rotate()** function set the rotation in degress to *angle*.

cpdf_save (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Saves current enviroment

```
void cpdf_save (int pdf document)
```

The **cpdf_save()** function saves the current enviroment. It works like the postscript command gsave. Very useful if you want to translate or rotate an object without effecting other objects.

See also cpdf_restore().

cpdf_save_to_file (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Writes the pdf document into a file

```
void cpdf_save_to_file (int pdf document, string filename)
```

The **cpdf_save_to_file()** function outputs the pdf document into a file if it has been created in memory.

This function is not needed if the pdf document has been open by specifying a filename as a parameter of cpdf_open().

See also cpdf_output_buffer(), cpdf_open().

cpdf_scale (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets scaling

```
void cpdf_scale (int pdf document, float x-scale, float y-scale)
```

The **cpdf_scale()** function set the scaling factor in both directions.

cpdf_set_char_spacing (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets character spacing

```
void cpdf_set_char_spacing (int pdf document, float space)
```

The **cpdf_set_char_spacing()** function sets the spacing between characters.

See also cpdf_set_word_spacing(), cpdf_set_leading().

cpdf_set_creator (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets the creator field in the pdf document

```
void cpdf_set_creator (string creator)
```

The **cpdf_set_creator()** function sets the creator of a pdf document.

See also cpdf_set_subject(), cpdf_set_title(), cpdf_set_keywords().

cpdf_set_current_page (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Sets current page

```
void cpdf_set_current_page (int pdf document, int page number)
```

The **cpdf_set_current_page()** function set the page on which all operations are performed. One can switch between pages until a page is finished with cpdf_finalize_page().

See also cpdf_finalize_page().

cpdf_set_font (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Select the current font face and size

```
void cpdf_set_font (int pdf document, string font name, float size, string
encoding)
```

The **cpdf_set_font()** function sets the current font face, font size and encoding. Currently only the standard postscript fonts are supported.

The last parameter *encoding* can take the following values: "MacRomanEncoding", "MacExpertEncoding", "WinAnsiEncoding", and "NULL". "NULL" stands for the font's built-in encoding.

See the ClibPDF Manual for more information, especially how to support asian fonts.

cpdf_set_horiz_scaling (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets horizontal scaling of text

```
void cpdf_set_horiz_scaling (int pdf document, float scale)
```

The **cpdf_set_horiz_scaling()** function sets the horizontal scaling to *scale* percent.

cpdf_set_keywords (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets the keywords field of the pdf document

```
void cpdf_set_keywords (string keywords)
```

The **cpdf_set_keywords()** function sets the keywords of a pdf document.

See also cpdf_set_title(), cpdf_set_creator(), cpdf_set_subject().

cpdf_set_leading (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets distance between text lines

```
void cpdf_set leading (int pdf document, float distance)
```

The **cpdf_set_leading()** function sets the distance between text lines. This will be used if text is output by cpdf_continue_text().

See also cpdf_continue_text().

cpdf_set_page_animation (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Sets duration between pages

```
void cpdf_set_page_animation (int pdf document, int transition, float
duration)
```

The **cpdf_set_page_animation()** function set the transition between following pages.

The value of *transition* can be

0 for none,
1 for two lines sweeping across the screen reveal the page,
2 for multiple lines sweeping across the screen reveal the page,
3 for a box reveals the page,
4 for a single line sweeping across the screen reveals the page,
5 for the old page dissolves to reveal the page,
6 for the dissolve effect moves from one screen edge to another,
7 for the old page is simply replaced by the new page (default)

The value of *duration* is the number of seconds between page flipping.

cpdf_set_subject (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets the subject field of the pdf document

```
void cpdf_set_subject (string subject)
```

The **cpdf_set_subject()** function sets the subject of a pdf document.

See also cpdf_set_title(), cpdf_set_creator(), cpdf_set_keywords().

cpdf_set_text_matrix (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets the text matrix

```
void cpdf_set_text_matrix (int pdf document, array matrix)
```

The **cpdf_set_text_matrix()** function sets a matrix which describes a transformation applied on the current text font.

cpdf_set_text_pos (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets text position

```
void cpdf_set_text_pos (int pdf document, float x-coor, float y-coor [,
int mode])
```

The **cpdf_set_text_pos()** function sets the position of text for the next cpdf_show() function call.

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

See also cpdf_show(), cpdf_text().

cpdf_set_text_rendering (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Determines how text is rendered

```
void cpdf_set_text_rendering (int pdf document, int mode)
```

The **cpdf_set_text_rendering()** function determines how text is rendered.

The possible values for *mode* are 0=fill text, 1=stroke text, 2=fill and stroke text, 3=invisible, 4=fill text and add it to clipping path, 5=stroke text and add it to clipping path, 6=fill and stroke text and add it to clipping path, 7=add it to clipping path.

cpdf_set_text_rise (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets the text rise

```
void cpdf_set_text_rise (int pdf document, float value)
```

The **cpdf_set_text_rise()** function sets the text rising to *value* units.

cpdf_set_title (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets the title field of the pdf document

```
void cpdf_set_title (string title)
```

The **cpdf_set_title()** function sets the title of a pdf document.

See also cpdf_set_subject(), cpdf_set_creator(), cpdf_set_keywords().

cpdf_set_word_spacing (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets spacing between words

```
void cpdf_set_word_spacing (int pdf document, float space)
```

The **cpdf_set_word_spacing()** function sets the spacing between words.

See also cpdf_set_char_spacing(), cpdf_set_leading().

cpdf_setdash (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets dash pattern

```
void cpdf_setdash (int pdf document, float white, float black)
```

The **cpdf_setdash()** function set the dash pattern *white* white units and *black* black units. If both are 0 a solid line is set.

cpdf_setflat (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets flatness

```
void cpdf_setflat (int pdf document, float value)
```

The **cpdf_setflat()** function set the flatness to a value between 0 and 100.

cpdf_setgray (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets drawing and filling color to gray value

```
void cpdf_setgray (int pdf document, float gray value)
```

The cpdf_setgray_stroke() function sets the current drawing and filling color to the given gray value.

See also cpdf_setrgbcolor_stroke(), cpdf_setrgbcolor_fill().

cpdf_setgray_fill (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets filling color to gray value

```
void cpdf_setgray_fill (int pdf document, float value)
```

The **cpdf_setgray_fill()** function sets the current gray value to fill a path.
See also cpdf_setrgbcolor_fill().

cpdf_setgray_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets drawing color to gray value

```
void cpdf_setgray_stroke (int pdf document, float gray value)
```

The **cpdf_setgray_stroke()** function sets the current drawing color to the given gray value.
See also cpdf_setrgbcolor_stroke().

cpdf_setlinecap (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets linecap parameter

```
void cpdf_setlinecap (int pdf document, int value)
```

The **cpdf_setlinecap()** function set the linecap parameter between a value of 0 and 2. 0 = butt end, 1 = round, 2 = projecting square.

cpdf_setlinejoin (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets linejoin parameter

```
void cpdf_setlinejoin (int pdf document, long value)
```

The **cpdf_setlinejoin()** function set the linejoin parameter between a value of 0 and 2. 0 = miter, 1 = round, 2 = bevel.

cpdf_setlinewidth (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets line width

```
void cpdf_setlinewidth (int pdf document, float width)
```

The **cpdf_setlinewidth()** function set the line width to *width*.

cpdf_setmiterlimit (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets miter limit

```
void cpdf_setmiterlimit (int pdf document, float value)
```

The **cpdf_setmiterlimit()** function set the miter limit to a value greater or equal than 1.

cpdf_setrgbcolor (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets drawing and filling color to rgb color value

```
void cpdf_setrgbcolor (int pdf document, float red value, float green
value, float blue value)
```

The cpdf_setrgbcolor_stroke() function sets the current drawing and filling color to the given rgb color value.

See also cpdf_setrgbcolor_stroke(), cpdf_setrgbcolor_fill().

cpdf_setrgbcolor_fill (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets filling color to rgb color value

```
void cpdf_setrgbcolor_fill (int pdf document, float red value, float green
value, float blue value)
```

The **cpdf_setrgbcolor_fill()** function sets the current rgb color value to fill a path.

See also cpdf_setrgbcolor_stroke(), cpdf_setrgbcolor().

cpdf_setrgbcolor_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets drawing color to rgb color value

```
void cpdf_setrgbcolor_stroke (int pdf document, float red value, float
green value, float blue value)
```

The **cpdf_setrgbcolor_stroke()** function sets the current drawing color to the given rgb color value.

See also cpdf_setrgbcolor_fill(), cpdf_setrgbcolor().

cpdf_show (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Output text at current position

```
void cpdf_show (int pdf document, string text)
```

The **cpdf_show()** function outputs the string in *text* at the current position.

See also cpdf_text(), cpdf_begin_text(), cpdf_end_text().

cpdf_show_xy (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Output text at position

```
void cpdf_show_xy (int pdf document, string text, float x-coor, float
y-coor [, int mode])
```

The **cpdf_show_xy()** function outputs the string *text* at position with coordinates (*x-coor*, *y-coor*).

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

Nota: The function **cpdf_show_xy()** is identical to cpdf_text() without the optional parameters.

See also cpdf_text().

cpdf_stringwidth (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Returns width of text in current font

```
float cpdf_stringwidth (int pdf document, string text)
```

The **cpdf_stringwidth()** function returns the width of the string in *text*. It requires a font to be set before.

See also cpdf_set_font().

cpdf_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Draw line along path

```
void cpdf_stroke (int pdf document)
```

The **cpdf_stroke()** function draws a line along current path.

See also cpdf_closepath(), cpdf_closepath_stroke().

cpdf_text (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Output text with parameters

```
void cpdf_text (int pdf document, string text, float x-coor, float y-coor
[, int mode [, float orientation [, int alignmode]]])
```

The **cpdf_text()** function outputs the string *text* at position with coordinates (*x-coor*, *y-coor*).

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit. The optional parameter *orientation* is the rotation of the text in degree. The optional parameter *alignmode* determines how the text is aligned.

See the ClibPDF documentation for possible values.

See also cpdf_show_xy().

cpdf_translate (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets origin of coordinate system

```
void cpdf_translate (int pdf document, float x-coor, float y-coor [, int mode])
```

The **cpdf_translate()** function set the origin of coordinate system to the point (*x-coor*, *y-coor*).

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

XI. CURL, Client URL Library Functions

PHP supports libcurl, a library created by Daniel Stenberg, that allows you to connect and communicate to many different types of servers with many different types of protocols. libcurl currently supports the http, https, ftp, gopher, telnet, dict, file, and ldap protocols. libcurl also supports HTTPS certificates, HTTP POST, HTTP PUT, FTP uploading (this can also be done with PHP's ftp extension), HTTP form based upload, proxies, cookies, and user+password authentication.

In order to use the CURL functions you need to install the CURL (http://curl.haxx.se/) package. PHP requires that you use CURL 7.0.2-beta or higher. PHP will not work with any version of CURL below version 7.0.2-beta.

To use PHP's CURL support you must also compile PHP `--with-curl[=DIR]` where DIR is the location of the directory containing the lib and include directories. In the "include" directory there should be a folder named "curl" which should contain the easy.h and curl.h files. There should be a file named "libcurl.a" located in the "lib" directory.

These functions have been added in PHP 4.0.2.

Once you've compiled PHP with CURL support, you can begin using the curl functions. The basic idea behind the CURL functions is that you initialize a CURL session using the curl_init(), then you can set all your options for the transfer via the curl_exec() and then you finish off your session using the curl_close(). Here is an example that uses the CURL functions to fetch the PHP homepage into a file:

Esempio 1. Using PHP's CURL module to fetch the PHP homepage

```
<?php

$ch = curl_init ("http://www.php.net/");
$fp = fopen ("php_homepage.txt", "w");

curl_setopt ($ch, CURLOPT_FILE, $fp);
curl_setopt ($ch, CURLOPT_HEADER, 0);

curl_exec ($ch);
curl_close ($ch);
fclose ($fp);
?>
```

curl_init (PHP 4 >= 4.0.2)

Initialize a CURL session

```
int curl_init  ([string url])
```

The **curl_init()** will initialize a new session and return a CURL handle for use with the curl_setopt(), curl_exec(), and curl_close() functions. If the optional *url* parameter is supplied then the CURLOPT_URL option will be set to the value of the parameter. You can manually set this using the curl_setopt() function.

Esempio 1. Initializing a new CURL session and fetching a webpage

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

See also: curl_close(), curl_setopt()

curl_setopt (PHP 4 >= 4.0.2)

Set an option for a CURL transfer

```
bool curl_setopt  (int ch, string option, mixed value)
```

The **curl_setopt()** function will set options for a CURL session identified by the *ch* parameter. The *option* parameter is the option you want to set, and the *value* is the value of the option given by the *option*.

The *value* should be a long for the following options (specified in the *option* parameter):

- *CURLOPT_INFILESIZE*: When you are uploading a file to a remote site, this option should be used to tell PHP what the expected size of the infile will be.

- `CURLOPT_VERBOSE`: Set this option to a non-zero value if you want CURL to report everything that is happening.
- `CURLOPT_HEADER`: Set this option to a non-zero value if you want the header to be included in the output.
    - `CURLOPT_NOPROGRESS`: Set this option to a non-zero value if you don't want PHP to display a progress meter for CURL transfers

    Nota: PHP automatically sets this option to a non-zero parameter, this should only be changed for debugging purposes.

- `CURLOPT_NOBODY`: Set this option to a non-zero value if you don't want the body included with the output.
- `CURLOPT_FAILONERROR`: Set this option to a non-zero value if you want PHP to fail silently if the HTTP code returned is greater than 300. The default behaviour is to return the page normally, ignoring the code.
- `CURLOPT_UPLOAD`: Set this option to a non-zero value if you want PHP to prepare for an upload.
- `CURLOPT_POST`: Set this option to a non-zero value if you want PHP to do a regular HTTP POST. This POST is a normal application/x-www-from-urlencoded kind, most commonly used by HTML forms.
- `CURLOPT_FTPLISTONLY`: Set this option to a non-zero value and PHP will just list the names of an FTP directory.
- `CURLOPT_FTPAPPEND`: Set this option to a non-zero value and PHP will append to the remote file instead of overwriting it.
- `CURLOPT_NETRC`: Set this option to a non-zero value and PHP will scan your ~./netrc file to find your username and password for the remote site that you're establishing a connection with.
- `CURLOPT_FOLLOWLOCATION`: Set this option to a non-zero value to follow any "Location: " header that the server sends as a part of the HTTP header (note this is recursive, PHP will follow as many "Location: " headers that it is sent.)
- `CURLOPT_PUT`: Set this option a non-zero value to HTTP PUT a file. The file to PUT must be set with the CURLOPT_INFILE and CURLOPT_INFILESIZE.
- `CURLOPT_MUTE`: Set this option to a non-zero value and PHP will be completely silent with regards to the CURL functions.
- `CURLOPT_TIMEOUT`: Pass a long as a parameter that contains the maximum time, in seconds, that you'll allow the curl functions to take.
- `CURLOPT_LOW_SPEED_LIMIT`: Pass a long as a parameter that contains the transfer speed in bytes per second that the transfer should be below during CURLOPT_LOW_SPEED_TIME seconds for PHP to consider it too slow and abort.

- *`CURLOPT_LOW_SPEED_TIME`*: Pass a long as a parameter that contains the time in seconds that the transfer should be below the CURLOPT_LOW_SPEED_LIMIT for PHP to consider it too slow and abort.
- *`CURLOPT_RESUME_FROM`*: Pass a long as a parameter that contains the offset, in bytes, that you want the transfer to start from.
- *`CURLOPT_SSLVERSION`*: Pass a long as a parameter that contains the SSL version (2 or 3) to use. By default PHP will try and determine this by itself, although, in some cases you must set this manually.
- *`CURLOPT_TIMECONDITION`*: Pass a long as a parameter that defines how the CURLOPT_TIMEVALUE is treated. You can set this parameter to TIMECOND_IFMODSINCE or TIMECOND_ISUNMODSINCE. This is a HTTP-only feature.
- *`CURLOPT_TIMEVALUE`*: Pass a long as a parameter that is the time in seconds since January 1st, 1970. The time will be used as specified by the CURLOPT_TIMEVALUE option, or by default the TIMECOND_IFMODSINCE will be used.

The *`value`* parameter should be a string for the following values of the *`option`* parameter:

- *`CURLOPT_URL`*: This is the URL that you want PHP to fetch. You can also set this option when initializing a session with the curl_init() function.
- *`CURLOPT_USERPWD`*: Pass a string formatted in the [username]:[password] manner, for PHP to use for the connection. connection.
- *`CURLOPT_PROXYUSERPWD`*: Pass a string formatted in the [username]:[password] format for connection to the HTTP proxy.
- *`CURLOPT_RANGE`*: Pass the specified range you want. It should be in the "X-Y" format, where X or Y may be left out. The HTTP transfers also support several intervals, seperated with commas as in X-Y,N-M.
- *`CURLOPT_POSTFIELDS`*: Pass a string containing the full data to post in an HTTP "POST" operation.
- *`CURLOPT_REFERER`*: Pass a string containing the "referer" header to be used in an HTTP request.
- *`CURLOPT_USERAGENT`*: Pass a string containing the "user-agent" header to be used in an HTTP request.
- *`CURLOPT_FTPPORT`*: Pass a string containing the which will be used to get the IP address to use for the ftp "POST" instruction. The POST instruction tells the remote server to connect to our specified IP address. The string may be a plain IP address, a hostname, a network interface name (under UNIX), or just a plain '-' to use the systems default IP address.
- *`CURLOPT_COOKIE`*: Pass a string containing the content of the cookie to be set in the HTTP header.
- *`CURLOPT_SSLCERT`*: Pass a string containing the filename of PEM formatted certificate.

- *CURLOPT_SSLCERTPASSWD*: Pass a string containing the password required to use the CURLOPT_SSLCERT certificate.
- *CURLOPT_COOKIEFILE*: Pass a string containing the name of the file containing the cookiee data. The cookie file can be in Netscape format, or just plain HTTP-style headers dumped into a file.
    - *CURLOPT_CUSTOMREQUEST*: Pass a string to be used instead of GET or HEAD when doing an HTTP request. This is useful for doing DELETE or another, more obscure, HTTP request.

    Nota: Don't do this without making sure your server supports the command first.

The following options expect a file descriptor that is obtained by using the fopen() function:

- *CURLOPT_FILE*: The file where the output of your transfer should be placed, the default is STDOUT.
- *CURLOPT_INFILE*: The file where the input of your transfer comes from.
- *CURLOPT_WRITEHEADER*: The file to write the header part of the output into.
- *CURLOPT_STDERR*: The file to write errors to instead of stderr.

curl_exec (PHP 4 >= 4.0.2)

Perform a CURL session

```
bool curl_exec  (int ch)
```

This function is should be called after you initialize a CURL session and all the options for the session are set. Its purpose is simply to execute the predefined CURL session (given by the *ch*).

curl_close (PHP 4 >= 4.0.2)

Close a CURL session

```
void curl_close  (int ch)
```

This functions closes a CURL session and frees all ressources. The CURL handle, *ch*, is also deleted.

curl_version (PHP 4 >= 4.0.2)

Return the current CURL version

```
string curl_version  ()
```

The **curl_version()** function returns a string containing the current CURL version.

XII. Cybercash payment functions

These functions are only available if the interpreter has been compiled with the `--with-cybercash=[DIR]`. These functions have been added in PHP 4.

cybercash_encr (PHP 4 >= 4.0.0)

???

```
array cybercash_encr (string wmk, string sk, string inbuff)
```

The function returns an associative array with the elements "errcode" and, if "errcode" is FALSE, "outbuff" (string), "outLth" (long) and "macbuff" (string).

cybercash_decr (PHP 4 >= 4.0.0)

???

```
array cybercash_decr (string wmk, string sk, string inbuff)
```

The function returns an associative array with the elements "errcode" and, if "errcode" is FALSE, "outbuff" (string), "outLth" (long) and "macbuff" (string).

cybercash_base64_encode (PHP 4 >= 4.0.0)

???

```
string cybercash_base64_encode  (string inbuff)
```

cybercash_base64_decode (PHP 4 >= 4.0.0)

```
string cybercash_base64_decode  (string inbuff)
```

XIII. Crédit Mutuel CyberMUT functions

This extension allows you to process credit cards transactions using Crédit Mutuel CyberMUT system (http://www.creditmutuel.fr/centre_commercial/vendez_sur_internet.html).

CynerMUT is a popular Web Payment Service in France, provided by the Crédit Mutuel bank. If you are foreign in France, these functions will not be useful for you.

These functions are only available if PHP has been compiled with the `--with-cybermut[=DIR]` option, where DIR is the location of `libcm-mac.a` and `cm-mac.h`. You will require the appropriate SDK for your platform, which may be sent to you after your CyberMUT's subscription (contact them via Web, or go to the nearest Crédit Mutuel).

The use of these functions is almost identical to the original functions, except for the parameters of return for cybermut_creerformulairecm() and cybermut_creerreponsecm(), which are returned directly by functions PHP, whereas they had passed in reference in the original functions.

These functions have been added in PHP 4.0.6.

> **Nota:** These functions only provide a link to CyberMUT SDK. Be sure to read the CyberMUT Developers Guide for full details of the required parameters.

cybermut_creerformulairecm (PHP 4 >= 4.0.5)

Generate HTML form of request for payment

```
string cybermut_creerformulairecm (string url_CM, string version, string
TPE, string montant, string ref_commande, string texte_libre, string
url_retour, string url_retour_ok, string url_retour_err, string langue,
string code_societe, string texte_bouton)
```

cynermut_creerformulairecm() is used to generate the HTML form of request for payment.

Esempio 1. First step of payment (equiv cgi1.c)

```
<?php
// Directory where are located the keys
putenv("CMKEYDIR=/var/creditmut/cles");

// Version number
$VERSION="1.2";

  $retour =  creditmut_creerformulairecm(
  "https://www.creditmutuel.fr/test/telepaiement/paiement.cgi",
  $VERSION,
  "1234567890",
  "300FRF",
  $REFERENCE,
  $TEXTE_LIBRE,
  $URL_RETOUR,
  $URL_RETOUR_OK,
  $URL_RETOUR_ERR,
  "francais",
  "company",
  "Paiement par carte bancaire");

  echo $retour;
?>
```

See also cybermut_testmac() and cybermut_creerreponsecm().

cybermut_testmac (PHP 4 >= 4.0.5)

Make sure that there no was data diddling contained in the received message of confirmation

bool **cybermut_testmac** (string *code_MAC*, string *version*, string *TPE*, string *cdate*, string *montant*, string *ref_commande*, string *texte_libre*, string *code-retour*)

cybermut_testmac() is used to make sure that there was not data diddling contained in the received message of confirmation. Pay attention to parameters code-retour and texte-libre, which cannot be evaluated as is, because auf the dash. You must retrieve them by using:

```
<?php
  $code_retour=$HTTP_GET_VARS["code-retour"];
  $texte_libre=$HTTP_GET_VARS["texte-libre"];
?>
```

Esempio 1. Last step of payment (equiv cgi2.c)

```
<?php
// Make sure that Enable Track Vars is ON.
// Directory where are located the keys
putenv("CMKEYDIR=/var/creditmut/cles");

// Version number
$VERSION="1.2";

$texte_libre = $HTTP_GET_VARS["texte-libre"];
$code_retour = $HTTP_GET_VARS["code-retour"];

$mac_ok = creditmut_testmac($MAC,$VERSION,$TPE,$date,$montant,$reference,$texte_libre,$co

if ($mac_ok) {

  //
  // insert data processing here
  //
  //

  $result=creditmut_creerreponsecm("OK");
  } else {
  $result=creditmut_creerreponsecm("Document Falsifie");}

?>
```

See also cybermut_creerformulairecm() and cybermut_creerreponsecm().

cybermut_creerreponsecm (PHP 4 >= 4.0.5)

Generate the acknowledgement of delivery of the confirmation of payment

```
string cybermut_creerreponsecm (string phrase)
```

cybermut_creerreponsecm() returns a string containing delivery aknowledgement message.

The parameter is "OK" if the message of confirmation of the payment were correctly auhentified by cybermut_testmac(). Any other chain is regarded as an error message.

See also cybermut_creerformulairecm() and cybermut_testmac().

XIV. Character type functions

> **Attenzione**
>
> Questo modulo è *SPERIMENTALE*. Ovvero, il comportamento di queste funzioni, i nomi di queste funzioni, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

These functions check whether a character or string falls into a certain character class according to the current locale.

When called with an integer argument these functions behave exactly like their C counterparts.

When called with a string argument they will check every character in the string and will only return `TRUE` if every character in the string matches the requested criteria.

Passing anything else but a string or integer will return `FALSE` immediately.

> **Attenzione**
>
> These functions are new as of PHP 4.0.4 and might change their name in the near future. Suggestions are to change them to **ctype_issomething()** instead of **ctype_something()** or even to make them part of `ext/standard` and use their original C-names, although this would possibly lead to further confusion regarding the isset() vs. **is_sometype()** problem.

ctype_alnum (PHP 4 >= 4.0.4)

Check for alphanumeric character(s)

```
bool ctype_alnum (string c)
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

See also setlocale().

ctype_alpha (PHP 4 >= 4.0.4)

Check for alphabetic character(s)

```
bool ctype_alpha (string c)
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

ctype_cntrl (PHP 4 >= 4.0.4)

Check for control character(s)

```
bool ctype_cntrl (string c)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

ctype_digit (PHP 4 >= 4.0.4)

Check for numeric character(s)

```
bool ctype_digit (string c)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

ctype_lower (PHP 4 >= 4.0.4)

Check for lowercase character(s)

```
bool ctype_lower (string c)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

ctype_graph (PHP 4 >= 4.0.4)

Check for any printable character(s) except space

```
bool ctype_graph (string c)
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

ctype_print (PHP 4 >= 4.0.4)

Check for printable character(s)

```
bool ctype_print (string c)
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

ctype_punct (PHP 4 >= 4.0.4)

Check for any printable character which is not whitespace or an alphanumeric character

```
bool ctype_punct (string c)
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

ctype_space (PHP 4 >= 4.0.4)

Check for whitespace character(s)

```
bool ctype_space (string c)
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

ctype_upper (PHP 4 >= 4.0.4)

Check for uppercase character(s)

```
bool ctype_upper (string c)
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

ctype_xdigit (PHP 4 >= 4.0.4)

Check for character(s) representing a hexadecimal digit

```
bool ctype_xdigit (string c)
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

XV. Database (dbm-style) abstraction layer functions

These functions build the foundation for accessing Berkeley DB style databases.

This is a general abstraction layer for several file-based databases. As such, functionality is limited to a common subset of features supported by modern databases such as Sleepycat Software's DB2 (http://www.sleepycat.com/). (This is not to be confused with IBM's DB2 software, which is supported through the ODBC functions.)

The behaviour of various aspects depends on the implementation of the underlying database. Functions such as dba_optimize() and dba_sync() will do what they promise for one database and will do nothing for others.

When invoking the dba_open() or dba_popen() functions, one of the following handler names must be supplied as an argument. The actually available list of handlers is displayed by invoking phpinfo(). (To add support for any of the following handlers during the production of PHP, add the specified `--with-XXXX` configure switch to your PHP configure line.)

Tabella 1. List of DBA handlers

Handler	Notes

Esempio 1. DBA example

```
<?php

$id = dba_open ("/tmp/test.db", "n", "db2");

if (!$id) {
    echo "dba_open failed\n";
    exit;
}

dba_replace ("key", "This is an example!", $id);

if (dba_exists ("key", $id)) {
    echo dba_fetch ("key", $id);
    dba_delete ("key", $id);
}

dba_close ($id);
?>
```

DBA is binary safe and does not have any arbitrary limits. However, it inherits all limits set by the underlying database implementation.

All file-based databases must provide a way of setting the file mode of a new created database, if that is possible at all. The file mode is commonly passed as the fourth argument to dba_open() or dba_popen().

You can access all entries of a database in a linear way by using the dba_firstkey() and dba_nextkey() functions. You may not change the database while traversing it.

Esempio 2. Traversing a database

```
<?php

# ...open database...

$key = dba_firstkey ($id);

while ($key != false) {
    if (...) { # remember the key to perform some action later
        $handle_later[] = $key;
    }
    $key = dba_nextkey ($id);
}

for ($i = 0; $i < count($handle_later); $i++)
    dba_delete ($handle_later[$i], $id);

?>
```

dba_close (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Close database

```
void dba_close (int handle)
```

dba_close() closes the established database and frees all resources specified by *handle*.
handle is a database handle returned by dba_open().
dba_close() does not return any value.
See also: dba_open() and dba_popen()

dba_delete (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Delete entry specified by key

```
bool dba_delete (string key, int handle)
```

dba_delete() deletes the entry specified by *key* from the database specified with *handle*.
key is the key of the entry which is deleted.
handle is a database handle returned by dba_open().
dba_delete() returns TRUE or FALSE, if the entry is deleted or not deleted, respectively.
See also: dba_exists(), dba_fetch(), dba_insert(), and dba_replace().

dba_exists (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Check whether key exists

```
bool dba_exists (string key, int handle)
```

dba_exists() checks whether the specified *key* exists in the database specified by *handle*.
Key is the key the check is performed for.
Handle is a database handle returned by dba_open().
dba_exists() returns TRUE or FALSE, if the key is found or not found, respectively.
See also: dba_fetch(), dba_delete(), dba_insert(), and dba_replace().

dba_fetch (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Fetch data specified by key

```
string dba_fetch (string key, int handle)
```

dba_fetch() fetches the data specified by *key* from the database specified with *handle*.

Key is the key the data is specified by.

Handle is a database handle returned by dba_open().

dba_fetch() returns the associated string or FALSE, if the key/data pair is found or not found, respectively.

See also: dba_exists(), dba_delete(), dba_insert(), and dba_replace().

dba_firstkey (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Fetch first key

```
string dba_firstkey (int handle)
```

dba_firstkey() returns the first key of the database specified by *handle* and resets the internal key pointer. This permits a linear search through the whole database.

Handle is a database handle returned by dba_open().

dba_firstkey() returns the key or FALSE depending on whether it succeeds or fails, respectively.

See also: dba_nextkey() and example 2 in the DBA overview

dba_insert (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Insert entry

```
bool dba_insert (string key, string value, int handle)
```

dba_insert() inserts the entry described with *key* and *value* into the database specified by *handle*. It fails, if an entry with the same *key* already exists.

key is the key of the entry to be inserted.

value is the value to be inserted.

handle is a database handle returned by dba_open().

dba_insert() returns TRUE or FALSE, depending on whether it succeeds of fails, respectively.

See also: dba_exists() dba_delete() dba_fetch() dba_replace()

dba_nextkey (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Fetch next key

```
string dba_nextkey (int handle)
```

dba_nextkey() returns the next key of the database specified by *handle* and advances the internal key pointer.

handle is a database handle returned by dba_open().

dba_nextkey() returns the key or FALSE depending on whether it succeeds or fails, respectively.

See also: dba_firstkey() and example 2 in the DBA overview

dba_popen (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Open database persistently

```
int dba_popen (string path, string mode, string handler [, ...])
```

dba_popen() establishes a persistent database instance for *path* with *mode* using *handler*.

path is commonly a regular path in your filesystem.

mode is "r" for read access, "w" for read/write access to an already existing database, "c" for read/write access and database creation if it doesn't currently exist, and "n" for create, truncate and read/write access.

handler is the name of the handler which shall be used for accessing *path*. It is passed all optional parameters given to **dba_popen()** and can act on behalf of them.

dba_popen() returns a positive handle or FALSE, in the case the open is successful or fails, respectively.

See also: dba_open() dba_close()

dba_open (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Open database

```
int dba_open (string path, string mode, string handler [, ...])
```

dba_open() establishes a database instance for *path* with *mode* using *handler*.

path is commonly a regular path in your filesystem.

mode is "r" for read access, "w" for read/write access to an already existing database, "c" for read/write access and database creation if it doesn't currently exist, and "n" for create, truncate and read/write access.

handler is the name of the handler which shall be used for accessing *path*. It is passed all optional parameters given to **dba_open()** and can act on behalf of them.

dba_open() returns a positive handle or FALSE, in the case the open is successful or fails, respectively.

See also: dba_popen() dba_close()

dba_optimize (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Optimize database

```
bool dba_optimize (int handle)
```

dba_optimize() optimizes the underlying database specified by *handle*.

handle is a database handle returned by dba_open().

dba_optimize() returns TRUE or FALSE, if the optimization succeeds or fails, respectively.

See also: dba_sync()

dba_replace (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Replace or insert entry

```
bool dba_replace (string key, string value, int handle)
```

dba_replace() replaces or inserts the entry described with *key* and *value* into the database specified by *handle*.

key is the key of the entry to be inserted.

value is the value to be inserted.

handle is a database handle returned by dba_open().

dba_replace() returns TRUE or FALSE, depending on whether it succeeds of fails, respectively.

See also: dba_exists(), dba_delete(), dba_fetch(), and dba_insert().

dba_sync (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Synchronize database

```
bool dba_sync (int handle)
```

dba_sync() synchronizes the database specified by *handle*. This will probably trigger a physical write to disk, if supported.

handle is a database handle returned by dba_open().

dba_sync() returns TRUE or FALSE, if the synchronization succeeds or fails, respectively.

See also: dba_optimize()

XVI. Date and Time functions

checkdate (PHP 3, PHP 4 >= 4.0.0)

Validate a gregorian date/time

```
int checkdate (int month, int day, int year)
```

Returns TRUE if the date given is valid; otherwise returns FALSE. Checks the validity of the date formed by the arguments. A date is considered valid if:

- year is between 1 and 32767 inclusive
- month is between 1 and 12 inclusive
- *Day* is within the allowed number of days for the given *month*. Leap *year*s are taken into consideration.

date (PHP 3, PHP 4 >= 4.0.0)

Format a local time/date

```
string date (string format [, int timestamp])
```

Returns a string formatted according to the given format string using the given *timestamp* or the current local time if no timestamp is given.

The following characters are recognized in the format string:

- a - "am" or "pm"
- A - "AM" or "PM"
- B - Swatch Internet time
- d - day of the month, 2 digits with leading zeros; i.e. "01" to "31"
- D - day of the week, textual, 3 letters; i.e. "Fri"
- F - month, textual, long; i.e. "January"
- g - hour, 12-hour format without leading zeros; i.e. "1" to "12"
- G - hour, 24-hour format without leading zeros; i.e. "0" to "23"
- h - hour, 12-hour format; i.e. "01" to "12"
- H - hour, 24-hour format; i.e. "00" to "23"
- i - minutes; i.e. "00" to "59"
- I (capital i) - "1" if Daylight Savings Time, "0" otherwise.

- j - day of the month without leading zeros; i.e. "1" to "31"
- l (lowercase 'L') - day of the week, textual, long; i.e. "Friday"
- L - boolean for whether it is a leap year; i.e. "0" or "1"
- m - month; i.e. "01" to "12"
- M - month, textual, 3 letters; i.e. "Jan"
- n - month without leading zeros; i.e. "1" to "12"
- r - RFC 822 formatted date; i.e. "Thu, 21 Dec 2000 16:01:07 +0200" (added in PHP 4.0.4)
- s - seconds; i.e. "00" to "59"
- S - English ordinal suffix, textual, 2 characters; i.e. "th", "nd"
- t - number of days in the given month; i.e. "28" to "31"
- T - Timezone setting of this machine; i.e. "MDT"
- U - seconds since the epoch
- w - day of the week, numeric, i.e. "0" (Sunday) to "6" (Saturday)
- Y - year, 4 digits; i.e. "1999"
- y - year, 2 digits; i.e. "99"
- z - day of the year; i.e. "0" to "365"
- Z - timezone offset in seconds (i.e. "-43200" to "43200"). The offset for timezones west of UTC is always negative, and for those east of UTC is always positive.

Unrecognized characters in the format string will be printed as-is. The "Z" format will always return "0" when using gmdate().

Esempio 1. date() example

```
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)));
```

It is possible to use **date()** and mktime() together to find dates in the future or the past.

Esempio 2. date() and mktime() example

```
$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);
```

Some examples of **date()** formatting. Note that you should escape any other characters, as any which currently have a special meaning will produce undesirable results, and other characters may be

assigned meaning in future PHP versions. When escaping, bu sure to use single quotes to prevent characters like \n from become newlines.

Esempio 3. date() Formatting

```
/* Today is March 10th, 2001, 5:16:18 pm */
$today = date("F j, Y, g:i a");                 // March 10, 2001, 5:16 pm
$today = date("m.d.y");                         // 03.10.01
$today = date("j, m, Y");                       // 10, 3, 2001
$today = date("Ymd");                           // 20010310
$today = date('h-i-s, j-m-y, it is w Day z ');  // 05-16-17, 10-03-
01, 1631 1618 6 Fripm01
$today = date('\i\t \i\s \t\h\e jS \d\a\y.');   // It is the 10th day.
$today = date("D M j G:i:s T Y");               // Sat Mar 10 15:16:08 MST 2001
$today = date('H:m:s \m \i\s\ \m\o\n\t\h');     // 17:03:17 m is month
$today = date("H:i:s");                         // 17:16:17
```

To format dates in other languages, you should use the setlocale() and strftime() functions.

See also gmdate(), mktime() and strftime().

getdate (PHP 3, PHP 4 >= 4.0.0)

Get date/time information

```
array getdate ([int timestamp])
```

Returns an associative array containing the date information of the *timestamp*, or the current local time if no timestamp is given, as the following array elements:

- "seconds" - seconds
- "minutes" - minutes
- "hours" - hours
- "mday" - day of the month
- "wday" - day of the week, numeric : from 0 as Sunday up to 6 as Saturday
- "mon" - month, numeric
- "year" - year, numeric
- "yday" - day of the year, numeric; i.e. "299"
- "weekday" - day of the week, textual, full; i.e. "Friday"
- "month" - month, textual, full; i.e. "January"

Esempio 1. getdate() example

```
$today = getdate();
$month = $today['month'];
$mday = $today['mday'];
$year = $today['year'];
echo "$month $mday, $year";
```

gettimeofday (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Get current time

```
array gettimeofday ()
```

This is an interface to gettimeofday(2). It returns an associative array containing the data returned from the system call.

- "sec" - seconds
- "usec" - microseconds
- "minuteswest" - minutes west of Greenwich
- "dsttime" - type of dst correction

gmdate (PHP 3, PHP 4 >= 4.0.0)

Format a GMT/CUT date/time

```
string gmdate (string format [, int timestamp])
```

Identical to the date() function except that the time returned is Greenwich Mean Time (GMT). For example, when run in Finland (GMT +0200), the first line below prints "Jan 01 1998 00:00:00", while the second prints "Dec 31 1997 22:00:00".

Esempio 1. gmdate() example

```
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));
```

See also date(), mktime(), gmmktime() and strftime()

gmmktime (PHP 3, PHP 4 >= 4.0.0)

Get UNIX timestamp for a GMT date

```
int gmmktime (int hour, int minute, int second, int month, int day, int
year [, int is_dst])
```

Identical to mktime() except the passed parameters represents a GMT date.

gmstrftime (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Format a GMT/CUT time/date according to locale settings

```
string gmstrftime (string format [, int timestamp])
```

Behaves the same as strftime() except that the time returned is Greenwich Mean Time (GMT). For example, when run in Eastern Standard Time (GMT -0500), the first line below prints "Dec 31 1998 20:00:00", while the second prints "Jan 01 1999 01:00:00".

Esempio 1. gmstrftime() example

```
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";
```

See also strftime().

localtime (PHP 4 >= 4.0.0)

Get the local time

```
array localtime ([int timestamp [, bool is_associative]])
```

The **localtime()** function returns an array identical to that of the structure returned by the C function call. The first argument to **localtime()** is the timestamp, if this is not given the current time is used.

The second argument to the **localtime()** is the *is_associative*, if this is set to 0 or not supplied than the array is returned as a regular, numerically indexed array. If the argument is set to 1 then **localtime()** is an associative array containing all the different elements of the structure returned by the C function call to localtime. The names of the different keys of the associative array are as follows:

- "tm_sec" - seconds
- "tm_min" - minutes
- "tm_hour" - hour
- "tm_mday" - day of the month
- "tm_mon" - month of the year
- "tm_year" - Years since 1900
- "tm_wday" - Day of the week
- "tm_yday" - Day of the year
- "tm_isdst" - Is daylight savings time in effect

microtime (PHP 3, PHP 4 >= 4.0.0)

Return current UNIX timestamp with microseconds

```
string microtime ()
```

Returns the string "msec sec" where sec is the current time measured in the number of seconds since the Unix Epoch (0:00:00 January 1, 1970 GMT), and msec is the microseconds part. This function is only available on operating systems that support the gettimeofday() system call.

Both portions of the string are returned in units of seconds.

Esempio 1. microtime() example

```
function getmicrotime(){
    list($usec, $sec) = explode(" ",microtime());
    return ((float)$usec + (float)$sec);
    }

$time_start = getmicrotime();

for ($i=0; $i < 1000; $i++){
    //do nothing, 1000 times
    }
```

```
$time_end = getmicrotime();
$time = $time_end - $time_start;

echo "Did nothing in $time seconds";
```

See also time().

mktime (PHP 3, PHP 4 >= 4.0.0)

Get UNIX timestamp for a date

```
int mktime (int hour, int minute, int second, int month, int day, int year
[, int is_dst])
```

Warning: Note the strange order of arguments, which differs from the order of arguments in a regular UNIX mktime() call and which does not lend itself well to leaving out parameters from right to left (see below). It is a common error to mix these values up in a script.

Returns the Unix timestamp corresponding to the arguments given. This timestamp is a long integer containing the number of seconds between the Unix Epoch (January 1 1970) and the time specified.

Arguments may be left out in order from right to left; any arguments thus omitted will be set to the current value according to the local date and time.

Is_dst can be set to 1 if the time is during daylight savings time, 0 if it is not, or -1 (the default) if it is unknown whether the time is within daylight savings time or not.

> **Nota:** *Is_dst* was added in 3.0.10.

mktime() is useful for doing date arithmetic and validation, as it will automatically calculate the correct value for out-of-range input. For example, each of the following lines produces the string "Jan-01-1998".

Esempio 1. mktime() example

```
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 may be a two or four digit value, with values between 0-69 mapping to 2000-2069 and 70-99 to 1970-1999 (on systems where time_t is a 32bit signed integer, as most common today, the valid range for *year* is somewhere between 1902 and 2037).

The last day of any given month can be expressed as the "0" day of the next month, not the -1 day. Both of the following examples will produce the string "The last day in Feb 2000 is: 29".

Esempio 2. Last day of next month

```
$lastday = mktime (0,0,0,3,0,2000);
echo strftime ("Last day in Feb 2000 is: %d", $lastday);

$lastday = mktime (0,0,0,4,-31,2000);
echo strftime ("Last day in Feb 2000 is: %d", $lastday);
```

Date with year, month and day equal to zero is considered illegal (otherwise it what be regarded as 30.11.1999, which would be strange behaviour).

See also date() and time().

strftime (PHP 3, PHP 4 >= 4.0.0)

Format a local time/date according to locale settings

```
string strftime (string format [, int timestamp])
```

Returns a string formatted according to the given format string using the given *timestamp* or the current local time if no timestamp is given. Month and weekday names and other language dependent strings respect the current locale set with setlocale().

The following conversion specifiers are recognized in the format string:

- %a - abbreviated weekday name according to the current locale
- %A - full weekday name according to the current locale
- %b - abbreviated month name according to the current locale
- %B - full month name according to the current locale
- %c - preferred date and time representation for the current locale
- %C - century number (the year divided by 100 and truncated to an integer, range 00 to 99)
- %d - day of the month as a decimal number (range 01 to 31)
- %D - same as %m/%d/%y

- %e - day of the month as a decimal number, a single digit is preceded by a space (range ’ 1’ to ’31’)
- %h - same as %b
- %H - hour as a decimal number using a 24-hour clock (range 00 to 23)
- %I - hour as a decimal number using a 12-hour clock (range 01 to 12)
- %j - day of the year as a decimal number (range 001 to 366)
- %m - month as a decimal number (range 01 to 12)
- %M - minute as a decimal number
- %n - newline character
- %p - either ‘am’ or ‘pm’ according to the given time value, or the corresponding strings for the current locale
- %r - time in a.m. and p.m. notation
- %R - time in 24 hour notation
- %S - second as a decimal number
- %t - tab character
- %T - current time, equal to %H:%M:%S
- %u - weekday as a decimal number [1,7], with 1 representing Monday
- %U - week number of the current year as a decimal number, starting with the first Sunday as the first day of the first week
- %V - The ISO 8601:1988 week number of the current year as a decimal number, range 01 to 53, where week 1 is the first week that has at least 4 days in the current year, and with Monday as the first day of the week.
- %W - week number of the current year as a decimal number, starting with the first Monday as the first day of the first week
- %w - day of the week as a decimal, Sunday being 0
- %x - preferred date representation for the current locale without the time
- %X - preferred time representation for the current locale without the date
- %y - year as a decimal number without a century (range 00 to 99)
- %Y - year as a decimal number including the century
- %Z - time zone or name or abbreviation
- %% - a literal ‘%’ character

Nota: Not all conversion specifiers may be supported by your C library, in which case they will not be supported by PHP’s **strftime()**.

Esempio 1. strftime() example

```
setlocale ("LC_TIME", "C");
print (strftime ("%A in Finnish is "));
setlocale ("LC_TIME", "fi_FI");
print (strftime ("%A, in French "));
setlocale ("LC_TIME", "fr_CA");
print (strftime ("%A and in German "));
setlocale ("LC_TIME", "de_DE");
print (strftime ("%A.\n"));
```

This example works if you have the respective locales installed in your system.

See also setlocale() and mktime() and the Open Group specification of **strftime()** (http://www.opengroup.org/onlinepubs/7908799/xsh/strftime.html).

time (PHP 3, PHP 4 >= 4.0.0)

Return current UNIX timestamp

```
int time ()
```

Returns the current time measured in the number of seconds since the Unix Epoch (January 1 1970 00:00:00 GMT).

See also date().

strtotime (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Parse about any english textual datetime description into a UNIX timestamp

```
int strtotime (string time [, int now])
```

The function expects to be given a string containing an english date format and will try to parse that format into a UNIX timestamp.

Esempio 1. strtotime() examples

```
echo strtotime ("now") . "\n";
echo strtotime ("10 September 2000") . "\n";
echo strtotime ("+1 day") . "\n";
echo strtotime ("+1 week") . "\n";
```

```
echo strtotime ("+1 week 2 days 4 hours 2 seconds") . "\n";
```

XVII. dBase functions

These functions allow you to access records stored in dBase-format (dbf) databases.

There is no support for indexes or memo fields. There is no support for locking, too. Two concurrent webserver processes modifying the same dBase file will very likely ruin your database.

dBase files are simple sequential files of fixed length records. Records are appended to the end of the file and delete records are kept until you call dbase_pack().

We recommend that you do not use dBase files as your production database. Choose any real SQL server instead; MySQL or Postgres are common choices with PHP. dBase support is here to allow you to import and export data to and from your web database, because the file format is commonly understood by Windows spreadsheets and organizers.

dbase_create (PHP 3, PHP 4 >= 4.0.0)

Creates a dBase database

```
int dbase_create (string filename, array fields)
```

The *fields* parameter is an array of arrays, each array describing the format of one field in the database. Each field consists of a name, a character indicating the field type, a length, and a precision.

The types of fields available are:

L

Boolean. These do not have a length or precision.

M

Memo. (Note that these aren't supported by PHP.) These do not have a length or precision.

D

Date (stored as YYYYMMDD). These do not have a length or precision.

N

Number. These have both a length and a precision (the number of digits after the decimal point).

C

String.

If the database is successfully created, a dbase_identifier is returned, otherwise FALSE is returned.

Esempio 1. Creating a dBase database file

```
// "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")
    );
```

```
// creation
if (!dbase_create($dbname, $def))
    print "<strong>Error!</strong>";
```

dbase_open (PHP 3, PHP 4 >= 4.0.0)

Opens a dBase database

```
int dbase_open (string filename, int flags)
```

The flags correspond to those for the open() system call. (Typically 0 means read-only, 1 means write-only, and 2 means read and write.)

Returns a dbase_identifier for the opened database, or FALSE if the database couldn't be opened.

> **Nota:** Quando safe-mode è abilitato, PHP controlla che i file o le directory sulle quali stai andando a lavorare, abbiano lo stesso UID dello script che è in esecuzione.

dbase_close (PHP 3, PHP 4 >= 4.0.0)

Close a dBase database

```
bool dbase_close (int dbase_identifier)
```

Closes the database associated with *dbase_identifier*.

dbase_pack (PHP 3, PHP 4 >= 4.0.0)

Packs a dBase database

```
bool dbase_pack (int dbase_identifier)
```

Packs the specified database (permanently deleting all records marked for deletion using dbase_delete_record()).

dbase_add_record (PHP 3, PHP 4 >= 4.0.0)

Add a record to a dBase database

```
bool dbase_add_record (int dbase_identifier, array record)
```

Adds the data in the *record* to the database. If the number of items in the supplied record isn't equal to the number of fields in the database, the operation will fail and FALSE will be returned.

dbase_replace_record (PHP 3>= 3.0.11, PHP 4 >= 4.0.0)

Replace a record in a dBase database

```
bool dbase_replace_record (int dbase_identifier, array record, int
dbase_record_number)
```

Replaces the data associated with the record *record_number* with the data in the *record* in the database. If the number of items in the supplied record is not equal to the number of fields in the database, the operation will fail and FALSE will be returned.

dbase_record_number is an integer which spans from 1 to the number of records in the database (as returned by dbase_numrecords()).

dbase_delete_record (PHP 3, PHP 4 >= 4.0.0)

Deletes a record from a dBase database

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

Marks *record* to be deleted from the database. To actually remove the record from the database, you must also call dbase_pack().

dbase_get_record (PHP 3, PHP 4 >= 4.0.0)

Gets a record from a dBase database

```
array dbase_get_record (int dbase_identifier, int record)
```

Returns the data from *record* in an array. The array is indexed starting at 0, and includes an associative member named 'deleted' which is set to 1 if the record has been marked for deletion (see dbase_delete_record().

Each field is converted to the appropriate PHP type. (Dates are left as strings.)

dbase_get_record_with_names (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Gets a record from a dBase database as an associative array

```
array dbase_get_record_with_names  (int dbase_identifier, int record)
```

Returns the data from *record* in an associative array. The array also includes an associative member named 'deleted' which is set to 1 if the record has been marked for deletion (see dbase_delete_record()).

Each field is converted to the appropriate PHP type. (Dates are left as strings.)

dbase_numfields (PHP 3, PHP 4 >= 4.0.0)

Find out how many fields are in a dBase database

```
int dbase_numfields (int dbase_identifier)
```

Returns the number of fields (columns) in the specified database. Field numbers are between 0 and dbase_numfields($db)-1, while record numbers are between 1 and dbase_numrecords($db).

Esempio 1. Using dbase_numfields()

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

dbase_numrecords (PHP 3, PHP 4 >= 4.0.0)

Find out how many records are in a dBase database

```
int dbase_numrecords (int dbase_identifier)
```

Returns the number of records (rows) in the specified database. Record numbers are between 1 and dbase_numrecords($db), while field numbers are between 0 and dbase_numfields($db)-1.

XVIII. DBM Functions

These functions allow you to store records stored in a dbm-style database. This type of database (supported by the Berkeley DB, GDBM, and some system libraries, as well as a built-in flatfile library) stores key/value pairs (as opposed to the full-blown records supported by relational databases).

Esempio 1. DBM example

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

dbmopen (PHP 3, PHP 4 >= 4.0.0)

Opens a DBM database

```
int dbmopen (string filename, string flags)
```

The first argument is the full-path filename of the DBM file to be opened and the second is the file open mode which is one of "r", "n", "c" or "w" for read-only, new (implies read-write, and most likely will truncate an already-existing database of the same name), create (implies read-write, and will not truncate an already-existing database of the same name) and read-write respectively.

Returns an identifer to be passed to the other DBM functions on success, or FALSE on failure.

If NDBM support is used, NDBM will actually create filename.dir and filename.pag files. GDBM only uses one file, as does the internal flat-file support, and Berkeley DB creates a `filename.db` file. Note that PHP does its own file locking in addition to any file locking that may be done by the DBM library itself. PHP does not delete the `.lck` files it creates. It uses these files simply as fixed inodes on which to do the file locking. For more information on DBM files, see your Unix man pages, or obtain GNU's GDBM (ftp://ftp.gnu.org/pub/gnu/gdbm/).

> **Nota:** Quando safe-mode è abilitato, PHP controlla che i file o le directory sulle quali stai andando a lavorare, abbiano lo stesso UID dello script che è in esecuzione.

dbmclose (PHP 3, PHP 4 >= 4.0.0)

Closes a dbm database

```
bool dbmclose (int dbm_identifier)
```

Unlocks and closes the specified database.

dbmexists (PHP 3, PHP 4 >= 4.0.0)

Tells if a value exists for a key in a DBM database

```
bool dbmexists (int dbm_identifier, string key)
```

Returns TRUE if there is a value associated with the *key*.

dbmfetch (PHP 3, PHP 4 >= 4.0.0)

Fetches a value for a key from a DBM database

```
string dbmfetch (int dbm_identifier, string key)
```

Returns the value associated with *key*.

dbminsert (PHP 3, PHP 4 >= 4.0.0)

Inserts a value for a key in a DBM database

```
int dbminsert (int dbm_identifier, string key, string value)
```

Adds the value to the database with the specified key.

Returns -1 if the database was opened read-only, 0 if the insert was successful, and 1 if the specified key already exists. (To replace the value, use dbmreplace().)

dbmreplace (PHP 3, PHP 4 >= 4.0.0)

Replaces the value for a key in a DBM database

```
bool dbmreplace (int dbm_identifier, string key, string value)
```

Replaces the value for the specified key in the database.

This will also add the key to the database if it didn't already exist.

dbmdelete (PHP 3, PHP 4 >= 4.0.0)

Deletes the value for a key from a DBM database

```
bool dbmdelete (int dbm_identifier, string key)
```

Deletes the value for *key* in the database.

Returns FALSE if the key didn't exist in the database.

dbmfirstkey (PHP 3, PHP 4 >= 4.0.0)

Retrieves the first key from a DBM database

```
string dbmfirstkey (int dbm_identifier)
```

Returns the first key in the database. Note that no particular order is guaranteed since the database may be built using a hash-table, which doesn't guarantee any ordering.

dbmnextkey (PHP 3, PHP 4 >= 4.0.0)

Retrieves the next key from a DBM database

```
string dbmnextkey (int dbm_identifier, string key)
```

Returns the next key after *key*. By calling dbmfirstkey() followed by successive calls to **dbmnextkey()** it is possible to visit every key/value pair in the dbm database. For example:

Esempio 1. Visiting every key/value pair in a DBM database

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

dblist (PHP 3, PHP 4 >= 4.0.0)

Describes the DBM-compatible library being used

```
string dblist ()
```

XIX. dbx functions

Attenzione

Questo modulo è *SPERIMENTALE*. Ovvero, il comportamento di queste funzioni, i nomi di queste funzioni, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

The dbx module is a database abstraction layer (db 'X', where 'X' is a supported database). The dbx functions allow you to access all supported databases using a single calling convention. In order to have these functions available, you must compile PHP with dbx support by using the `--enable-dbx` option and all options for the databases that will be used, e.g. for MySQL you must also specify `--with-mysql`. The dbx-functions themselves do not interface directly to the databases, but interface to the modules that are used to support these databases. To be able to use a database with the dbx-module, the module must be either linked or loaded into PHP, and the database module must be supported by the dbx-module. Currently, MySQL, PostgreSQL, Microsoft SQL Server, FrontBase and ODBC are supported, but others will follow (soon, I hope :-).

Documentation for adding additional database support to dbx can be found at http://www.guidance.nl/php/dbx/doc/.

dbx_close (PHP 4 >= 4.0.6)

Close an open connection/database

```
bool dbx_close (dbx_link_object link_identifier)
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

Returns TRUE on success, FALSE on error.

Esempio 1. dbx_close() example

```
<?php
$link = dbx_connect ("mysql", "localhost", "db", "username", "password")
    or die ("Could not connect");
print("Connected successfully");
dbx_close($link);
?>
```

Nota: Always refer to the module-specific documentation as well.

See also: dbx_connect().

dbx_connect (PHP 4 >= 4.0.6)

Open a connection/database

```
dbx_link_object dbx_connect (string module, string host, string database,
string username, string password [, int persistent])
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

Returns: a dbx_link_object on success, FALSE on error. If a connection has been made but the database could not be selected, the connection is closed and FALSE is returned. The 'persistent' parameter can be set to DBX_PERSISTENT so a persistent connection will be created.

The *`module`* parameter can be either a string or a constant. The possible values are given below, but keep in mind that they only work if the module is actually loaded.

- module DBX_MYSQL : "mysql"
- module DBX_ODBC : "odbc"
- module DBX_PGSQL : "pgsql"
- module DBX_MSSQL : "mssql"
- module DBX_FBSQL : "fbsql" (CVS only)

The dbx_link_object has three members, a 'handle', a 'module' and a 'database'. The 'database' member is the name of the currently selected database. The 'module' member is for internal use by dbx only, and is actually the module number mentioned above. The 'handle' member is a valid handle for the connected database, and as such can be used in module-specific functions (if required), e.g.

```
<?php
$link = dbx_connect ("mysql", "localhost", "db", "username", "password");
mysql_close ($link->handle); // dbx_close($link) would be better here
?>
```

Host, database, username and password parameters are expected, but not always used, depending on the connect-functions for the abstracted module.

Esempio 1. dbx_connect() example

```
<?php
$link = dbx_connect ("odbc", "", "db", "username", "pass-
word", DBX_PERSISTENT)
    or die ("Could not connect");
print ("Connected successfully");
dbx_close ($link);
```

```
?>
```

Nota: Always refer to the module-specific documentation as well.

See also: dbx_close().

dbx_error (PHP 4 >= 4.0.6)

Report the error message of the latest function call in the module (not just in the connection)

```
string dbx_error (dbx_link_object link_identifier)
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

Returns a string containing the error-message from the last function call of the module (e.g. mysql-module). If there are multiple connections on the same module, just the last error is given. If there are connections on different modules, the latest error is returned for the specified module (specified by the link parameter, that is). Note that the ODBC-module doesn't support an error_reporting function at the moment.

Esempio 1. dbx_error() example

```
<?php
$link = dbx_connect ("mysql", "localhost", "db", "username", "password")
    or die ("Could not connect");
$result = dbx_query ($link, "select id from nonexistingtbl");
if ($result==0) {
    echo dbx_error ($link);
}
dbx_close ($link);
?>
```

Nota: Always refer to the module-specific documentation as well.

The error-message for Microsoft SQL Server is actually the result of the mssql_get_last_message() function.

dbx_query (PHP 4 >= 4.0.6)

Send a query and fetch all results (if any)

```
dbx_result_object dbx_query (dbx_link_object link_identifier, string
sql_statement [, long flags])
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

Returns a dbx_result_object or 1 on success (a result object is only returned for sql-statements that return results) or 0 on failure. The `flags` parameter is used to control the amount of information that is returned. It may be any combination of the constants DBX_RESULT_INFO, DBX_RESULT_INDEX, DBX_RESULT_ASSOC, OR-ed together. DBX_RESULT_INFO provides info about columns, such as field names and field types. DBX_RESULT_INDEX returns the results in a 2d indexed array (e.g. data[2][3], where 2 is the row (or record) number and 3 is the column (or field) number), where the first row and column are indexed at 0. DBX_RESULT_ASSOC associates the column indices with field names. Note that DBX_RESULT_INDEX is always returned, regardless of the `flags` parameter. If DBX_RESULT_ASSOC is specified, DBX_RESULT_INFO is also returned even if it wasn't specified. This means that effectively only the combinations DBX_RESULT_INDEX, DBX_RESULT_INDEX | DBX_RESULT_INFO and DBX_RESULT_INDEX | DBX_RESULT_INFO | DBX_RESULT_ASSOC are possible. This last combination is the default if the `flags` parameter isn't specified. Associated results are actual references to the indexed data, so if you modify `data[0][0]`, then `data[0]['fieldnameforfirstcolumn']` is modified as well.

A dbx_result_object has five members (possibly four depending on `flags`), 'handle', 'cols', 'rows', 'info' (optional) and 'data'. Handle is a valid result identifier for the specified module, and as such can be used in module-specific functions, as seen in the example:

```
$result = dbx_query ($link, "SELECT id FROM tbl");
mysql_field_len ($result->handle, 0);
```

The cols and rows members contain the number of columns (or fields) and rows (or records) respectively, e.g.

```
$result = dbx_query ($link, "SELECT id FROM tbl");
echo "result size: " . $result->rows . " x " . $result->cols . "<br>\n";
```

The info member is only returned if DBX_RESULT_INFO and/or DBX_RESULT_ASSOC are specified in the *`flags`* parameter. It is a 2d array, that has two named rows ("name" and "type") to retrieve column information, e.g.

```
$result = dbx_query ($link, "SELECT id FROM tbl");
echo "column name: " . $result->info["name"][0] . "<br>\n";
echo "column type: " . $result->info["type"][0] . "<br>\n";
```

The data member contains the actual resulting data, possibly associated with column names as well. If DBX_RESULT_ASSOC is set, it is possible to use `$result->data[2]["fieldname"]`.

Esempio 1. dbx_query() example

```
<?php
$link = dbx_connect ("odbc", "", "db", "username", "password")
    or die ("Could not connect");
$result = dbx_query ($link, "SELECT id, parentid, description FROM tbl");
if ($result==0) echo "Query failed\n<br>";
elseif ($result==1) {
    echo "Query executed successfully\n<br>";
} else {
    $rows=$result->rows;
    $cols=$result->cols;
    echo "<p>table dimension: {$result->rows} x {$result-
>cols}<br><table border=1>\n";
    echo "<tr>";
    for ($col=0; $col<$cols; ++$col) {
        echo "<td>-{$result->info["name"][$col]}-<br>-{$result-
>info["type"][$col]}-</td>";
    }
    echo "</tr>\n";
    for ($row=0; $row<$rows; ++$row){
        echo "<tr>";
        for ($col=0; $col<$cols; ++$col) {
            echo "<td>-{$result->data[$row][$col]}-</td>";
        }
        echo "</tr>\n";
    }
```

```
    echo "</table><p>\n";
    echo "table dimension: {$result-
>rows} x id, parentid, description<br><table border=1>\n";
    for ($row=0; $row<$rows; ++$row) {
        echo "<tr>";
        echo "<td>-{$result->data[$row]["id"]}-</td>";
        echo "<td>-{$result->data[$row]["parentid"]}-</td>";
        echo "<td>-{$result->data[$row]["description"]}-</td>";
        echo "</tr>\n";
    }
    echo "</table><p>\n";
}
dbx_close($link);
?>
```

Nota: Always refer to the module-specific documentation as well.

See also: dbx_connect().

dbx_sort (PHP 4 >= 4.0.6)

Sort a result from a dbx_query by a custom sort function

```
bool dbx_sort (dbx_result_object result, string user_compare_function)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

Returns TRUE on success, FALSE on error.

Esempio 1. dbx_sort() example

```
<?php
function user_re_order ($a, $b) {
    $rv = dbx_compare ($a, $b, "parentid", DBX_CMP_DESC);
    if (!$rv) $rv = dbx_compare ($a, $b, "id");
    return $rv;
}
```

```
$link = dbx_connect ("odbc", "", "db", "username", "password")
    or die ("Could not connect");
$result = dbx_query ($link, "SELECT id, parentid, description FROM tbl OR-
DER BY id");
echo "resulting data is now ordered by id<br>";
dbx_sort ($result, "user_re_order");
echo "resulting data is now ordered by parentid (descend-
ing), then by id<br>";
dbx_close ($link);
?>
```

See also dbx_compare().

dbx_compare (PHP 4 >= 4.0.7RC1)

Compare two rows for sorting purposes

```
int dbx_compare (array row_a, array row_b, string columnname_or_index [,
int flags])
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

Returns 0 if row_a[$columnname_or_index] is equal to row_b[$columnname_or_index], 1 if it is greater and -1 if it is smaller (or vice versa if the DBX_CMP_DESC flag is set).

The *flags* can be set to specify comparison direction (whether sorting is ascending or descending) and comparison type (force string or numeric compare by converting the data). The constants for direction are DBX_CMP_ASC and DBX_CMP_DESC. The constants for comparison type are DBX_CMP_NATIVE (no conversion), DBX_CMP_TEXT and DBX_CMP_NUMBER. These constants can be OR-ed together. The default value for the *flags* parameter is DBX_CMP_ASC | DBX_CMP_NATIVE.

Esempio 1. dbx_compare() example

```
<?php
function user_re_order ($a, $b) {
    $rv = dbx_compare ($a, $b, "parentid", DBX_CMP_DESC);
```

```
    if (!$rv) {
        $rv = dbx_compare ($a, $b, "id");
        return $rv;
    }
}

$link = dbx_connect ("odbc", "", "db", "username", "password")
    or die ("Could not connect");
$result = dbx_query ($link, "SELECT id, parentid, description FROM tbl OR-
DER BY id");
echo "resulting data is now ordered by id<br>";
dbx_sort ($result, "user_re_order");
echo "resulting data is now ordered by parentid (descend-
ing), then by id<br>";
dbx_close ($link);
?>
```

See also dbx_sort().

XX. DB++ Functions

Attenzione

Questo modulo è *SPERIMENTALE*. Ovvero, il comportamento di queste funzioni, i nomi di queste funzioni, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

DB++ Database Functions

db++, made by the german company Concept asa (http://www.concept-asa.de/), is a relational database system with high performance and low memory and disk usage in mind. While providing SQL as an additional language interface it is not really a SQL database in the first place but provides its own AQL query language which is much more influenced by the relational algebra then SQL is.

Concept asa always had an interest in supporting open source languages, db++ has had Perl and Tcl call interfaces for years now and uses Tcl as its internal stored procedure language.

Requirements

You need the developement libraries and header files included in every db++ installaion archive. Concept asa (http://www.concept-asa.de/) provides additional documentation (http://www.concept-asa.de/downloads/doc-eng.tar.gz) and Demo versions (http://www.concept-asa.de/down-eng.html) of db++ for Linux, some other UNIX versions and Windows95/NT.

Installation

Creation and installation of this extension requires the db++ client libraries and header files to be installed on your system. You have to run **configure** with option `--with-dbplus` to build this extension.

configure looks for the client libraries and header files under the default pathes `/usr/dbplus`, `/usr/local/dbplus` and `/opt/dblus`. If you have installed db++ in a different place you have add the installation path to the **configure** option like this: `--with-dbplus=/your/installation/path`.

db++ error codes

Tabella 1. DB++ Error Codes

PHP Constant	db++ constant	meaning

dbplus_add (PHP 4 >= 4.0.7RC1)

Add a tuple to a relation

```
int dbplus_add (resource relation, array tuple)
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

This function will add a tuple to a relation. The *tuple* data is an array of attribute/value pairs to be inserted into the given *relation*. After successfull execution the *tuple* array will contain the complete data of the newly created tuple, including all implicitly set domain fields like sequences.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

dbplus_aql (PHP 4 >= 4.0.7RC1)

Perform AQL query

```
resource dbplus_aql (string query [, string server [, string dbpath]])
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

dbplus_aql() will execute an AQL *query* on the given *server* and *dbpath*.

On success it will return a relation handle. The result data may be fetched from this relation by calling dbplus_next() and **dbplus_current()**. Other relation access functions will not work on a result relation.

Further information on the AQL A... Query Language is provided in the opriginal db++ manual.

dbplus_chdir (PHP 4 >= 4.0.7RC1)

Get/Set database virtual current directory

```
string dbplus_chdir ([string newdir])
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

dbplus_chdir() will change the virtual current directory where relation files will be looked for by dbplus_open(). **dbplus_chdir()** will return the absolute path of the current directory. Calling **dbplus_chdir()** without giving any *newdir* may be used to query the current working directory.

dbplus_close (PHP 4 >= 4.0.7RC1)

Close a relation

```
int dbplus_close (resource relation)
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

Calling **dbplus_close()** will close a relation previously opened by dbplus_open().

dbplus_curr (PHP 4 >= 4.0.7RC1)

Get current tuple from relation

```
int dbplus_curr (resource relation, array tuple)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

dbplus_curr() will read the data for the current tuple for the given *relation* and will pass it back as an associative array in *tuple*.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

See also dbplus_first(), dbplus_prev(), dbplus_next(), and dbplus_last().

dbplus_errcode (PHP 4 >= 4.0.7RC1)

Get error string for given errorcode or last error

```
string dbplus_errcode (int errno)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

dbplus_errcode() returns a cleartext error string for the error code passed as *errno* of for the result code of the last db++ operation if no parameter is given.

dbplus_errno (PHP 4 >= 4.0.7RC1)

Get error code for last operation

```
int dbplus_errno ()
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

dbplus_errno() will return the error code returned by the last db++ operation.

See also dbplus_errcode().

dbplus_find (PHP 4 >= 4.0.7RC1)

Set a constraint on a relation

```
int dbplus_find (resource relation, array constraints, mixed tuple)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

dbplus_find() will place a constraint on the given relation. Further calls to functions like dbplus_curr() or dbplus_next() will only return tuples matching the given constraints.

Constraints are triplets of strings containing of a domain name, a comparison operator and a comparison value. The *constraints* parameter array may consist of a collection of string arrays, each of which contains a domain, an operator and a value, or of a single string array containing a multiple of three elements.

The comparison operator may be one of the following strings: '==', '>', '>=', '<', '<=', '!=', '~' for a regular expression match and 'BAND' or 'BOR' for bitwise operations.

See also dbplus_unselect().

dbplus_first (PHP 4 >= 4.0.7RC1)

Get first tuple from relation

```
int dbplus_first (resource relation, array tuple)
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

dbplus_curr() will read the data for the first tuple for the given *`relation`*, make it the current tuple and pass it back as an associative array in *`tuple`*.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

See also dbplus_curr(), dbplus_prev(), dbplus_next(), and dbplus_last().

dbplus_flush (PHP 4 >= 4.0.7RC1)

Flush all changes made on a relation

```
int dbplus_flush (resource relation)
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

dbplus_flush() will write all changes applied to *`relation`* since the last flush to disk.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

dbplus_freealllocks (PHP 4 >= 4.0.7RC1)

Free all locks held by this client

```
int dbplus_freealllocks ()
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

dbplus_freeaalllocks() will free all tuple locks held by this client.

See also dbplus_getlock(), dbplus_freelock(), and dbplus_freerlocks().

dbplus_freelock (PHP 4 >= 4.0.7RC1)

Release write lock on tuple

```
int dbplus_freelock (resource relation, string tname)
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

dbplus_freelock() will release a write lock on the given `tuple` previously obtained by dbplus_getlock().

See also dbplus_getlock(), dbplus_freerlocks(), and dbplus_freealllocks().

dbplus_freerlocks (PHP 4 >= 4.0.7RC1)

Free all tuple locks on given relation

```
int dbplus_freerlocks (resource relation)
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

dbplus_freerlocks() will free all tuple locks held on the given *relation*.

See also dbplus_getlock(), dbplus_freelock(), and dbplus_freealllocks().

dbplus_getlock (PHP 4 >= 4.0.7RC1)

Get a write lock on a tuple

```
int dbplus_getlock (resource relation, string tname)
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

dbplus_getlock() will request a write lock on the speified *tuple*. It will return zero on success or a non-zero error code, especially DBPLUS_ERR_WLOCKED, on failure.

See also dbplus_freelock(), dbplus_freerlocks(), and dbplus_freealllocks().

dbplus_getunique (PHP 4 >= 4.0.7RC1)

Get a id number unique to a relation

```
int dbplus_getunique (resource relation, int uniqueid)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

dbplus_getunique() will obtain a number guaranteed to be unique for the given *`relation`* and will pass it back in the variable given as *`uniqueid`*.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

dbplus_info (PHP 4 >= 4.0.7RC1)

???

```
int dbplus_info (resource relation, string key, array )
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

Not implemented yet.

dbplus_last (PHP 4 >= 4.0.7RC1)

Get last tuple from relation

```
int dbplus_last (resource relation, array tuple)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

dbplus_curr() will read the data for the last tuple for the given *relation*, make it the current tuple and pass it back as an associative array in *tuple*.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

See also dbplus_first(), dbplus_curr(), dbplus_prev(), and dbplus_next().

dbplus_lockrel (unknown)

Request write lock on relation

```
int dbplus_lockrel (resource relation)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

dbplus_lockrel() will request a write lock on the given relation. Other clients may still query the relation, but can't alter it while it is locked.

dbplus_next (PHP 4 >= 4.0.7RC1)

Get next tuple from relation

```
int dbplus_next (resource relation, array )
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

dbplus_curr() will read the data for the next tuple for the given *relation*, will make it the current tuple and will pass it back as an associative array in *tuple*.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

See also dbplus_first(), dbplus_curr(), dbplus_prev(), and dbplus_last().

dbplus_open (PHP 4 >= 4.0.7RC1)

Open relation file

```
resource dbplus_open (string name)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

The relation file *name* will be opened. *name* can be either a file name or a relative or absolute path name. This will be mapped in any case to an absolute relation file path on a specific host machine and server.

On success a relation file resource (cursor) is returned which must be used in any subsequent commanads referencing the relation. Failure leads to a zero return value, the actual error code may be asked for by calling dbplus_errno().

dbplus_prev (PHP 4 >= 4.0.7RC1)

Get previous tuple from relation

```
int dbplus_prev (resource relation, array tuple)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

dbplus_curr() will read the data for the next tuple for the given *relation*, will make it the current tuple and will pass it back as an associative array in *tuple*.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

See also dbplus_first(), dbplus_curr(), dbplus_next(), and dbplus_last().

dbplus_rchperm (PHP 4 >= 4.0.7RC1)

Change relation permissions

```
int dbplus_rchperm (resource relation, int mask, string user, string
group)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

dbplus_rchperm() will change access permissions as specified by *mask*, *user* and *group*. The values for these are operating system specific.

dbplus_rcreate (PHP 4 >= 4.0.7RC1)

```
resource dbplus_rcreate (string name, mixed domlist [, boolean overwrite])
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

dbplus_rcreate() will create a new relation named *`name`*. An existing relation by the same name will only be overwritten if the relation is currently not in use and *`overwrite`* is set to TRUE.

`domlist` should contain the domain specification for the new relation within an array of domain description strings. (**dbplus_rcreate()** will also accept a string with space delimited domain description strings, but it is recommended to use an array). A domain description string consists of a domain name unique to this relation, a slash and a type specification character. See the db++ documentatiion, especially the dbcreate(1) manpage, for a description of available type specifiers and their meanings.

dbplus_rcrtexact (PHP 4 >= 4.0.7RC1)

```
resource dbplus_rcrtexact (string name, resource relation, boolean
overwrite)
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

dbplus_rcrtexact() will create an exact but empty copy of the given *`relation`* under a new *`name`*. An existing relation by the same *`name`* will only be overwritten if *`overwrite`* is TRUE and no other process is currently using the relation.

dbplus_rcrtlike (PHP 4 >= 4.0.7RC1)

```
resource dbplus_rcrtlike (string name, resource relation, int flag)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

dbplus_rcrtexact() will create an empty copy of the given *relation* under a new *name*, but with default indices. An existing relation by the same *name* will only be overwritten if *overwrite* is TRUE and no other process is currently using the relation.

dbplus_resolve (PHP 4 >= 4.0.7RC1)

Resolve host information for relation

```
int dbplus_resolve (string relation_name)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

dbplus_resolve() will try to resolve the given *relation_name* and find out internal server id, real hostname and the database path on this host. The function will return an array containing these values under the keys 'sid', 'host' and 'host_path' or FALSE on error.

See also dbplus_tcl().

dbplus_rkeys (PHP 4 >= 4.0.7RC1)

Specify new primary key for a relation

```
resource dbplus_rkeys (resource relation, mixed domlist)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

dbplus_rkeys() will replace the current primary key for *relation* with the combination of domains specified by *domlist*.

domlist may be passed as a single domain name string or as an array of domain names.

dbplus_restorepos (PHP 4 >= 4.0.7RC1)

???

```
int dbplus_restorepos (resource relation, array tuple)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

Not implemented yet.

dbplus_ropen (PHP 4 >= 4.0.7RC1)

Open relation file local

```
resource dbplus_ropen (string name)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

dbplus_ropen() will open the relation *file* localy for quick access without any client/server overhead. Access is read only and only **dbplus_current()** and dbplus_next() may be applied to the returned relation.

dbplus_rquery (PHP 4 >= 4.0.7RC1)

Perform local (raw) AQL query

```
int dbplus_rquery (string query, string dbpath)
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

dbplus_rquery() performs a local (raw) AQL query using an AQL interpreter embedded into the db++ client library. **dbplus_rquery()** is faster than dbplus_aql() but will work on local data only.

dbplus_rrename (PHP 4 >= 4.0.7RC1)

Rename a relation

```
int dbplus_rrename (resource relation, string name)
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

dbplus_rrename() will change the name of *relation* to *name*.

dbplus_rsecindex (PHP 4 >= 4.0.7RC1)

Create a new secondary index for a relation

```
resource dbplus_rsecindex (resource relation, mixed domlist, int type)
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

dbplus_rsecindex() will create a new secondary index for *relation* with consists of the domains specified by *domlist* and is of type *type*

domlist may be passed as a single domain name string or as an array of domain names.

dbplus_runlink (PHP 4 >= 4.0.7RC1)

Remove relation from filesystem

```
int dbplus_runlink (resource relation)
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

dbplus_unlink() will close and remove the *relation*.

dbplus_rzap (PHP 4 >= 4.0.7RC1)

Remove all tuples from relation

```
int dbplus_rzap (resource relation)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

dbplus_rzap() will remove all tuples from *relation*.

dbplus_savepos (PHP 4 >= 4.0.7RC1)

???

```
int dbplus_savepos (resource relation)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

Not implemented yet.

dbplus_setindex (PHP 4 >= 4.0.7RC1)

???

```
int dbplus_setindex (resource relation, string idx_name)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

Not implemented yet.

dbplus_setindexbynumber (PHP 4 >= 4.0.7RC1)

???

```
int dbplus_setindexbynumber  (resource relation, int idx_number)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

Not implemented yet.

dbplus_sql (PHP 4 >= 4.0.7RC1)

Perform SQL query

```
resource dbplus_sql (string query, string server, string dbpath)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

Not implemented yet.

dbplus_tcl (PHP 4 >= 4.0.7RC1)

Execute TCL code on server side

```
int dbplus_tcl (int sid, string script)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

A db++ server will prepare a TCL interpreter for each client connection. This interpreter will enable the server to execute TCL code provided by the client as a sort of stored procedures to improve the performance of database operations by avoiding client/server data transfers and context switches.

dbplus_tcl() needs to pass the client connection id the TCL *`script`* code should be executed by. dbplus_resolve() will provide this connection id. The function will return whatever the TCL code returns or a TCL error message if the TCL code fails.

See also dbplus_resolve().

dbplus_tremove (PHP 4 >= 4.0.7RC1)

Remove tuple and return new current tuple

```
int dbplus_tremove (resource relation, array tuple [, array current])
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

dbplus_tremove() removes *`tuple`* from *`relation`* if it perfectly matches a tuple within the relation. *`current`*, if given, will contain the data of the new current tuple after calling **dbplus_tremove()**.

dbplus_undo (PHP 4 >= 4.0.7RC1)

???

```
int dbplus_undo (resource relation)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

Not implemented yet.

dbplus_undoprepare (PHP 4 >= 4.0.7RC1)

???

```
int dbplus_undoprepare (resource relation)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

Not implemented yet.

dbplus_unlockrel (PHP 4 >= 4.0.7RC1)

Give up write lock on relation

```
int dbplus_unlockrel (resource relation)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

dbplus_unlockrel() will release a write lock previously obtained by dbplus_lockrel().

dbplus_unselect (PHP 4 >= 4.0.7RC1)

Remove a constraint from relation

```
int dbplus_unselect (resource relation)
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

Calling **dbplus_unselect()** will remove a constraint previously set by dbplus_find() on `relation`.

dbplus_update (PHP 4 >= 4.0.7RC1)

Update specified tuple in relation

```
int dbplus_update (resource relation, array old, array new)
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

dbplus_update() replaces the tuple given by `old` with the data from `new` if and only if `old` completely matches a tuple within `relation`.

dbplus_xlockrel (PHP 4 >= 4.0.7RC1)

Request exclusive lock on relation

```
int dbplus_xlockrel (resource relation)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

dbplus_xlockrel() will request an exclusive lock on *relation* preventing even read access from other clients.

See also dbplus_xunlockrel().

dbplus_xunlockrel (PHP 4 >= 4.0.7RC1)

Free exclusive lock on relation

```
int dbplus_xunlockrel (resource relation)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

dbplus_xunlockrel() will release an exclusive lock on *relation* previously obtained by dbplus_xlockrel().

XXI. Directory functions

chroot (PHP 4 >= 4.0.5)

change the root directory

```
bool chroot (string directory)
```

Changes the root directory of the current process to *directory*. Returns FALSE if unable to change the root directory, TRUE otherwise.

> **Nota:** It's not wise to use this function when running in a webserver environment, because it's not possible to reset the root directory to / again at the end of the request. This function will only function correct when running as CGI this way.

chdir (PHP 3, PHP 4 >= 4.0.0)

change directory

```
bool chdir (string directory)
```

Changes PHP's current directory to *directory*. Returns FALSE if unable to change directory, TRUE otherwise.

dir (PHP 3, PHP 4 >= 4.0.0)

directory class

```
object dir (string directory)
```

A pseudo-object oriented mechanism for reading a directory. The given *directory* is opened. Two properties are available once the directory has been opened. The handle property can be used with other directory functions such as readdir(), rewinddir() and closedir(). The path property is set to path the directory that was opened. Three methods are available: read, rewind and close.

Esempio 1. dir() example

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

```
    echo $entry."<br>\n";
}
$d->close();
```

closedir (PHP 3, PHP 4 >= 4.0.0)

close directory handle

```
void closedir (resource dir_handle)
```

Closes the directory stream indicated by *dir_handle*. The stream must have previously been opened by opendir().

getcwd (PHP 4 >= 4.0.0)

gets the current working directory

```
string getcwd ()
```

Returns the current working directory.

opendir (PHP 3, PHP 4 >= 4.0.0)

open directory handle

```
resource opendir (string path)
```

Returns a directory handle to be used in subsequent closedir(), readdir(), and rewinddir() calls.

If *path* is not a valid directory or the directory can not be opened due to permission restrictions or filesystem errors, **opendir()** returns FALSE and generates a PHP error. You can suppress the error output of **opendir()** by prepending '@' to the front of the function name.

Esempio 1. opendir() example

```
<?php
```

```
if ($dir = @opendir("/tmp")) {
  while ($file = readdir($dir)) {
    echo "$file\n";
  }
  closedir($dir);
}

?>
```

readdir (PHP 3, PHP 4 >= 4.0.0)

read entry from directory handle

```
string readdir (resource dir_handle)
```

Returns the filename of the next file from the directory. The filenames are not returned in any particular order.

Esempio 1. List all files in the current directory

```
// Note that !== did not exist until 4.0.0-RC2
<?php
$handle=opendir('.');
echo "Directory handle: $handle\n";
echo "Files:\n";
while (false !== ($file = readdir($handle))) {
    echo "$file\n";
}
closedir($handle);
?>
```

Note that **readdir()** will return the `.` and `..` entries. If you don't want these, simply strip them out:

Esempio 2. List all files in the current directory and strip out `.` and `..`

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

```
}
closedir($handle);
?>
```

rewinddir (PHP 3, PHP 4 >= 4.0.0)

rewind directory handle

```
void rewinddir (resource dir_handle)
```

Resets the directory stream indicated by *dir_handle* to the beginning of the directory.

XXII. DOM XML functions

Attenzione

Questo modulo è *SPERIMENTALE*. Ovvero, il comportamento di queste funzioni, i nomi di queste funzioni, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

This documentation is not finished yet. Don't start to translate it or use it as a programming reference (steinm@php.net).

These functions are only available if PHP was configured with `--with-dom=[DIR]`, using the GNOME xml library (http://www.xmlsoft.org). You will need at least libxml-2.2.7 These functions have been added in PHP 4.

The extension allows you to operate on an XML document with the DOM API. It also provides a function xmltree() to turn the complete XML document into a tree of PHP objects. Currently this tree should be considered read-only - you can modify it but this would not make any sense since **dumpmem()** cannot be applied to it. Therefore, if you want to read an XML file and write a modified version use the **add_node()**, **set_attribute()**, etc. and finally **dumpmem()** functions.

This module defines the following constants:

Tabella 1. XML constants

Constant	Value	Description

Each function in this extension can be used in two ways. In a non-object oriented way by passing the object to apply the function to as a first argument, or in an object oriented way by calling the function as a method of an object. This documentation describes the non-object oriented functions, though you get the object methods by skipping the prefix "domxml_".

This module defines a number of classes, which are listed — including their properties and method — in the following table.

Tabella 2. DomDocument class (methods)

Method name	Function name	Description

Tabella 3. DomDocument class (attributes)

Name	Type	Description

Tabella 4. DomNode class (methods)

Name	PHP name	Description

Tabella 5. DomNode class (attributes)

Name	Type	Description

xmldoc (PHP 4 >= 4.0.0)

Creates a DOM object of an XML document

```
object xmldoc (string str)
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

The function parses the XML document in *str* and returns an object of class "Dom document", having the properties as listed above.

See also xmldocfile()

xmldocfile (PHP 4 >= 4.0.0)

Creates a DOM object from XML file

```
object xmldocfile (string filename)
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

The function parses the XML document in the file named *filename* and returns an object of class "Dom document", having the properties as listed above. The file is accessed read-only.

See also xmldoc()

xmltree (PHP 4 >= 4.0.0)

Creates a tree of PHP objects from XML document

```
object xmltree (string str)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

The function parses the XML document in *str* and returns a tree PHP objects as the parsed document. This function is isolated from the other functions, which means you cannot access the tree with any of the other functions. Modifying it, for example by adding nodes, makes no sense since there is currently no way to dump it as an XML file. However this function may be valuable if you want to read a file and investigate the content.

domxml_root (PHP 4 >= 4.0.0)

Returns root element node

```
object domxml_root (object doc)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

domxml_root() takes one argument, an object of class "Dom document", and returns the root element node. There are actually other possible nodes like comments which are currently disregarded.

The following example returns just the element with name CHAPTER and prints it. The other root node -- the comment -- is not returned.

Esempio 1. Retrieving root element

```
<?php
$xmlstr = "<?xml version='1.0' standalone='yes'?>
<!DOCTYPE chapter SYSTEM '/share/sgml/Norman_Walsh/db3xml10/db3xml10.dtd'
[ <!ENTITY sp \"spanish\">
]>
<!-- lsfj  -->
```

```
<chapter language='en'><title language='en'>Title</title>
 <para language='ge'>
  &sp;
  <!-- comment -->
  <informaltable language='&sp;'>
   <tgroup cols='3'>
    <tbody>
     <row><entry>a1</entry><entry
morerows='1'>b1</entry><entry>c1</entry></row>
<row><entry>a2</entry><entry>c2</entry></row>
     <row><entry>a3</entry><entry>b3</entry><entry>c3</entry></row>
    </tbody>
   </tgroup>
  </informaltable>
 </para>
</chapter>";

if(!$dom = xmldoc($xmlstr)) {
  echo "Error while parsing the document\n";
  exit;
}

$root = $dom->root();
/* or $root = domxml_root($dom); */
print_r($root);
?>
```

domxml_add_root (PHP 4 >= 4.0.0)

Adds a further root node

```
resource domxml_add_root (resource doc, string name)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

Adds a root element node to a dom document and returns the new node. The element name is given in the second parameter.

Esempio 1. Creating a simple HTML document header

```
<?php
$doc = new_xmldoc("1.0");
$root = $doc->add_root("HTML");
$head = $root->new_child("HEAD", "");
$head->new_child("TITLE", "Hier der Titel");
echo $doc->dumpmem();
?>
```

domxml_dumpmem (PHP 4 >= 4.0.0)

Dumps the internal XML tree back into a string

```
string domxml_dumpmem (resource doc)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

Creates an XML document from the dom representation. This function usually is called after a building a new dom document from scratch as in the example of domxml_add_root().

See also domxml_add_root()

domxml_attributes (PHP 4 >= 4.0.0)

Returns an array of attributes of a node

```
array domxml_attributes (resource node)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

Returns all attributes of a node as array of objects of type "dom attribute".

domxml_get_attribute (PHP 4 >= 4.0.5)

Returns a certain attribute of a node

```
object domxml_get_attribute (resource node, string name)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

Returns the attribute with name *name* of the given node.

See also domxml_set_attribute()

domxml_set_attribute (PHP 4 >= 4.0.5)

```
object domxml_set_attribute (resource node, string name, string value)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

Sets an attribute with name *name* of the given node on a value.

If we take the example from domxml_add_root() it is simple to add an attribute to the HEAD element by the simply calling the **set_attribute()** function of the node class.

Esempio 1. Adding an attribute to an element

```
<?php
$doc = new_xmldoc("1.0");
$root = $doc->add_root("HTML");
$head = $root->new_child("HEAD", "");
$head->new_child("TITLE", "Hier der Titel");
$head->set_attribute("Language", "ge");
echo $doc->dumpmem();
?>
```

domxml_children (PHP 4 >= 4.0.0)

Returns children of a node or document

```
array domxml_children (object doc|node)
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

Returns all children of a node as an array of nodes.

In the following example the variable `children` will contain an array with one node of type XML_ELEMENT. This node is the TITLE element.

Esempio 1. Adding an attribute to an element

```
<?php
$doc = new_xmldoc("1.0");
$root = $doc->add_root("HTML");
$head = $root->new_child("HEAD", "");
$head->new_child("TITLE", "Hier der Titel");
$head->set_attribute("Language", "ge");
$children = $head->children();
?>
```

domxml_new_child (PHP 4 >= 4.0.0)

Adds new child node

```
resource domxml_new_child (string name, string content)
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

Adds a new child of type element to a node and returns it.

domxml_new_xmldoc (PHP 4 >= 4.0.0)

Creates new empty XML document

```
object domxml_new_xmldoc (string version)
```

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

Creates a new dom document from scratch and returns it.

See also domxml_add_root()

xpath_new_context (PHP 4 >= 4.0.4)

Creates new xpath context

```
object xpath_new_context (object dom document)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

See also **()**

xpath_eval (PHP 4 >= 4.0.4)

Evaluates an xpath expression

```
array xpath_eval (object xpath context)
```

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva TUTTO ciò che è documentato qui può cambiare nei futuri rilasci del PHP SENZA PREAVVISO. Siete avvisati, l'uso di questo modulo è a vostro rischio.

See also **()**

XXIII. Error Handling and Logging Functions

These are functions dealing with error handling and logging. They allow you to define your own error handling rules, as well as modify the way the errors can be logged. This allows you to change and enhance error reporting to suit your needs.

With the logging functions, you can send messages directly to other machines, to an email (or email to pager gateway!), to system logs, etc., so you can selectively log and monitor the most important parts of your applications and websites.

The error reporting functions allow you to customize what level and kind of error feedback is given, ranging from simple notices to customized functions returned during errors.

error_log (PHP 3, PHP 4 >= 4.0.0)

send an error message somewhere

```
int error_log (string message, int message_type [, string destination [,
string extra_headers]])
```

Sends an error message to the web server's error log, a TCP port or to a file. The first parameter, *message*, is the error message that should be logged. The second parameter, *message_type* says where the message should go:

Tabella 1. error_log() log types

> **Attenzione**
>
> Remote debugging via TCP/IP is a PHP 3 feature that is *not* available in PHP 4.

Esempio 1. error_log() examples

```
// Send notification through the server log if we can not
// connect to the database.
if (!Ora_Logon ($username, $password)) {
    error_log ("Oracle database not available!", 0);
}

// Notify administrator by email if we run out of FOO
if (!($foo = allocate_new_foo()) {
    error_log ("Big trouble, we're all out of FOOs!", 1,
               "operator@mydomain.com");
}

// other ways of calling error_log():
error_log ("You messed up!", 2, "127.0.0.1:7000");
error_log ("You messed up!", 2, "loghost");
error_log ("You messed up!", 3, "/var/tmp/my-errors.log");
```

error_reporting (PHP 3, PHP 4 >= 4.0.0)

set which PHP errors are reported

```
int error_reporting ([int level])
```

Sets PHP's error reporting level and returns the old level. The error reporting level is either a bitmask, or named constant. Using named constants is strongly encouraged to ensure compatibility for future versions. As error levels are added, the range of integers increases, so older integer-based error levels will not always behave as expected.

Esempio 1. Error Integer changes

```
error_reporting (55);   // PHP 3 equivalent to E_ALL ^ E_NOTICE

/* ...in PHP 4, '55' would mean (E_ERROR | E_WARNING | E_PARSE |
E_CORE_ERROR | E_CORE_WARNING) */

error_reporting (2039); // PHP 4 equivalent to E_ALL ^ E_NOTICE

error_reporting (E_ALL ^ E_NOTICE); // The same in both PHP 3 and 4
```

Follow the links for the internal values to get their meanings:

Tabella 1. error_reporting() bit values

constant	value

Esempio 2. error_reporting() examples

```
error_reporting(0);
/* Turn off all reporting */

error_reporting (7); // Old syntax, PHP 2/3
error_reporting  (E_ERROR | E_WARNING | E_PARSE); // New syn-
tax for PHP 3/4
/* Good to use for simple running errors  */

error_reporting  (15); // Old syntax, PHP 2/3
error_reporting (E_ERROR | E_WARNING | E_PARSE | E_NOTICE); // New syn-
tax for PHP 3/4
/*  good for code authoring to report uninitialized or (possibly mis-
spelled) variables */

error_reporting (63); // Old syntax, PHP 2/3
error_reporting (E_ALL); // New syntax for PHP 3/4
/* report all PHP errors */
```

restore_error_handler (PHP 4)

Restores the previous error handler function

```
void restore_error_handler ()
```

Used after changing the error handler function using set_error_handler(), to revert to the previous error handler (which could be the built-in or a user defined function)

See also error_reporting(), set_error_handler(), trigger_error(), user_error()

set_error_handler (PHP 4)

Sets a user-defined error handler function.

```
string set_error_handler (string error_handler)
```

Sets a user function (*error_handler*) to handle errors in a script. Returns the previously defined error handler (if any), or FALSE on error. This function can be used for defining your own way of handling errors during runtime, for example in applications in which you need to do cleanup of data/files when a critical error happens, or when you need to trigger an error under certain conditions (using trigger_error())

The user function needs to accept 2 parameters: the error code, and a string describing the error. From PHP 4.0.2, an additional 3 optional parameters are supplied: the filename in which the error occured, the line number in which the error occured, and the context in which the error occured (an array that points to the active symbol table at the point the error occurred).

The example below shows the handling of internal execptions by triggering errors and handling them with a user defined function:

Esempio 1. Error handling with set_error_handler() and trigger_error()

```
<?php

// redefine the user error constants - PHP 4 only
define (FATAL,E_USER_ERROR);
define (ERROR,E_USER_WARNING);
define (WARNING,E_USER_NOTICE);
```

```
// set the error reporting level for this script
error_reporting (FATAL | ERROR | WARNING);

// error handler function
function myErrorHandler ($errno, $errstr, $errfile, $errline) {
  switch ($errno) {
  case FATAL:
    echo "<b>FATAL</b> [$errno] $errstr<br>\n";
    echo "  Fatal error in line ".$errline." of file ".$errfile;
    echo ", PHP ".PHP_VERSION." (".PHP_OS.")<br>\n";
    echo "Aborting...<br>\n";
    exit -1;
    break;
  case ERROR:
    echo "<b>ERROR</b> [$errno] $errstr<br>\n";
    break;
  case WARNING:
    echo "<b>WARNING</b> [$errno] $errstr<br>\n";
    break;
    default:
    echo "Unkown error type: [$errno] $errstr<br>\n";
    break;
  }
}

// function to test the error handling
function scale_by_log ($vect, $scale) {
  if ( !is_numeric($scale) || $scale <= 0 )
    trigger_error("log(x) for x <= 0 is unde-
fined, you used: scale = $scale",
      FATAL);
  if (!is_array($vect)) {
    trigger_error("Incorrect input vector, array of values ex-
pected", ERROR);
    return null;
  }
  for ($i=0; $i<count($vect); $i++) {
    if (!is_numeric($vect[$i]))
      trigger_error("Value at position $i is not a number, us-
ing 0 (zero)",
        WARNING);
    $temp[$i] = log($scale) * $vect[$i];
  }
  return $temp;
}

// set to the user defined error handler
$old_error_handler = set_error_handler("myErrorHandler");
```

```
// trigger some errors, first define a mixed array with a non-numeric item
echo "vector a\n";
$a = array(2,3,"foo",5.5,43.3,21.11);
print_r($a);

// now generate second array, generating a warning
echo "----\nvector b - a warning (b = log(PI) * a)\n";
$b = scale_by_log($a, M_PI);
print_r($b);

// this is trouble, we pass a string instead of an array
echo "----\nvector c - an error\n";
$c = scale_by_log("not array",2.3);
var_dump($c);

// this is a critical error, log of zero or negative number is undefined
echo "----\nvector d - fatal error\n";
$d = scale_by_log($a, -2.5);

?>
```

And when you run this sample script, the output will be

```
vector a
Array
(
    [0] => 2
    [1] => 3
    [2] => foo
    [3] => 5.5
    [4] => 43.3
    [5] => 21.11
)
----
vector b - a warning (b = log(PI) * a)
<b>WARNING</b> [1024] Value at position 2 is not a number, us-
ing 0 (zero)<br>
Array
(
    [0] => 2.2894597716988
    [1] => 3.4341896575482
    [2] => 0
    [3] => 6.2960143721717
    [4] => 49.566804057279
    [5] => 24.165247890281
)
----
```

```
vector c - an error
<b>ERROR</b> [512] Incorrect input vector, array of values expected<br>
NULL
----
vector d - fatal error
<b>FATAL</b> [256] log(x) for x <= 0 is undefined, you used: scale = -
2.5<br>
  Fatal error in line 36 of file trigger_error.php, PHP 4.0.2 (Linux)<br>
Aborting...<br>
```

It is important to remember that the standard PHP error handler is completely bypassed. error_reporting() settings will have no effect and your error handler will be called regardless - however you are still able to read the current value of error_reporting() and act appropriately. Of particular note is that this value will be 0 if the statement that caused the error was prepended by the @ error-control operator.

Also note that it is your responsibility to die() if necessary. If the error-handler function returns, script execution will continue with the next statement after the one that caused an error.

See also error_reporting(), restore_error_handler(), trigger_error(), user_error()

trigger_error (PHP 4)

Generates a user-level error/warning/notice message

```
void trigger_error (string error_msg [, int error_type])
```

Used to trigger a user error condition, it can be used by in conjunction with the built-in error handler, or with a user defined function that has been set as the new error handler (set_error_handler()). It only works with the E_USER family of constants, and will default to `E_USER_NOTICE`.

This function is useful when you need to generate a particular response to an exception at runtime. For example:

```
if (assert ($divisor == 0))
   trigger_error ("Cannot divide by zero", E_USER_ERROR);
```

Nota: See set_error_handler() for a more extensive example.

See also error_reporting(), set_error_handler(), restore_error_handler(), user_error()

user_error (PHP 4 >= 4.0.0)

Generates a user-level error/warning/notice message

```
void user_error (string error_msg [, int error_type])
```

This is an alias for the function trigger_error().

See also error_reporting(), set_error_handler(), restore_error_handler(), and trigger_error()

XXIV. FrontBase functions

These functions allow you to access FrontBase database servers. In order to have these functions available, you must compile php with fbsql support by using the `--with-fbsql` option. If you use this option without specifying the path to fbsql, php will search for the fbsql client libraries in the default installation location for the platform. Users who installed FrontBase in a non standard directory should always specify the path to fbsql: `--with-fbsql=/path/to/fbsql`. This will force php to use the client libraries installed by FrontBase, avoiding any conflicts.

More information about FrontBase can be found at http://www.frontbase.com/.

Documentation for FrontBase can be found at http://www.frontbase.com/cgi-bin/WebObjects/FrontBase.woa/wa/productsPage?currentPage=Documentation.

Frontbase support has been added to PHP 4.0.6.

fbsql_affected_rows (PHP 4 >= 4.0.6)

Get number of affected rows in previous FrontBase operation

```
int fbsql_affected_rows ([int link_identifier])
```

fbsql_affected_rows() returns the number of rows affected by the last INSERT, UPDATE or DELETE query associated with *link_identifier*. If the link identifier isn't specified, the last link opened by fbsql_connect() is assumed.

> **Nota:** If you are using transactions, you need to call **fbsql_affected_rows()** after your INSERT, UPDATE, or DELETE query, not after the commit.

If the last query was a DELETE query with no WHERE clause, all of the records will have been deleted from the table but this function will return zero.

> **Nota:** When using UPDATE, FrontBase will not update columns where the new value is the same as the old value. This creates the possiblity that **fbsql_affected_rows()** may not actually equal the number of rows matched, only the number of rows that were literally affected by the query.

If the last query failed, this function will return -1.

See also: fbsql_num_rows().

fbsql_autocommit (PHP 4 >= 4.0.6)

Enable or disable autocommit.

```
bool fbsql_autocommit (resource link_identifier [, bool OnOff])
```

fbsql_autocommit() returns the current autocommit status. if the optional OnOff parameter is given the auto commit status will be changed. With OnOff set to TRUE each statement will be commited automatically, if no errors was found. With OnOff set to FALSE the user must commit or rollback the transaction unsing either fbsql_commit() or fbsql_rollback().

See also: fbsql_commit() and fbsql_rollback()

fbsql_change_user (unknown)

Change logged in user of the active connection

```
resource fbsql_change_user (string user, string password [, string
database [, int link_identifier]])
```

fbsql_change_user() changes the logged in user of the current active connection, or the connection given by the optional parameter link_identifier. If a database is specified, this will default or current database after the user has been changed. If the new user and password authorization fails, the current connected user stays active.

fbsql_close (PHP 4 >= 4.0.6)

Close FrontBase connection

```
boolean fbsql_close ([resource link_identifier])
```

Returns: TRUE on success, FALSE on error.

fbsql_close() closes the connection to the FrontBase server that's associated with the specified link identifier. If *link_identifier* isn't specified, the last opened link is used.

Using **fbsql_close()** isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution.

Esempio 1. fbsql_close() example

```
<?php
    $link = fbsql_connect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    print ("Connected successfully");
    fbsql_close ($link);
?>
```

See also: fbsql_connect(), and fbsql_pconnect().

fbsql_commit (PHP 4 >= 4.0.6)

Commits a transaction to the database

```
bool fbsql_commit ([resource link_identifier])
```

Returns: TRUE on success, FALSE on failure.

fbsql_commit() ends the current transaction by writing all insertsm updates and deletes to the disk and unlucking all row and table locks held by the transaction. This command is only needed if autocommit is set to false.

See also: fbsql_autocommit() and fbsql_rollback()

fbsql_connect (PHP 4 >= 4.0.6)

Open a connection to a FrontBase Server

```
resource fbsql_connect ([string hostname [, string username [, string
password]]])
```

Returns a positive FrontBase link identifier on success, or an error message on failure.

fbsql_connect() establishes a connection to a FrontBase server. The following defaults are assumed for missing optional parameters: *hostname* = 'NULL', *username* = '_SYSTEM' and *password* = empty password.

If a second call is made to **fbsql_connect()** with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned.

The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling fbsql_close().

Esempio 1. fbsql_connect() example

```
<?php

    $link = fbsql_connect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    print ("Connected successfully");
    fbsql_close ($link);

?>
```

See also fbsql_pconnect(), and fbsql_close().

fbsql_create_db (PHP 4 >= 4.0.6)

Create a FrontBase database

```
bool fbsql_create_db (string database name [, resource link_identifier])
```

fbsql_create_db() attempts to create a new database on the server associated with the specified link identifier.

Esempio 1. fbsql_create_db() example

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    if (fbsql_create_db ("my_db")) {
        print("Database created successfully\n");
    } else {
        printf("Error creating database: %s\n", fbsql_error ());
    }
?>
```

See also: fbsql_drop_db().

fbsql_database_password (PHP 4 >= 4.0.6)

Sets or retreives the password for a FrontBase database

```
string fbsql_database_password (resource link_identifier [, string
database_password])
```

Returns: The database password for the database represented by the link identifier.

fbsql_database_password() sets and retreives the database password for the current database. if the second optional parameter is given the function sets the database password for the database on the server that's associated with the specified link identifier. If no link identifier is specified, the last opened link is assumed. If no link is open, the function will try to establish a link as if fbsql_connect() was called, and use it.

See also: fbsql_connect() and fbsql_pconnect()

fbsql_data_seek (PHP 4 >= 4.0.6)

Move internal result pointer

```
bool fbsql_data_seek (resource result_identifier, int row_number)
```

Returns: `TRUE` on success, `FALSE` on failure.

fbsql_data_seek() moves the internal row pointer of the FrontBase result associated with the specified result identifier to point to the specified row number. The next call to fbsql_fetch_row() would return that row.

`Row_number` starts at 0.

Esempio 1. fbsql_data_seek() example

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");

    fbsql_select_db ("samp_db")
        or die ("Could not select database");

    $query = "SELECT last_name, first_name FROM friends;";
    $result = fbsql_query ($query)
        or die ("Query failed");

    # fetch rows in reverse order

    for ($i = fbsql_num_rows ($result) - 1; $i >=0; $i--) {
        if (!fbsql_data_seek ($result, $i)) {
            printf ("Cannot seek to row %d\n", $i);
            continue;
        }

        if(!($row = fbsql_fetch_object ($result)))
            continue;

        printf("%s %s<BR>\n", $row->last_name, $row->first_name);
    }

    fbsql_free_result ($result);
?>
```

fbsql_db_query (PHP 4 >= 4.0.6)

Send a FrontBase query

```
resource fbsql_db_query (string database, string query [, resource
link_identifier])
```

Returns: A positive FrontBase result identifier to the query result, or FALSE on error.

fbsql_db_query() selects a database and executes a query on it. If the optional link identifier isn't specified, the function will try to find an open link to the FrontBase server and if no such link is found it'll try to create one as if fbsql_connect() was called with no arguments

See also fbsql_connect().

fbsql_db_status (PHP 4 >= 4.0.7RC1)

Get the status for a given database.

```
int fbsql_db_status (string database_name [, resource link_identifier])
```

Returns: An integer value with the current status.

fbsql_db_status() requests the current status of the database specified by *database_name*. if the *link_identifier* is omitted the default link_identifier will be used.

The return value can be one of the following constants:

- FALSE - The exec handler for the host was invalid. This error will ocour when the link_identifier connects directly to a database by using a port number. FBExec can be available on the server but no connection has been made for it.
- FBSQL_UNKNOWN - The Status is unknown.
- FBSQL_STOPPED - The database is not running. Use fbsql_start_db() to start the database.
- FBSQL_STARTING - The database is starting.
- FBSQL_RUNNING - The database is running and can be used to perform SQL operations.
- FBSQL_STOPPING - The database is stopping.
- FBSQL_NOEXEC - FBExec is not running on the server and it is not possible to get the status of the database.

See also: fbsql_start_db() and fbsql_stop_db()

fbsql_drop_db (PHP 4 >= 4.0.6)

Drop (delete) a FrontBase database

```
bool fbsql_drop_db (string database_name [, resource link_identifier])
```

Returns: `TRUE` on success, `FALSE` on failure.

fbsql_drop_db() attempts to drop (remove) an entire database from the server associated with the specified link identifier.

fbsql_errno (PHP 4 >= 4.0.6)

Returns the numerical value of the error message from previous FrontBase operation

```
int fbsql_errno ([resource link_identifier])
```

Returns the error number from the last fbsql function, or `0` (zero) if no error occurred.

Errors coming back from the fbsql database backend dont issue warnings. Instead, use **fbsql_errno()** to retrieve the error code. Note that this function only returns the error code from the most recently executed fbsql function (not including fbsql_error() and **fbsql_errno()**), so if you want to use it, make sure you check the value before calling another fbsql function.

```
<?php
fbsql_connect("marliesle");
echo fbsql_errno().": ".fbsql_error()."<BR>";
fbsql_select_db("nonexistentdb");
echo fbsql_errno().": ".fbsql_error()."<BR>";
$conn = fbsql_query("SELECT * FROM nonexistenttable;");
echo fbsql_errno().": ".fbsql_error()."<BR>";
?>
```

See also: fbsql_error(), fbsql_warnings()

fbsql_error (PHP 4 >= 4.0.6)

Returns the text of the error message from previous FrontBase operation

```
string fbsql_error ([resource link_identifier])
```

Returns the error text from the last fbsql function, or `''` (the empty string) if no error occurred.

Errors coming back from the fbsql database backend dont issue warnings. Instead, use **fbsql_error()** to retrieve the error text. Note that this function only returns the error text from the most recently executed fbsql function (not including **fbsql_error()** and fbsql_errno()), so if you want to use it, make sure you check the value before calling another fbsql function.

```
<?php
fbsql_connect("marliesle");
echo fbsql_errno().": ".fbsql_error()."<BR>";
fbsql_select_db("nonexistentdb");
echo fbsql_errno().": ".fbsql_error()."<BR>";
$conn = fbsql_query("SELECT * FROM nonexistenttable;");
echo fbsql_errno().": ".fbsql_error()."<BR>";
?>
```

See also: fbsql_errno(), fbsql_warnings()

fbsql_fetch_array (PHP 4 >= 4.0.6)

Fetch a result row as an associative array, a numeric array, or both.

```
array fbsql_fetch_array (resource result [, int result_type])
```

Returns an array that corresponds to the fetched row, or FALSE if there are no more rows.

fbsql_fetch_array() is an extended version of fbsql_fetch_row(). In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

If two or more columns of the result have the same field names, the last column will take precedence. To access the other column(s) of the same name, you must the numeric index of the column or make an alias for the column.

```
select t1.f1 as foo t2.f1 as bar from t1, t2
```

An important thing to note is that using **fbsql_fetch_array()** is NOT significantly slower than using fbsql_fetch_row(), while it provides a significant added value.

The optional second argument *result_type* in **fbsql_fetch_array()** is a constant and can take the following values: FBSQL_ASSOC, FBSQL_NUM, and FBSQL_BOTH.

For further details, see also fbsql_fetch_row() and fbsql_fetch_assoc().

Esempio 1. fbsql_fetch_array() example

```
<?php
fbsql_connect ($host, $user, $password);
$result = fbsql_db_query ("database","select user_id, full-
name from table");
```

```
while ($row = fbsql_fetch_array ($result)) {
    echo "user_id: ".$row["user_id"]."<br>\n";
    echo "user_id: ".$row[0]."<br>\n";
    echo "fullname: ".$row["fullname"]."<br>\n";
    echo "fullname: ".$row[1]."<br>\n";
}
fbsql_free_result ($result);
?>
```

fbsql_fetch_assoc (PHP 4 >= 4.0.6)

Fetch a result row as an associative array

```
array fbsql_fetch_assoc (resource result)
```

Returns an associative array that corresponds to the fetched row, or FALSE if there are no more rows.

fbsql_fetch_assoc() is equivalent to calling fbsql_fetch_array() with FBSQL_ASSOC for the optional second parameter. It only returns an associative array. This is the way fbsql_fetch_array() originally worked. If you need the numeric indices as well as the associative, use fbsql_fetch_array().

If two or more columns of the result have the same field names, the last column will take precedence. To access the other column(s) of the same name, you must use fbsql_fetch_array() and have it return the numeric indices as well.

An important thing to note is that using **fbsql_fetch_assoc()** is NOT significantly slower than using fbsql_fetch_row(), while it provides a significant added value.

For further details, see also fbsql_fetch_row() and fbsql_fetch_array().

Esempio 1. fbsql_fetch_assoc() example

```
<?php
fbsql_connect ($host, $user, $password);
$result = fbsql_db_query ("database","select * from table");
while ($row = fbsql_fetch_assoc ($result)) {
    echo $row["user_id"];
    echo $row["fullname"];
}
fbsql_free_result ($result);
?>
```

fbsql_fetch_field (PHP 4 >= 4.0.6)

Get column information from a result and return as an object

```
object fbsql_fetch_field (resource result [, int field_offset])
```

Returns an object containing field information.

fbsql_fetch_field() can be used in order to obtain information about fields in a certain query result. If the field offset isn't specified, the next field that wasn't yet retrieved by **fbsql_fetch_field()** is retrieved.

The properties of the object are:

- name - column name
- table - name of the table the column belongs to
- max_length - maximum length of the column
- not_null - 1 if the column cannot be `NULL`
- type - the type of the column

Esempio 1. fbsql_fetch_field() example

```
<?php
fbsql_connect ($host, $user, $password)
    or die ("Could not connect");
$result = fbsql_db_query ("database", "select * from table")
    or die ("Query failed");
# get column metadata
$i = 0;
while ($i < fbsql_num_fields ($result)) {
    echo "Information for column $i:<BR>\n";
    $meta = fbsql_fetch_field ($result);
    if (!$meta) {
        echo "No information available<BR>\n";
    }
    echo "<PRE>
max_length:   $meta->max_length
name:         $meta->name
not_null:     $meta->not_null
table:        $meta->table
type:         $meta->type
</PRE>";
    $i++;
}
```

```
fbsql_free_result ($result);
?>
```

See also fbsql_field_seek().

fbsql_fetch_lengths (PHP 4 >= 4.0.6)

Get the length of each output in a result

```
array fbsql_fetch_lengths ([resource result])
```

Returns: An array that corresponds to the lengths of each field in the last row fetched by fbsql_fetch_row(), or FALSE on error.

fbsql_fetch_lengths() stores the lengths of each result column in the last row returned by fbsql_fetch_row(), fbsql_fetch_array(), and fbsql_fetch_object() in an array, starting at offset 0.

See also: fbsql_fetch_row().

fbsql_fetch_object (PHP 4 >= 4.0.6)

Fetch a result row as an object

```
object fbsql_fetch_object (resource result [, int result_type])
```

Returns an object with properties that correspond to the fetched row, or FALSE if there are no more rows.

fbsql_fetch_object() is similar to fbsql_fetch_array(), with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

The optional argument *result_type* is a constant and can take the following values: FBSQL_ASSOC, FBSQL_NUM, and FBSQL_BOTH.

Speed-wise, the function is identical to fbsql_fetch_array(), and almost as quick as fbsql_fetch_row() (the difference is insignificant).

Esempio 1. fbsql_fetch_object() example

```
<?php
fbsql_connect ($host, $user, $password);
```

```
$result = fbsql_db_query ("database", "select * from table");
while ($row = fbsql_fetch_object ($result)) {
    echo $row->user_id;
    echo $row->fullname;
}
fbsql_free_result ($result);
?>
```

See also: fbsql_fetch_array() and fbsql_fetch_row().

fbsql_fetch_row (PHP 4 >= 4.0.6)

Get a result row as an enumerated array

```
array fbsql_fetch_row (resource result)
```

Returns: An array that corresponds to the fetched row, or FALSE if there are no more rows.

fbsql_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

Subsequent call to **fbsql_fetch_row()** would return the next row in the result set, or FALSE if there are no more rows.

See also: fbsql_fetch_array(), fbsql_fetch_object(), fbsql_data_seek(), fbsql_fetch_lengths(), and fbsql_result().

fbsql_field_flags (PHP 4 >= 4.0.6)

Get the flags associated with the specified field in a result

```
string fbsql_field_flags (resource result, int field_offset)
```

fbsql_field_flags() returns the field flags of the specified field. The flags are reported as a single word per flag separated by a single space, so that you can split the returned value using explode().

fbsql_field_name (PHP 4 >= 4.0.6)

Get the name of the specified field in a result

```
string fbsql_field_name (resource result, int field_index)
```

fbsql_field_name() returns the name of the specified field index. *result* must be a valid result identifier and *field_index* is the numerical offset of the field.

> **Nota:** *field_index* starts at 0.
>
> e.g. The index of the third field would actually be 2, the index of the fourth field would be 3 and so on.

Esempio 1. fbsql_field_name() example

```
// The users table consists of three fields:
//   user_id
//   username
//   password.

$res = fbsql_db_query("users", "select * from users", $link);

echo fbsql_field_name($res, 0) . "\n";
echo fbsql_field_name($res, 2);
```

The above example would produce the following output:

```
user_id
password
```

fbsql_field_len (PHP 4 >= 4.0.6)

Returns the length of the specified field

```
int fbsql_field_len (resource result, int field_offset)
```

fbsql_field_len() returns the length of the specified field.

fbsql_field_seek (PHP 4 >= 4.0.6)

Set result pointer to a specified field offset

```
bool fbsql_field_seek (resource result, int field_offset)
```

Seeks to the specified field offset. If the next call to fbsql_fetch_field() doesn't include a field offset, the field offset specified in **fbsql_field_seek()** will be returned.

See also: fbsql_fetch_field().

fbsql_field_table (PHP 4 >= 4.0.6)

Get name of the table the specified field is in

```
string fbsql_field_table (resource result, int field_offset)
```

Returns the name of the table that the specifed field is in.

fbsql_field_type (PHP 4 >= 4.0.6)

Get the type of the specified field in a result

```
string fbsql_field_type (resource result, int field_offset)
```

fbsql_field_type() is similar to the fbsql_field_name() function. The arguments are identical, but the field type is returned instead. The field type will be one of "int", "real", "string", "blob", and others as detailed in the FrontBase documentation (http://www.frontbase.com/cgi-bin/WebObjects/FrontBase.woa/wa/productsPage?currentPage=Documentation).

Esempio 1. fbsql_field_type() example

```
<?php

fbsql_connect ("localhost", "_SYSTEM", "");
fbsql_select_db ("wisconsin");
$result = fbsql_query ("SELECT * FROM onek;");
$fields = fbsql_num_fields ($result);
$rows   = fbsql_num_rows ($result);
```

```
$i = 0;
$table = fbsql_field_table ($result, $i);
echo "Your '".$table."' ta-
ble has ".$fields." fields and ".$rows." records <BR>";
echo "The table has the following fields <BR>";
while ($i < $fields) {
    $type  = fbsql_field_type  ($result, $i);
    $name  = fbsql_field_name  ($result, $i);
    $len   = fbsql_field_len   ($result, $i);
    $flags = fbsql_field_flags ($result, $i);
    echo $type." ".$name." ".$len." ".$flags."<BR>";
    $i++;
}
fbsql_close();

?>
```

fbsql_free_result (PHP 4 >= 4.0.6)

Free result memory

```
bool fbsql_free_result (int result)
```

fbsql_free_result() will free all memory associated with the result identifier *result*.

fbsql_free_result() only needs to be called if you are concerned about how much memory is being used for queries that return large result sets. All associated result memory is automatically freed at the end of the script's execution.

fbsql_insert_id (PHP 4 >= 4.0.6)

Get the id generated from the previous INSERT operation

```
int fbsql_insert_id ([resource link_identifier])
```

fbsql_insert_id() returns the ID generated for an column defined as DEFAULT UNIQUE by the previous INSERT query using the given *link_identifier*. If *link_identifier* isn't specified, the last opened link is assumed.

fbsql_insert_id() returns 0 if the previous query does not generate an DEFAULT UNIQUE value. If you need to save the value for later, be sure to call fbsql_insert_id() immediately after the query that generates the value.

> **Nota:** The value of the FrontBase SQL function `LAST_INSERT_ID()` always contains the most recently generated DEFAULT UNIQUE value, and is not reset between queries.

fbsql_list_dbs (PHP 4 >= 4.0.6)

List databases available on a FrontBase server

```
resource fbsql_list_dbs ([resource link_identifier])
```

fbsql_list_dbs() will return a result pointer containing the databases available from the current fbsql daemon. Use the fbsql_tablename() function to traverse this result pointer.

Esempio 1. fbsql_list_dbs() example

```
$link = fbsql_connect('localhost', 'myname', 'secret');
$db_list = fbsql_list_dbs($link);

while ($row = fbsql_fetch_object($db_list)) {
  echo $row->Database . "\n";
}
```

The above example would produce the following output:

```
database1
database2
database3
...
```

> **Nota:** The above code would just as easily work with fbsql_fetch_row() or other similar functions.

fbsql_list_fields (PHP 4 >= 4.0.6)

List FrontBase result fields

```
resource fbsql_list_fields (string database_name, string table_name [,
resource link_identifier])
```

fbsql_list_fields() retrieves information about the given tablename. Arguments are the database name and the table name. A result pointer is returned which can be used with fbsql_field_flags(), fbsql_field_len(), fbsql_field_name(), and fbsql_field_type().

A result identifier is a positive integer. The function returns -1 if a error occurs. A string describing the error will be placed in `$phperrmsg`, and unless the function was called as `@fbsql()` then this error string will also be printed out.

Esempio 1. fbsql_list_fields() example

```
$link = fbsql_connect('localhost', 'myname', 'secret');

$fields = fbsql_list_fields("database1", "table1", $link);
$columns = fbsql_num_fields($fields);

for ($i = 0; $i < $columns; $i++) {
  echo fbsql_field_name($fields, $i) . "\n";;
}
```

The above example would produce the following output:

```
field1
field2
field3
...
```

fbsql_list_tables (PHP 4 >= 4.0.6)

List tables in a FrontBase database

```
resource fbsql_list_tables (string database [, resource link_identifier])
```

fbsql_list_tables() takes a database name and returns a result pointer much like the fbsql_db_query() function. The fbsql_tablename() function should be used to extract the actual table names from the result pointer.

fbsql_next_result (PHP 4 >= 4.0.6)

Move the internal result pointer to the next result

```
bool fbsql_next_result  (int result_id)
```

When sending more than one SQL statement to the server or executing a stored procedure with multiple results will cause the server to return multiple result sets. This function will test for additional results available form the server. if an additional result set exists it will free the existing result set and prepare to fetch the wors from the new result set. The function will return `TRUE` if an additional result set was available or `FALSE` othervise.

Esempio 1. fbsql_next_result() example

```
<?php
    $link = fbsql_connect ("localhost", "_SYSTEM", "secret");
    fbsql_select_db("MyDB", $link);
    $SQL = "Select * from table1; select * from table2;";
    $rs = fbsql_query($SQL, $link);
    do {
        while ($row = fbsql_fetch_row($rs)) {
        }
    } while (fbsql_next_result($rs));
    fbsql_free_result($rs);
    fbsql_close ($link);
?>
```

fbsql_num_fields (PHP 4 >= 4.0.6)

Get number of fields in result

```
int fbsql_num_fields (resource result)
```

fbsql_num_fields() returns the number of fields in a result set.

See also: fbsql_db_query(), fbsql_query(), fbsql_fetch_field(), fbsql_num_rows().

fbsql_num_rows (PHP 4 >= 4.0.6)

Get number of rows in result

```
int fbsql_num_rows (resource result)
```

fbsql_num_rows() returns the number of rows in a result set. This command is only valid for SELECT statements. To retrieve the number of rows returned from a INSERT, UPDATE or DELETE query, use fbsql_affected_rows().

Esempio 1. fbsql_num_rows() example

```
<?php

$link = fbsql_connect("localhost", "username", "password");
fbsql_select_db("database", $link);

$result = fbsql_query("SELECT * FROM table1;", $link);
$num_rows = fbsql_num_rows($result);

echo "$num_rows Rows\n";

?>
```

See also: fbsql_affected_rows(), fbsql_connect(), fbsql_select_db() and fbsql_query().

fbsql_pconnect (PHP 4 >= 4.0.6)

Open a persistent connection to a FrontBase Server

```
resource fbsql_pconnect ([string hostname [, string username [, string
password]]])
```

Returns: A positive FrontBase persistent link identifier on success, or FALSE on error.

fbsql_pconnect() establishes a connection to a FrontBase server. The following defaults are assumed for missing optional parameters: *host* = 'localhost', *username* = "_SYSTEM" and *password* = empty password.

fbsql_pconnect() acts very much like fbsql_connect() with two major differences.

To set Frontbase server port number, use fbsql_select_db().

First, when connecting, the function would first try to find a (persistent) link that's already open with the same host, username and password. If one is found, an identifier for it will be returned instead of opening a new connection.

Second, the connection to the SQL server will not be closed when the execution of the script ends. Instead, the link will remain open for future use.

This type of links is therefore called ’persistent’.

fbsql_query (PHP 4 >= 4.0.6)

Send a FrontBase query

```
resource fbsql_query (string query [, resource link_identifier])
```

fbsql_query() sends a query to the currently active database on the server that’s associated with the specified link identifier. If *link_identifier* isn’t specified, the last opened link is assumed. If no link is open, the function tries to establish a link as if fbsql_connect() was called with no arguments, and use it.

> **Nota:** The query string shall always end with a semicolon.

fbsql_query() returns TRUE (non-zero) or FALSE to indicate whether or not the query succeeded. A return value of TRUE means that the query was legal and could be executed by the server. It does not indicate anything about the number of rows affected or returned. It is perfectly possible for a query to succeed but affect no rows or return no rows.

The following query is syntactically invalid, so **fbsql_query()** fails and returns FALSE:

Esempio 1. fbsql_query() example

```
<?php
$result = fbsql_query ("SELECT * WHERE 1=1")
    or die ("Invalid query");
?>
```

The following query is semantically invalid if my_col is not a column in the table my_tbl, so **fbsql_query()** fails and returns FALSE:

Esempio 2. fbsql_query() example

```
<?php
$result = fbsql_query ("SELECT my_col FROM my_tbl")
    or die ("Invalid query");
?>
```

fbsql_query() will also fail and return FALSE if you don’t have permission to access the table(s) referenced by the query.

Assuming the query succeeds, you can call fbsql_num_rows() to find out how many rows were returned for a SELECT statment or fbsql_affected_rows() to find out how many rows were affected by a DELETE, INSERT, REPLACE, or UPDATE statement.

For SELECT statements, **fbsql_query()** returns a new result identifier that you can pass to fbsql_result(). When you are done with the result set, you can free the resources associated with it by calling fbsql_free_result(). Although, the memory will automatically be freed at the end of the script's execution.

See also: fbsql_affected_rows(), fbsql_db_query(), fbsql_free_result(), fbsql_result(), fbsql_select_db(), and fbsql_connect().

fbsql_result (PHP 4 >= 4.0.6)

Get result data

```
mixed fbsql_result (resource result, int row [, mixed field])
```

fbsql_result() returns the contents of one cell from a FrontBase result set. The field argument can be the field's offset, or the field's name, or the field's table dot field's name (tabledname.fieldname). If the column name has been aliased ('select foo as bar from...'), use the alias instead of the column name.

When working on large result sets, you should consider using one of the functions that fetch an entire row (specified below). As these functions return the contents of multiple cells in one function call, they're MUCH quicker than **fbsql_result()**. Also, note that specifying a numeric offset for the field argument is much quicker than specifying a fieldname or tablename.fieldname argument.

Calls to **fbsql_result()** should not be mixed with calls to other functions that deal with the result set.

Recommended high-performance alternatives: fbsql_fetch_row(), fbsql_fetch_array(), and fbsql_fetch_object().

fbsql_rollback (PHP 4 >= 4.0.6)

Rollback a transaction to the database

```
bool fbsql_rollback ([resource link_identifier])
```

Returns: TRUE on success, FALSE on failure.

fbsql_rollback() ends the current transaction by rolling back all statements issued since last commit. This command is only needed if autocommit is set to false.

See also: fbsql_autocommit() and fbsql_commit()

fbsql_select_db (PHP 4 >= 4.0.6)

Select a FrontBase database

```
bool fbsql_select_db (string database_name [, resource link_identifier])
```

Returns: TRUE on success, FALSE on error.

fbsql_select_db() sets the current active database on the server that's associated with the specified link identifier. If no link identifier is specified, the last opened link is assumed. If no link is open, the function will try to establish a link as if fbsql_connect() was called, and use it.

The client contacts FBExec to obtain the port number to use for the connection to the database. if the database name is a number the system will use that as a port number and it will not ask FBExec for the port number. The FrontBase server can be stared as FRontBase -FBExec=No -port=<port number> <database name>.

Every subsequent call to fbsql_query() will be made on the active database.

See also: fbsql_connect(), fbsql_pconnect(), and fbsql_query().

fbsql_start_db (PHP 4 >= 4.0.6)

Start a database on local or remote server

```
bool fbsql_start_db (string database_name [, resource link_identifier])
```

Returns: TRUE on success, FALSE on failure.

fbsql_start_db()

See also: fbsql_db_status() and fbsql_stop_db()

fbsql_stop_db (PHP 4 >= 4.0.6)

Stop a database on local or remote server

```
bool fbsql_stop_db (string database_name [, resource link_identifier])
```

Returns: TRUE on success, FALSE on failure.

fbsql_stop_db()

See also: fbsql_db_status() and fbsql_start_db()

fbsql_tablename (unknown)

Get table name of field

```
string fbsql_tablename (resource result, int i)
```

fbsql_tablename() takes a result pointer returned by the fbsql_list_tables() function as well as an integer index and returns the name of a table. The fbsql_num_rows() function may be used to determine the number of tables in the result pointer.

Esempio 1. fbsql_tablename() example

```
<?php
fbsql_connect ("localhost", "_SYSTEM", "");
$result = fbsql_list_tables ("wisconsin");
$i = 0;
while ($i < fbsql_num_rows ($result)) {
    $tb_names[$i] = fbsql_tablename ($result, $i);
    echo $tb_names[$i] . "<BR>";
    $i++;
}
?>
```

fbsql_warnings (PHP 4 >= 4.0.6)

Enable or disable FrontBase warnings

```
bool fbsql_warnings ([bool OnOff])
```

Returns TRUE if warnings is turned on otherwise FALSE.

fbsql_warnings() enables or disables FrontBase warnings.

XXV. filePro functions

These functions allow read-only access to data stored in filePro databases.

filePro is a registered trademark of fP Technologies, Inc. You can find more information about filePro at http://www.fptech.com/.

filepro (PHP 3, PHP 4 >= 4.0.0)

read and verify the map file

```
bool filepro (string directory)
```

This reads and verifies the map file, storing the field count and info.

No locking is done, so you should avoid modifying your filePro database while it may be opened in PHP.

> **Nota:** Quando safe-mode è abilitato, PHP controlla che i file o le directory sulle quali stai andando a lavorare, abbiano lo stesso UID dello script che è in esecuzione.

filepro_fieldname (PHP 3, PHP 4 >= 4.0.0)

gets the name of a field

```
string filepro_fieldname (int field_number)
```

Returns the name of the field corresponding to *field_number*.

filepro_fieldtype (PHP 3, PHP 4 >= 4.0.0)

gets the type of a field

```
string filepro_fieldtype (int field_number)
```

Returns the edit type of the field corresponding to *field_number*.

filepro_fieldwidth (PHP 3, PHP 4 >= 4.0.0)

gets the width of a field

```
int filepro_fieldwidth (int field_number)
```

Returns the width of the field corresponding to *field_number*.

filepro_retrieve (PHP 3, PHP 4 >= 4.0.0)

retrieves data from a filePro database

```
string filepro_retrieve (int row_number, int field_number)
```

Returns the data from the specified location in the database.

> **Nota:** Quando safe-mode è abilitato, PHP controlla che i file o le directory sulle quali stai andando a lavorare, abbiano lo stesso UID dello script che è in esecuzione.

filepro_fieldcount (PHP 3, PHP 4 >= 4.0.0)

find out how many fields are in a filePro database

```
int filepro_fieldcount ()
```

Returns the number of fields (columns) in the opened filePro database.

See also filepro().

filepro_rowcount (PHP 3, PHP 4 >= 4.0.0)

find out how many rows are in a filePro database

```
int filepro_rowcount ()
```

Returns the number of rows in the opened filePro database.

> **Nota:** Quando safe-mode è abilitato, PHP controlla che i file o le directory sulle quali stai andando a lavorare, abbiano lo stesso UID dello script che è in esecuzione.

See also filepro().

XXVI. Filesystem functions

basename (PHP 3, PHP 4 >= 4.0.0)

Returns filename component of path

```
string basename (string path [, string suffix])
```

Given a string containing a path to a file, this function will return the base name of the file. If the filename ends in *suffix* this will also be cut off.

On Windows, both slash (/) and backslash (\) are used as path separator character. In other environments, it is the forward slash (/).

Esempio 1. basename() example

```
$path = "/home/httpd/html/index.php";
$file = basename ($path);        // $file is set to "index.php"
$file = basename ($path,".php"); // $file is set to "index"
```

Nota: The *suffix* parameter was added in PHP 4.0.7.

See also: dirname()

chgrp (PHP 3, PHP 4 >= 4.0.0)

Changes file group

```
int chgrp (string filename, mixed group)
```

Attempts to change the group of the file *filename* to *group*. Only the superuser may change the group of a file arbitrarily; other users may change the group of a file to any group of which that user is a member.

Returns TRUE on success; otherwise returns FALSE.

See also chown() and chmod().

Nota: This function does not work on Windows systems

chmod (PHP 3, PHP 4 >= 4.0.0)

Changes file mode

```
int chmod (string filename, int mode)
```

Attempts to change the mode of the file specified by *filename* to that given in *mode*.

Note that *mode* is not automatically assumed to be an octal value, so strings (such as "g+w") will not work properly. To ensure the expected operation, you need to prefix *mode* with a zero (0):

```
chmod ("/somedir/somefile", 755);   // decimal; probably incorrect
chmod ("/somedir/somefile", "u+rwx,go+rx"); // string; incorrect
chmod ("/somedir/somefile", 0755);  // octal; correct value of mode
```

Returns TRUE on success and FALSE otherwise.

See also chown() and chgrp().

Nota: This function does not work on Windows systems

chown (PHP 3, PHP 4 >= 4.0.0)

Changes file owner

```
int chown (string filename, mixed user)
```

Attempts to change the owner of the file filename to user user. Only the superuser may change the owner of a file.

Returns TRUE on success; otherwise returns FALSE.

See also **chown()** and chmod().

Nota: This function does not work on Windows systems

clearstatcache (PHP 3, PHP 4 >= 4.0.0)

Clears file stat cache

```
void clearstatcache ()
```

Invoking the stat or lstat system call on most systems is quite expensive. Therefore, the result of the last call to any of the status functions (listed below) is stored for use on the next such call using the same filename. If you wish to force a new status check, for instance if the file is being checked many times and may change or disappear, use this function to clear the results of the last call from memory.

This value is only cached for the lifetime of a single request.

Affected functions include stat(), lstat(), file_exists(), is_writable(), is_readable(), is_executable(), is_file(), is_dir(), is_link(), filectime(), fileatime(), filemtime(), fileinode(), filegroup(), fileowner(), filesize(), filetype(), and fileperms().

copy (PHP 3, PHP 4 >= 4.0.0)

Copies file

```
int copy (string source, string dest)
```

Makes a copy of a file. Returns TRUE if the copy succeeded, FALSE otherwise.

Esempio 1. copy() example

```
if (!copy($file, $file.'.bak')) {
    print ("failed to copy $file...<br>\n");
}
```

See also: rename().

delete (unknown)

A dummy manual entry

```
void delete (string file)
```

This is a dummy manual entry to satisfy those people who are looking for unlink() or unset() in the wrong place.

See also: unlink() to delete files, unset() to delete variables.

dirname (PHP 3, PHP 4 >= 4.0.0)

Returns directory name component of path

```
string dirname (string path)
```

Given a string containing a path to a file, this function will return the name of the directory.

On Windows, both slash (/) and backslash (\) are used as path separator character. In other environments, it is the forward slash (/).

Esempio 1. dirname() example

```
$path = "/etc/passwd";
$file = dirname ($path); // $file is set to "/etc"
```

See also: basename()

diskfreespace (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Returns available space in directory

```
float diskfreespace (string directory)
```

Given a string containing a directory, this function will return the number of bytes available on the corresponding filesystem or disk partition.

Esempio 1. diskfreespace() example

```
$df = diskfreespace("/"); // $df contains the number of bytes
                          // available on "/"
```

disk_total_space (PHP 4 >= 4.0.7RC1)

Returns the total size of a directory

```
float disk_total_space (string directory)
```

Given a string containing a directory, this function will return the total number of bytes on the corresponding filesystem or disk partition.

Esempio 1. disk_total_space() example

```
$df = disk_total_space("/"); // $df contains the total number of
                             // bytes available on "/"
```

fclose (PHP 3, PHP 4 >= 4.0.0)

Closes an open file pointer

```
bool fclose (int fp)
```

The file pointed to by *fp* is closed.

Returns TRUE on success and FALSE on failure.

The file pointer must be valid, and must point to a file successfully opened by fopen() or fsockopen().

feof (PHP 3, PHP 4 >= 4.0.0)

Tests for end-of-file on a file pointer

```
int feof (int fp)
```

Returns TRUE if the file pointer is at EOF or an error occurs; otherwise returns FALSE.

The file pointer must be valid, and must point to a file successfully opened by fopen(), popen(), or fsockopen().

fflush (PHP 4)

Flushes the output to a file

```
int fflush (int fp)
```

This function forces a write of all buffered output to the to the resource pointed to by the file handle *fp*. Returns TRUE if succesful, FALSE otherwise.

The file pointer must be valid, and must point to a file successfully opened by fopen(), popen(), or fsockopen().

fgetc (PHP 3, PHP 4 >= 4.0.0)

Gets character from file pointer

```
string fgetc (int fp)
```

Returns a string containing a single character read from the file pointed to by fp. Returns FALSE on EOF.

The file pointer must be valid, and must point to a file successfully opened by fopen(), popen(), or fsockopen().

See also fread(), fopen(), popen(), fsockopen(), and fgets().

fgetcsv (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Gets line from file pointer and parse for CSV fields

```
array fgetcsv (int fp, int length [, string delimiter])
```

Similar to fgets() except that **fgetcsv()** parses the line it reads for fields in CSV format and returns an array containing the fields read. The field delimiter is a comma, unless you specify another delimiter with the optional third parameter.

Fp must be a valid file pointer to a file successfully opened by fopen(), popen(), or fsockopen()

Length must be greater than the longest line to be found in the CSV file (allowing for trailing line-end characters).

fgetcsv() returns FALSE on error, including end of file.

N.B. A blank line in a CSV file will be returned as an array comprising a single NULL field, and will not be treated as an error.

Esempio 1. fgetcsv() example - Read and print entire contents of a CSV file

```
$row = 1;
$fp = fopen ("test.csv","r");
while ($data = fgetcsv ($fp, 1000, ",")) {
    $num = count ($data);
    print "<p> $num fields in line $row: <br>";
    $row++;
    for ($c=0; $c<$num; $c++) {
        print $data[$c] . "<br>";
    }
}
fclose ($fp);
```

fgets (PHP 3, PHP 4 >= 4.0.0)

Gets line from file pointer

```
string fgets (int fp, int length)
```

Returns a string of up to length - 1 bytes read from the file pointed to by fp. Reading ends when length - 1 bytes have been read, on a newline (which is included in the return value), or on EOF (whichever comes first).

If an error occurs, returns FALSE.

Common Pitfalls:

People used to the 'C' semantics of fgets should note the difference in how EOF is returned.

The file pointer must be valid, and must point to a file successfully opened by fopen(), popen(), or fsockopen().

A simple example follows:

Esempio 1. Reading a file line by line

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

See also fread(), fopen(), popen(), fgetc(), fsockopen(), and socket_set_timeout().

fgetss (PHP 3, PHP 4 >= 4.0.0)

Gets line from file pointer and strip HTML tags

```
string fgetss (int fp, int length [, string allowable_tags])
```

Identical to fgets(), except that fgetss attempts to strip any HTML and PHP tags from the text it reads.

You can use the optional third parameter to specify tags which should not be stripped.

Nota: *allowable_tags* was added in PHP 3.0.13, PHP4B3.

See also fgets(), fopen(), fsockopen(), popen(), and strip_tags().

file (PHP 3, PHP 4 >= 4.0.0)

Reads entire file into an array

```
array file (string filename [, int use_include_path])
```

Identical to readfile(), except that **file()** returns the file in an array. Each element of the array corresponds to a line in the file, with the newline still attached.

You can use the optional second parameter and set it to "1", if you want to search for the file in the include_path, too.

```
<?php
// get a web page into an array and print it out
$fcontents = file ('http://www.php.net');
while (list ($line_num, $line) = each ($fcontents)) {
    echo "<b>Line $line_num:</b> " . htmlspecialchars ($line) . "<br>\n";
}

// get a web page into a string
$fcontents = join ('', file ('http://www.php.net'));
?>
```

See also readfile(), fopen(), fsockopen(), and popen().

file_exists (PHP 3, PHP 4 >= 4.0.0)

Checks whether a file exists

```
bool file_exists (string filename)
```

Returns TRUE if the file specified by *filename* exists; FALSE otherwise.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

The results of this function are cached. See clearstatcache() for more details.

fileatime (PHP 3, PHP 4 >= 4.0.0)

Gets last access time of file

```
int fileatime (string filename)
```

Returns the time the file was last accessed, or FALSE in case of an error. The time is returned as a Unix timestamp.

The results of this function are cached. See clearstatcache() for more details.

Note: The atime of a file is supposed to change whenever the data blocks of a file are being read. This can be costly performancewise when an application regularly accesses a very large number of files or directories. Some Unix filesystems can be mounted with atime updates disabled to increase the performance of such applications; USENET news spools are a common example. On such filesystems this function will be useless.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

filectime (PHP 3, PHP 4 >= 4.0.0)

Gets inode change time of file

```
int filectime (string filename)
```

Returns the time the file was last changed, or FALSE in case of an error. The time is returned as a Unix timestamp.

The results of this function are cached. See clearstatcache() for more details.

Note: In most Unix filesystems, a file is considered changed when its inode data is changed; that is, when the permissions, owner, group, or other metadata from the inode is updated. See also filemtime() (which is what you want to use when you want to create "Last Modified" footers on web pages) and fileatime().

Note also that in some Unix texts the ctime of a file is referred to as being the creation time of the file. This is wrong. There is no creation time for Unix files in most Unix filesystems.

This function will not work on remote files; the file to be examined must be accessible via the server’s filesystem.

filegroup (PHP 3, PHP 4 >= 4.0.0)

Gets file group

```
int filegroup (string filename)
```

Returns the group ID of the owner of the file, or FALSE in case of an error. The group ID is returned in numerical format, use posix_getgrgid() to resolve it to a group name.

The results of this function are cached. See clearstatcache() for more details.

> **Nota:** This function does not work on Windows systems
>
> This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

fileinode (PHP 3, PHP 4 >= 4.0.0)

Gets file inode

```
int fileinode (string filename)
```

Returns the inode number of the file, or FALSE in case of an error.

The results of this function are cached. See clearstatcache() for more details.

This function will not work on remote files; the file to be examined must be accessible via the server’s filesystem.

> **Nota:** This function does not work on Windows systems

filemtime (PHP 3, PHP 4 >= 4.0.0)

Gets file modification time

```
int filemtime (string filename)
```

Returns the time the file was last modified, or FALSE in case of an error. The time is returned as a Unix timestamp.

The results of this function are cached. See clearstatcache() for more details.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

Note: This function returns the time when the data blocks of a file were being written to, that is, the time when the content of the file was changed. Use date() on the result of this function to get a printable modification date for use in page footers.

fileowner (PHP 3, PHP 4 >= 4.0.0)

Gets file owner

```
int fileowner (string filename)
```

Returns the user ID of the owner of the file, or FALSE in case of an error. The user ID is returned in numerical format, use posix_getpwuid() to resolve it to a username.

The results of this function are cached. See clearstatcache() for more details.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

> **Nota:** This function does not work on Windows systems

fileperms (PHP 3, PHP 4 >= 4.0.0)

Gets file permissions

```
int fileperms (string filename)
```

Returns the permissions on the file, or FALSE in case of an error.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

The results of this function are cached. See clearstatcache() for more details.

filesize (PHP 3, PHP 4 >= 4.0.0)

Gets file size

```
int filesize (string filename)
```

Returns the size of the file, or FALSE in case of an error.

The results of this function are cached. See clearstatcache() for more details.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

filetype (PHP 3, PHP 4 >= 4.0.0)

Gets file type

```
string filetype (string filename)
```

Returns the type of the file. Possible values are fifo, char, dir, block, link, file, and unknown.

Returns FALSE if an error occurs.

The results of this function are cached. See clearstatcache() for more details.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

flock (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Portable advisory file locking

```
bool flock (int fp, int operation [, int wouldblock])
```

PHP supports a portable way of locking complete files in an advisory way (which means all accessing programs have to use the same way of locking or it will not work).

flock() operates on *fp* which must be an open file pointer. *operation* is one of the following values:

- To acquire a shared lock (reader), set *operation* to LOCK_SH (set to 1 prior to PHP 4.0.1).
- To acquire an exclusive lock (writer), set *operation* to LOCK_EX (set to 2 prior to PHP 4.0.1).
- To release a lock (shared or exclusive), set *operation* to LOCK_UN (set to 3 prior to PHP 4.0.1).
- If you don't want **flock()** to block while locking, add LOCK_NB (4 prior to PHP 4.0.1) to *operation*.

flock() allows you to perform a simple reader/writer model which can be used on virtually every platform (including most Unices and even Windows). The optional 3rd argument is set to TRUE if the lock would block (EWOULDBLOCK errno condition)

flock() returns TRUE on success and FALSE on error (e.g. when a lock could not be acquired).

> **Attenzione**
>
> On most operation systems **flock()** is implemented at the process level. When using a multithreaded server API like ISAPI you cannot rely on **flock()** to protect files against other PHP scripts running in parallel threads of the same server instance!

fopen (PHP 3, PHP 4 >= 4.0.0)

Opens file or URL

```
int fopen (string filename, string mode [, int use_include_path])
```

If *filename* begins with "http://" (not case sensitive), an HTTP 1.0 connection is opened to the specified server, the page is requested using the HTTP GET method, and a file pointer is returned to the beginning of the body of the response. A 'Host:' header is sent with the request in order to handle name-based virtual hosts.

Note that the file pointer allows you to retrieve only the *body* of the response; you cannot access the HTTP response header using this function.

Versions prior to PHP 4.0.5 do not handle HTTP redirects. Because of this, directories must include trailing slashes.

If *filename* begins with "ftp://" (not case sensitive), an ftp connection to the specified server is opened and a pointer to the requested file is returned. If the server does not support passive mode ftp, this will fail. You can open files for either reading or writing via ftp (but not both simultaneously).

If *filename* is one of "php://stdin", "php://stdout", or "php://stderr", the corresponding stdio stream will be opened. (This was introduced in PHP 3.0.13; in earlier versions, a filename such as "/dev/stdin" or "/dev/fd/0" must be used to access the stdio streams.)

If *filename* begins with anything else, the file will be opened from the filesystem, and a file pointer to the file opened is returned.

If the open fails, the function returns FALSE.

mode may be any of the following:

- 'r' - Open for reading only; place the file pointer at the beginning of the file.
- 'r+' - Open for reading and writing; place the file pointer at the beginning of the file.
- 'w' - Open for writing only; place the file pointer at the beginning of the file and truncate the file to zero length. If the file does not exist, attempt to create it.
- 'w+' - Open for reading and writing; place the file pointer at the beginning of the file and truncate the file to zero length. If the file does not exist, attempt to create it.
- 'a' - Open for writing only; place the file pointer at the end of the file. If the file does not exist, attempt to create it.
- 'a+' - Open for reading and writing; place the file pointer at the end of the file. If the file does not exist, attempt to create it.

> **Nota:** The *mode* may contain the letter 'b'. This is useful only on systems which differentiate between binary and text files (i.e. Windows. It's useless on Unix). If not needed, this will be ignored.

You can use the optional third parameter and set it to "1", if you want to search for the file in the include_path, too.

Esempio 1. fopen() example

```
$fp = fopen ("/home/rasmus/file.txt", "r");
$fp = fopen ("/home/rasmus/file.gif", "wb");
$fp = fopen ("http://www.php.net/", "r");
$fp = fopen ("ftp://user:password@example.com/", "w");
```

If you are experiencing problems with reading and writing to files and you're using the server module version of PHP, remember to make sure that the files and directories you're using are accessible to the server process.

On the Windows platform, be careful to escape any backslashes used in the path to the file, or use forward slashes.

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

See also fclose(), fsockopen(), socket_set_timeout(), and popen().

fpassthru (PHP 3, PHP 4 >= 4.0.0)

Output all remaining data on a file pointer

```
int fpassthru (int fp)
```

Reads to EOF on the given file pointer and writes the results to standard output.

If an error occurs, **fpassthru()** returns FALSE.

The file pointer must be valid, and must point to a file successfully opened by fopen(), popen(), or fsockopen(). The file is closed when **fpassthru()** is done reading it (leaving *fp* useless).

If you just want to dump the contents of a file to stdout you may want to use the readfile(), which saves you the fopen() call.

See also readfile(), fopen(), popen(), and fsockopen()

fputs (PHP 3, PHP 4 >= 4.0.0)

Writes to a file pointer

```
int fputs (int fp, string str [, int length])
```

fputs() is an alias to fwrite(), and is identical in every way. Note that the *length* parameter is optional and if not specified the entire string will be written.

fread (PHP 3, PHP 4 >= 4.0.0)

Binary-safe file read

```
string fread (int fp, int length)
```

fread() reads up to *length* bytes from the file pointer referenced by *fp*. Reading stops when *length* bytes have been read or EOF is reached, whichever comes first.

```
// get contents of a file into a string
$filename = "/usr/local/something.txt";
$fd = fopen ($filename, "r");
$contents = fread ($fd, filesize ($filename));
fclose ($fd);
```

> **Nota:** On systems which differentiate between binary and text files (i.e. Windows) the file must be opened with 'b' included in fopen() mode parameter.

```
$filename = "c:\\files\\somepic.gif";
$fd = fopen ($filename, "rb");
$contents = fread ($fd, filesize ($filename));
fclose ($fd);
```

See also fwrite(), fopen(), fsockopen(), popen(), fgets(), fgetss(), fscanf(), file(), and fpassthru().

fscanf (PHP 4)

Parses input from a file according to a format

```
mixed fscanf (int handle, string format [, string var1...])
```

The function **fscanf()** is similar to sscanf(), but it takes its input from a file associated with *handle* and interprets the input according to the specified *format*. If only two parameters were passed to this function, the values parsed will be returned as an array. Otherwise, if optional parameters are passed, the function will return the number of assigned values. The optional parameters must be passed by reference.

Esempio 1. fscanf() Example

```
$fp = fopen ("users.txt","r");
while ($userinfo = fscanf ($fp, "%s\t%s\t%s\n")) {
    list ($name, $profession, $countrycode) = $userinfo;
    //... do something with the values
}
fclose($fp);
```

Esempio 2. users.txt

```
javier  argonaut        pe
hiroshi sculptor        jp
robert  slacker us
luigi   florist it
```

See also fread(), fgets(), fgetss(), sscanf(), printf(), and sprintf().

fseek (PHP 3, PHP 4 >= 4.0.0)

Seeks on a file pointer

```
int fseek (int fp, int offset [, int whence])
```

Sets the file position indicator for the file referenced by *fp*.The new position, measured in bytes from the beginning of the file, is obtained by adding *offset* to the position specified by *whence*, whose values are defined as follows:

SEEK_SET - Set position equal to *offset bytes.*
SEEK_CUR - Set position to current location plus *offset.*
SEEK_END - Set position to end-of-file plus *offset.*

If *whence is not specified, it is assumed to be SEEK_SET.*

Upon success, returns 0; otherwise, returns -1. Note that seeking past EOF is not considered an error.

May not be used on file pointers returned by fopen() if they use the "http://" or "ftp://" formats.

> **Nota:** The *whence* argument was added after PHP 4.0 RC1.

See also ftell() and rewind().

fstat (PHP 4 >= 4.0.0)

Gets information about a file using an open file pointer

```
array fstat (int fp)
```

Gathers the statistics of the file opened by the file pointer fp. This function is similar to the stat() function except that it operates on an open file pointer instead of a filename.

Returns an array with the statistics of the file with the following elements:

1. device
2. inode
3. number of links
4. user id of owner
5. group id owner
6. device type if inode device *
7. size in bytes
8. time of last access
9. time of last modification
10. time of last change
11. blocksize for filesystem I/O *
12. number of blocks allocated

* - only valid on systems supporting the st_blksize type--other systems (i.e. Windows) return -1

The results of this function are cached. See clearstatcache() for more details.

ftell (PHP 3, PHP 4 >= 4.0.0)

Tells file pointer read/write position

```
int ftell (int fp)
```

Returns the position of the file pointer referenced by fp; i.e., its offset into the file stream.

If an error occurs, returns FALSE.

The file pointer must be valid, and must point to a file successfully opened by fopen() or popen().

See also fopen(), popen(), fseek() and rewind().

ftruncate (PHP 4 >= 4.0.0)

Truncates a file to a given length.

```
int ftruncate (int fp, int size)
```

Takes the filepointer, fp, and truncates the file to length, size. This function returns TRUE on success and FALSE on failure.

fwrite (PHP 3, PHP 4 >= 4.0.0)

Binary-safe file write

```
int fwrite (int fp, string string [, int length])
```

fwrite() writes the contents of *string* to the file stream pointed to by *fp*. If the *length* argument is given, writing will stop after *length* bytes have been written or the end of *string* is reached, whichever comes first.

Note that if the *length* argument is given, then the magic_quotes_runtime configuration option will be ignored and no slashes will be stripped from *string*.

> **Nota:** On systems which differentiate between binary and text files (i.e. Windows) the file must be opened with 'b' included in fopen() mode parameter.

See also fread(), fopen(), fsockopen(), popen(), and fputs().

set_file_buffer (PHP 3>= 3.0.8, PHP 4)

Sets file buffering on the given file pointer

```
int set_file_buffer (int fp, int buffer)
```

Output using fwrite() is normally buffered at 8K. This means that if there are two processess wanting to write to the same output stream (a file), each is paused after 8K of data to allow the other to write. **set_file_buffer()** sets the buffering for write operations on the given filepointer *fp* to *buffer* bytes. If *buffer* is 0 then write operations are unbuffered. This ensures that all writes with fwrite() are completed before other processes are allowed to write to that output stream.

The function returns 0 on success, or EOF if the request cannot be honored.

The following example demonstrates how to use **set_file_buffer()** to create an unbuffered stream.

Esempio 1. set_file_buffer() example

```
$fp=fopen($file, "w");
if($fp){
  set_file_buffer($fp, 0);
  fputs($fp, $output);
  fclose($fp);
}
```

See also fopen(), fwrite().

is_dir (PHP 3, PHP 4 >= 4.0.0)

Tells whether the filename is a directory

```
bool is_dir (string filename)
```

Returns TRUE if the filename exists and is a directory.

The results of this function are cached. See clearstatcache() for more details.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

See also is_file() and is_link().

is_executable (PHP 3, PHP 4 >= 4.0.0)

Tells whether the filename is executable

```
bool is_executable (string filename)
```

Returns TRUE if the filename exists and is executable.

The results of this function are cached. See clearstatcache() for more details.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

See also is_file() and is_link().

is_file (PHP 3, PHP 4 >= 4.0.0)

Tells whether the filename is a regular file

```
bool is_file (string filename)
```

Returns TRUE if the filename exists and is a regular file.

The results of this function are cached. See clearstatcache() for more details.

See also is_dir() and is_link().

is_link (PHP 3, PHP 4 >= 4.0.0)

Tells whether the filename is a symbolic link

```
bool is_link (string filename)
```

Returns TRUE if the filename exists and is a symbolic link.

The results of this function are cached. See clearstatcache() for more details.

See also is_dir() and is_file().

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

> **Nota:** This function does not work on Windows systems

is_readable (PHP 3, PHP 4 >= 4.0.0)

Tells whether the filename is readable

```
bool is_readable (string filename)
```

Returns TRUE if the filename exists and is readable.

Keep in mind that PHP may be accessing the file as the user id that the web server runs as (often 'nobody'). Safe mode limitations are not taken into account.

The results of this function are cached. See clearstatcache() for more details.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

See also is_writable().

is_writable (PHP 4 >= 4.0.0)

Tells whether the filename is writable

```
bool is_writable (string filename)
```

Returns TRUE if the filename exists and is writable. The filename argument may be a directory name allowing you to check if a directory is writeable.

Keep in mind that PHP may be accessing the file as the user id that the web server runs as (often 'nobody'). Safe mode limitations are not taken into account.

The results of this function are cached. See clearstatcache() for more details.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

See also is_readable().

is_writeable (PHP 3, PHP 4 >= 4.0.0)

Tells whether the filename is writable

```
bool is_writeable (string filename)
```

This is an alias for is_writable()

is_uploaded_file (PHP 3>= 3.0.17, PHP 4 >= 4.0.3)

Tells whether the file was uploaded via HTTP POST.

```
bool is_uploaded_file (string filename)
```

This function is available only in versions of PHP 3 after PHP 3.0.16, and in versions of PHP 4 after 4.0.2.

Returns `TRUE` if the file named by `filename` was uploaded via HTTP POST. This is useful to help ensure that a malicious user hasn't tried to trick the script into working on files upon which it should not be working--for instance, `/etc/passwd`.

This sort of check is especially important if there is any chance that anything done with uploaded files could reveal their contents to the user, or even to other users on the same system.

See also move_uploaded_file(), and the section Handling file uploads for a simple usage example.

link (PHP 3, PHP 4 >= 4.0.0)

Create a hard link

```
int link (string target, string link)
```

link() creates a hard link.

See also the symlink() to create soft links, and readlink() along with linkinfo().

> **Nota:** This function does not work on Windows systems

linkinfo (PHP 3, PHP 4 >= 4.0.0)

Gets information about a link

```
int linkinfo (string path)
```

linkinfo() returns the st_dev field of the UNIX C stat structure returned by the lstat system call. This function is used to verify if a link (pointed to by *path*) really exists (using the same method as the S_ISLNK macro defined in stat.h). Returns 0 or `FALSE` in case of error.

See also symlink(), link(), and readlink().

> **Nota:** This function does not work on Windows systems

mkdir (PHP 3, PHP 4 >= 4.0.0)

Makes directory

```
int mkdir (string pathname, int mode)
```

Attempts to create the directory specified by pathname.

Note that you probably want to specify the mode as an octal number, which means it should have a leading zero. The mode is also modified by the current umask, which you can change using umask().

```
mkdir ("/path/to/my/dir", 0700);
```

Returns TRUE on success and FALSE on failure.

See also rmdir().

move_uploaded_file (PHP 4 >= 4.0.3)

Moves an uploaded file to a new location.

```
bool move_uploaded_file (string filename, string destination)
```

This function is available only in versions of PHP 3 after PHP 3.0.16, and in versions of PHP 4 after 4.0.2.

This function checks to ensure that the file designated by *filename* is a valid upload file (meaning that it was uploaded via PHP's HTTP POST upload mechanism). If the file is valid, it will be moved to the filename given by *destination*.

If *filename* is not a valid upload file, then no action will occur, and **move_uploaded_file()** will return FALSE.

If *filename* is a valid upload file, but cannot be moved for some reason, no action will occur, and **move_uploaded_file()** will return FALSE. Additionally, a warning will be issued.

This sort of check is especially important if there is any chance that anything done with uploaded files could reveal their contents to the user, or even to other users on the same system.

> **Nota:** Quando safe-mode è abilitato, PHP controlla che i file o le directory sulle quali stai andando a lavorare, abbiano lo stesso UID dello script che è in esecuzione.

See also is_uploaded_file(), and the section Handling file uploads for a simple usage example.

pathinfo (PHP 4 >= 4.0.3)

Returns information about a file path

```
array pathinfo (string path)
```

pathinfo() returns an associative array containing information about *path*. The following array elements are returned: *dirname*, *basename* and *extension*.

Esempio 1. pathinfo() Example

```
<?php

$path_parts = pathinfo("/www/htdocs/index.html");

echo $path_parts["dirname"] . "\n";
echo $path_parts["basename"] . "\n";
echo $path_parts["extension"] . "\n";

?>
```

Would produce:

```
/www/htdocs
index.html
html
```

See also dirname(), basename() and realpath().

pclose (PHP 3, PHP 4 >= 4.0.0)

Closes process file pointer

```
int pclose (int fp)
```

Closes a file pointer to a pipe opened by popen().

The file pointer must be valid, and must have been returned by a successful call to popen().

Returns the termination status of the process that was run.

See also popen().

popen (PHP 3, PHP 4 >= 4.0.0)

Opens process file pointer

```
int popen (string command, string mode)
```

Opens a pipe to a process executed by forking the command given by command.

Returns a file pointer identical to that returned by fopen(), except that it is unidirectional (may only be used for reading or writing) and must be closed with pclose(). This pointer may be used with fgets(), fgetss(), and fputs().

If an error occurs, returns FALSE.

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

See also pclose().

readfile (PHP 3, PHP 4 >= 4.0.0)

Outputs a file

```
int readfile (string filename [, int use_include_path])
```

Reads a file and writes it to standard output.

Returns the number of bytes read from the file. If an error occurs, FALSE is returned and unless the function was called as @readfile, an error message is printed.

If *filename* begins with "http://" (not case sensitive), an HTTP 1.0 connection is opened to the specified server and the text of the response is written to standard output.

Versions prior to PHP 4.0.5 do not handle HTTP redirects. Because of this, directories must include trailing slashes.

If *filename* begins with "ftp://" (not case sensitive), an ftp connection to the specified server is opened and the requested file is written to standard output. If the server does not support passive mode ftp, this will fail.

If *filename* begins with neither of these strings, the file will be opened from the filesystem and its contents written to standard output.

You can use the optional second parameter and set it to "1", if you want to search for the file in the include_path, too.

See also fpassthru(), file(), fopen(), include(), require(), and virtual().

readlink (PHP 3, PHP 4 >= 4.0.0)

Returns the target of a symbolic link

```
string readlink (string path)
```

readlink() does the same as the readlink C function and returns the contents of the symbolic link path or 0 in case of error.

See also symlink(), **readlink()** and linkinfo().

> **Nota:** This function does not work on Windows systems

rename (PHP 3, PHP 4 >= 4.0.0)

Renames a file

```
int rename (string oldname, string newname)
```

Attempts to rename *oldname* to *newname*.

Returns TRUE on success and FALSE on failure.

rewind (PHP 3, PHP 4 >= 4.0.0)

Rewind the position of a file pointer

```
int rewind (int fp)
```

Sets the file position indicator for fp to the beginning of the file stream.

If an error occurs, returns 0.

The file pointer must be valid, and must point to a file successfully opened by fopen().

See also fseek() and ftell().

rmdir (PHP 3, PHP 4 >= 4.0.0)

Removes directory

```
int rmdir (string dirname)
```

Attempts to remove the directory named by pathname. The directory must be empty, and the relevant permissions must permit. this.

If an error occurs, returns 0.

See also mkdir().

stat (PHP 3, PHP 4 >= 4.0.0)

Gives information about a file

```
array stat (string filename)
```

Gathers the statistics of the file named by filename.

Returns an array with the statistics of the file with the following elements:

1. device
2. inode
3. inode protection mode
4. number of links
5. user id of owner
6. group id owner
7. device type if inode device *
8. size in bytes
9. time of last access
10. time of last modification
11. time of last change
12. blocksize for filesystem I/O *
13. number of blocks allocated

* - only valid on systems supporting the st_blksize type--other systems (i.e. Windows) return -1.

Returns FALSE in case of error.

stat() doesn't handle URL as does fopen().

The results of this function are cached. See clearstatcache() for more details.

lstat (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Gives information about a file or symbolic link

```
array lstat (string filename)
```

Gathers the statistics of the file or symbolic link named by filename. This function is identical to the stat() function except that if the *filename* parameter is a symbolic link, the status of the symbolic link is returned, not the status of the file pointed to by the symbolic link.

Returns an array with the statistics of the file with the following elements:

1. device
2. inode
3. inode protection mode
4. number of links
5. user id of owner
6. group id owner
7. device type if inode device *
8. size in bytes
9. time of last access
10. time of last modification
11. time of last change
12. blocksize for filesystem I/O *
13. number of blocks allocated

* - only valid on systems supporting the st_blksize type--other systems (i.e. Windows) return -1

The results of this function are cached. See clearstatcache() for more details.

realpath (PHP 4 >= 4.0.0)

Returns canonicalized absolute pathname

```
string realpath (string path)
```

realpath() expands all symbolic links and resolves references to '/./', '/../' and extra '/' characters in the input *path* and return the canonicalized absolute pathname. The resulting path will have no symbolic link, '/./' or '/../' components.

Esempio 1. realpath() example

```
$real_path = realpath ("../../index.php");
```

symlink (PHP 3, PHP 4 >= 4.0.0)

Creates a symbolic link

```
int symlink (string target, string link)
```

symlink() creates a symbolic link from the existing *target* with the specified name *link*.

See also link() to create hard links, and readlink() along with linkinfo().

> **Nota:** This function does not work on Windows systems.

tempnam (PHP 3, PHP 4 >= 4.0.0)

Creates unique file name

```
string tempnam (string dir, string prefix)
```

Creates a unique temporary filename in the specified directory. If the directory does not exist, **tempnam()** may generate a filename in the system's temporary directory.

Prior to PHP 4.0.6, the behaviour of the **tempnam()** function was system dependent. On Windows the TMP environment variable will override the *dir* parameter, on Linux the TMPDIR environment variable has precedence, while SVR4 will always use your *dir* parameter if the directory it points to exists. Consult your system documentation on the tempnam(3) function if in doubt.

Returns the new temporary filename, or the `NULL` string on failure.

Esempio 1. tempnam() example

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

> **Nota:** This function's behavior changed in 4.0.3. The temporary file is also created to avoid a race condition where the file might appear in the filesystem between the time the string was generated and before the the script gets around to creating the file.

See also tmpfile().

tmpfile (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Creates a temporary file

```
int tmpfile ()
```

Creates a temporary file with an unique name in write mode, returning a file handle similar to the one returned by fopen(). The file is automatically removed when closed (using fclose()), or when the script ends.

For details, consult your system documentation on the `tmpfile(3)` function, as well as the `stdio.h` header file.

See also tempnam().

touch (PHP 3, PHP 4 >= 4.0.0)

Sets modification time of file

```
int touch (string filename [, int time])
```

Attempts to set the modification time of the file named by filename to the value given by time. If the option time is not given, uses the present time.

If the file does not exist, it is created.

Returns `TRUE` on success and `FALSE` otherwise.

Esempio 1. touch() example

```
if (touch ($FileName)) {
    print "$FileName modification time has been
           changed to todays date and time";
```

```
} else {
    print "Sorry Could Not change modification time of $FileName";
}
```

umask (PHP 3, PHP 4 >= 4.0.0)

Changes the current umask

```
int umask (int mask)
```

umask() sets PHP's umask to mask & 0777 and returns the old umask. When PHP is being used as a server module, the umask is restored when each request is finished.

umask() without arguments simply returns the current umask.

> **Nota:** This function may not work on Windows systems.

unlink (PHP 3, PHP 4 >= 4.0.0)

Deletes a file

```
int unlink (string filename)
```

Deletes *filename*. Similar to the Unix C unlink() function.

Returns 0 or FALSE on an error.

See also rmdir() for removing directories.

> **Nota:** This function may not work on Windows systems.

XXVII. Forms Data Format functions

Forms Data Format (FDF) is a format for handling forms within PDF documents. You should read the documentation at http://partners.adobe.com/asn/developer/acrosdk/forms.html for more information on what FDF is and how it is used in general.

> **Nota:** If you run into problems configuring php with fdftk support, check whether the header file FdfTk.h and the library libFdfTk.so are at the right place. They should be in fdftk-dir/include and fdftk-dir/lib. This will not be the case if you just unpack the FdfTk distribution.

The general idea of FDF is similar to HTML forms. The difference is basically the format how data is transmitted to the server when the submit button is pressed (this is actually the Form Data Format) and the format of the form itself (which is the Portable Document Format, PDF). Processing the FDF data is one of the features provided by the fdf functions. But there is more. One may as well take an existing PDF form and populated the input fields with data without modifying the form itself. In such a case one would create a FDF document (fdf_create()) set the values of each input field (fdf_set_value()) and associate it with a PDF form (fdf_set_file()). Finally it has to be sent to the browser with MimeType `application/vnd.fdf`. The Acrobat reader plugin of your browser recognizes the MimeType, reads the associated PDF form and fills in the data from the FDF document.

If you look at an FDF-document with a text editor you will find a catalogue object with the name `FDF`. Such an object may contain a number of entries like `Fields`, `F`, `Status` etc.. The most commonly used entries are `Fields` which points to a list of input fields, and `F` which contains the filename of the PDF-document this data belongs to. Those entries are referred to in the FDF documention as /F-Key or /Status-Key. Modifying this entries is done by functions like fdf_set_file() and fdf_set_status(). Fields are modified with fdf_set_value(), fdf_set_opt() etc..

The following examples shows just the evaluation of form data.

Esempio 1. Evaluating a FDF document

```
<?php
// Save the FDF data into a temp file
$fdffp = fopen("test.fdf", "w");
fwrite($fdffp, $HTTP_FDF_DATA, strlen($HTTP_FDF_DATA));
fclose($fdffp);

// Open temp file and evaluate data
// The pdf form contained several input text fields with the names
// volume, date, comment, publisher, preparer, and two checkboxes
// show_publisher and show_preparer.
$fdf = fdf_open("test.fdf");
$volume = fdf_get_value($fdf, "volume");
```

```
echo "The volume field has the value ’<B>$volume</B>’<BR>";

$date = fdf_get_value($fdf, "date");
echo "The date field has the value ’<B>$date</B>’<BR>";

$comment = fdf_get_value($fdf, "comment");
echo "The comment field has the value ’<B>$comment</B>’<BR>";

if(fdf_get_value($fdf, "show_publisher") == "On") {
  $publisher = fdf_get_value($fdf, "publisher");
  echo "The publisher field has the value ’<B>$publisher</B>’<BR>";
} else
  echo "Publisher shall not be shown.<BR>";

if(fdf_get_value($fdf, "show_preparer") == "On") {
  $preparer = fdf_get_value($fdf, "preparer");
  echo "The preparer field has the value ’<B>$preparer</B>’<BR>";
} else
  echo "Preparer shall not be shown.<BR>";
fdf_close($fdf);
?>
```

fdf_open (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Open a FDF document

```
int fdf_open (string filename)
```

The **fdf_open()** function opens a file with form data. This file must contain the data as returned from a PDF form. Currently, the file has to be created 'manually' by using fopen() and writing the content of HTTP_FDF_DATA with fwrite() into it. A mechanism like for HTML form data where for each input field a variable is created does not exist.

Esempio 1. Accessing the form data

```
<?php
// Save the FDF data into a temp file
$fdffp = fopen("test.fdf", "w");
fwrite($fdffp, $HTTP_FDF_DATA, strlen($HTTP_FDF_DATA));
fclose($fdffp);

// Open temp file and evaluate data
$fdf = fdf_open("test.fdf");
...
fdf_close($fdf);
?>
```

See also fdf_close().

fdf_close (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Close an FDF document

```
bool fdf_close (int fdf_document)
```

The **fdf_close()** function closes the FDF document.

See also fdf_open().

fdf_create (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Create a new FDF document

```
int fdf_create ()
```

The **fdf_create()** creates a new FDF document. This function is needed if one would like to populate input fields in a PDF document with data.

Esempio 1. Populating a PDF document

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

See also fdf_close(), fdf_save(), fdf_open().

fdf_save (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Save a FDF document

```
int fdf_save (string filename)
```

The **fdf_save()** function saves a FDF document. The FDF Toolkit provides a way to output the document to stdout if the parameter *filename* is '.'. This does not work if PHP is used as an apache module. In such a case one will have to write to a file and use e.g. fpassthru() to output it.

See also fdf_close() and example for fdf_create().

fdf_get_value (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Get the value of a field

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

The **fdf_get_value()** function returns the value of a field.

See also fdf_set_value().

fdf_set_value (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Set the value of a field

```
bool fdf_set_value (int fdf_document, string fieldname, string value, int isName)
```

The **fdf_set_value()** function sets the value of a field. The last parameter determines if the field value is to be converted to a PDF Name (*isName* = 1) or set to a PDF String (*isName* = 0).

See also fdf_get_value().

fdf_next_field_name (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Get the next field name

```
string fdf_next_field_name (int fdf_document, string fieldname)
```

The **fdf_next_field_name()** function returns the name of the field after the field in *fieldname* or the field name of the first field if the second paramter is NULL.

See also **fdf_set_field()**, **fdf_get_field()**.

fdf_set_ap (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Set the appearance of a field

```
bool fdf_set_ap (int fdf_document, string field_name, int face, string filename, int page_number)
```

The **fdf_set_ap()** function sets the appearance of a field (i.e. the value of the /AP key). The possible values of *face* are 1=FDFNormalAP, 2=FDFRolloverAP, 3=FDFDownAP.

fdf_set_status (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Set the value of the /STATUS key

```
bool fdf_set_status (int fdf_document, string status)
```

The **fdf_set_status()** sets the value of the /STATUS key.

See also fdf_get_status().

fdf_get_status (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Get the value of the /STATUS key

```
string fdf_get_status (int fdf_document)
```

The **fdf_get_status()** returns the value of the /STATUS key.

See also fdf_set_status().

fdf_set_file (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Set the value of the /F key

```
bool fdf_set_file (int fdf_document, string filename)
```

The **fdf_set_file()** sets the value of the /F key. The /F key is just a reference to a PDF form which is to be populated with data. In a web environment it is a URL (e.g. http:/testfdf/resultlabel.pdf).

See also fdf_get_file() and example for fdf_create().

fdf_get_file (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Get the value of the /F key

```
string fdf_get_file (int fdf_document)
```

The fdf_set_file() returns the value of the /F key.