

Manual de PHP

Stig Sæther Bakken

Alexander Aulbach

Egon Schmid

Jim Winstead

Lars Torben Wilson

Rasmus Lerdorf

Zeev Suraski

Andrei Zmievski

Jouni Ahto

Editado por

Rafael Martínez

Manual de PHP

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

Editado por Rafael Martínez

Publicado 25-02-2001

Copyright © 1997, 1998, 1999, 2000, 2001 por el Grupo de documentación de PHP

Copyright

Este manual es © Copyright 1997, 1998, 1999, 2000, 2001 del Grupo de documentación de PHP. Los miembros de este grupo se encuentran listados en la [primera página de este manual](#).

Este manual puede ser redistribuido bajo los términos de la "GNU General Public License" publicada por la "Free Software Foundation"; tanto bajo la versión 2 de esta licencia o bajo versiones posteriores.

Tabla de contenidos

Prefacio	37
Sobre este Manual	37
Sobre la traducción	37
I. Conceptos Básicos	39
1. Introducción	39
Qué es PHP?	41
Qué se puede hacer con PHP?	41
Corta historia de PHP	41
2. Instalación	43
Bajándose la última versión	45
Instalación en sistemas UNIX	45
Instrucciones Rápidas de Instalación (Versión Módulo de Apache)	45
Configuración	45
Módulo del Apache	46
Módulo fhttpd	46
CGI version	46
Opciones de soporte para Base de Datos	46
Adabas D	46
dBase	46
filePro	46
mSQL	47
MySQL	47
iODBC	47
OpenLink ODBC	47
Oracle	47
PostgreSQL	48
Solid	48
Sybase	48
Sybase-CT	48
Velocis	48
Una librería a medida de ODBC	48
ODBC Unificado	49
LDAP	49
Otras opciones de configuración	49
–with-mcrypt= <i>DIR</i>	49
–enable-sysvsem	49
–enable-sysvshm	49
–with-xml	49
–enable-maintainer-mode	50
–with-system-regex	50
–with-config-file-path	50
–with-exec-dir	50
–enable-debug	50
–enable-safe-mode	50
–enable-track-vars	51
–enable-magic-quotes	51
–enable-debugger	51
–enable-discard-path	51
–enable-bcmath	51
–enable-force-cgi-redirect	51
–disable-short-tags	52
–enable-url-includes	52
–disable-syntax-hl	52

CPPFLAGS y LDFLAGS	52
Construyendo.....	52
Probando.....	52
Comprobando la velocidad	52
Instalación en sistemas Windows 95/98/NT	53
Pasos Generales de Instalación.....	53
Windows 95/98/NT y PWS/IIS 3	53
Windows NT e IIS 4	54
Windows 9x/NT y Apache 1.3.x.....	55
Omni HTTPd 2.0b1 para Windows	55
Módulos del PHP	55
¿Problemas?	56
Lea las PMF (FAQ).....	56
Informes de error	56
Otros problemas.....	56
3. Configuración.....	57
El archivo de configuración.....	59
Directivas Generales de Configuración	59
Directivas de Configuración de Correo.....	62
Directivas de Configuración de Modo Seguro	62
Directivas de Configuración del Debugger	63
Directivas de Carga de Extensiones.....	63
Directivas de Configuración de MySQL.....	63
Directivas de Configuración de mSQL.....	63
Directivas de Configuración de Postgres	64
SESAM Configuration Directives	64
Directivas de Configuración de Sybase	64
Directivas de Configuración de Sybase-CT	65
Directivas de Configuración de Informix	65
Directivas de Configuración de Matemática BC	66
Directivas de Configuración de Capacidades de los Navegadores	66
Directivas Unificadas de Configuración de ODBC	66
4. Seguridad	69
Binarios CGI	71
Posibles ataques	71
Caso 1: solamente se sirven ficheros publicos.....	71
Caso 2: usando –enable-force-cgi-redirect	72
Caso 3: Usando doc_root or user_dir	72
Caso 4: Analizador PHP fuera del arbol web.....	72
Modulo Apache	73
II. Referencia del Lenguaje.....	75
5. Sintaxis básica.....	75
Saliendo de HTML.....	77
Separación de instrucciones	77
Comentarios	77
6. Types	79
Enteros.....	81
Números en punto flotante	81
Cadenas	81
Conversión de cadenas.....	83
Arrays	83
Arrays unidimensionales	83
Arrays Multidimensionales	84
Objetos	85
Inicialización de Objetos	85

Type juggling.....	86
Forzado de tipos.....	86
7. Variables.....	89
Conceptos Básicos.....	91
Variables predefinidas	91
Variables de Apache	92
Variables de entorno	93
Variables de PHP	93
Ámbito de las variables	94
Variables variables.....	96
Variables externas a PHP.....	96
Formularios HTML (GET y POST)	96
IMAGE SUBMIT variable names.....	97
Cookies HTTP	97
Variables de entorno	98
Puntos en los nombres de variables de entrada.....	98
Determinando los tipos de variables.....	98
8. Constantes.....	99
9. Expresiones.....	103
10. Operadores.....	107
Operadores Aritméticos.....	109
Operadores de Asignación	109
Operadores Bit a bit.....	109
Operadores de Comparación	110
Operador de ejecución.....	110
Operadores de Incremento/decremento.....	111
Operadores Lógicos.....	111
Precedencia de Operadores	111
Operadores de Cadenas	112
11. Estructuras de Control.....	113
if.....	115
else	115
elseif	115
Sintaxis Alternativa de Estructuras de Control	116
while	116
do..while	117
for	118
foreach.....	119
break	120
continue	121
switch.....	121
require().....	123
include().....	124
require_once().....	127
include_once().....	129
12. Funciones	131
Funciones definidas por el usuario	133
Parámetros de las funciones	133
Pasar parámetros por referencia.....	133
Parámetros por defecto	134
Lista de longitud variable de parámetros	135
Devolver valores.....	135
old_function.....	135
Funciones variable.....	135
13. Clases y Objetos.....	137
class	139

14. References Explained.....	141
What are References.....	143
What do References	143
What aren't References	143
Returning References	143
Unsetting References.....	144
Spotting the Reference	144
global References.....	144
\$this.....	144
III. Características.....	145
15. Manejando errores	145
16. Creando imágenes GIF	149
17. Autenticación HTTP con PHP.....	153
18. Cookies	157
19. El envío de archivos	161
Envío de archivos con el método POST	163
Errores comunes	163
Envío de más de un archivo.....	163
Soporte del método PUT	164
20. Usando archivos remotos	167
21. Manejando conexiones.....	171
22. Conexiones persistentes a bases de datos	175
IV. Referencia de las Funciones	179
I. Funciones específicas de Apache	179
apache_lookup_uri	181
apache_note	181
getallheaders	181
virtual	182
II. Funciones de matrices	183
array	185
array_count_values.....	185
array_flip	185
array_keys	186
array_merge	186
array_pad	187
array_pop	187
array_push	188
array_reverse	188
array_shift	189
array_slice	189
array_splice	190
array_unshift	191
array_values.....	191
array_walk	192
arsort	192
asort	193
compact	193
count	194
current.....	194
each	195
end	196
extract	196
in_array	197
key	197
krsort	198

ksort	198
list	198
next	199
pos	199
prev	200
rango	200
reset	200
rsort	200
shuffle	201
sizeof	201
sort	201
uasort	202
uksort	202
usort	203
III. Funciones Ortográficas	205
aspell_new	207
aspell_check	207
aspell_check-raw	207
aspell_suggest	208
IV. Funciones matemáticas de precisión arbitraria	209
bcaadd	211
bccomp	211
bcddiv	211
bcmmod	211
bcmul	211
bcpow	212
bcscale	212
bcsqrt	212
bcssub	212
V. Bzip2 Compression Functions	213
bzclose	215
bzcompress	215
bzdecompress	215
bzerrno	216
bzerror	216
bzerrstr	216
bzflush	216
bzopen	217
bzread	217
bzwrite	217
VI. Funciones de calendario	219
JDToGregorian	221
GregorianToJD	221
JDToJulian	221
JulianToJD	221
JDToJewish	222
JewishToJD	222
JDTоФранцуз	222
FrenchToJD	222
JDMonthName	222
JDDayOfWeek	223
easter_date	223
easter_days	224
VII. CCVS API Functions	225
	227
VIII. soporte de las funciones COM para Windows	229

com_load	231
com_invoke	231
com_propget	231
com_get	231
com_propput	231
com_propset	231
com_set	231
IX. Funciones de Clases/Objectos	233
get_class_methods	235
get_class_vars	235
get_object_vars	235
method_exists	235
X. Funciones de ClibPDF	237
cpdf_global_set_document_limits	241
cpdf_set_creator	241
cpdf_set_title	241
cpdf_set_subject	241
cpdf_set_keywords	241
cpdf_open	242
cpdf_close	242
cpdf_page_init	242
cpdf_finalize_page	242
cpdf_finalize	243
cpdf_output_buffer	243
cpdf_save_to_file	243
cpdf_set_current_page	243
cpdf_begin_text	244
cpdf_end_text	244
cpdf_show	244
cpdf_show_xy	244
cpdf_text	245
cpdf_set_font	245
cpdf_set_leading	245
cpdf_set_text_rendering	246
cpdf_set_horiz_scaling	246
cpdf_set_text_rise	246
cpdf_set_text_matrix	246
cpdf_set_text_pos	246
cpdf_set_char_spacing	247
cpdf_set_word_spacing	247
cpdf_continue_text	247
cpdf_stringwidth	247
cpdf_save	247
cpdf_restore	248
cpdf_translate	248
cpdf_scale	248
cpdf_rotate	248
cpdf_setflat	249
cpdf_setlinejoin	249
cpdf_setlinecap	249
cpdf_setmiterlimit	249
cpdf_setlinewidth	249
cpdf_setdash	250
cpdf_moveto	250
cpdf_rmoveto	250
cpdf_curveto	250

cpdf_lineto	251
cpdf_rlineto	251
cpdf_circle	251
cpdf_arc	251
cpdf_rect	252
cpdf_closepath	252
cpdf_stroke	252
cpdf_closepath_stroke	252
cpdf_fill	252
cpdf_fill_stroke	253
cpdf_closepath_fill_stroke	253
cpdf_clip	253
cpdf_setgray_fill	253
cpdf_setgray_stroke	254
cpdf_setgray	254
cpdf_setrgbcolor_fill	254
cpdf_setrgbcolor_stroke	254
cpdf_setrgbcolor	254
cpdf_add_outline	255
cpdf_set_page_animation	255
cpdf_import_jpeg	256
cpdf_place_inline_image	256
cpdf_add_annotation	256
XI. CURL, Client URL Library Functions	257
curl_init	259
curl_setopt	259
curl_exec	261
curl_close	261
curl_version	261
XII. Funciones de pago electrónico	263
cybercash_encr	265
cybercash_decr	265
cybercash_base64_encode	265
cybercash_base64_decode	265
XIII. Character type functions	267
ctype_alnum	269
ctype_alpha	269
ctype_cntrl	269
ctype_digit	269
ctype_lower	269
ctype_graph	269
ctype_print	269
ctype_punct	270
ctype_space	270
ctype_upper	270
ctype_xdigit	270
XIV. Funciones de la capa de abstraccion de bases de datos (dbm-style)	271
dba_close	273
dba_delete	273
dba_exists	273
dba_fetch	273
dba_firstkey	274
dba_insert	274
dba_nextkey	274
dba_popen	274
dba_open	275

dba_optimize	275
dba_replace	275
dba_sync	276
XV. Funciones de fecha y hora	277
checkdate	279
date	279
getdate	280
gettimeofday	280
gmdate	281
gmmktime	281
gmstrftime	281
microtime	282
mktime	282
strftime	283
time	284
XVI. Funciones para dBase	285
dbase_create	287
dbase_open	287
dbase_close	288
dbase_pack	288
dbase_add_record	288
dbase_replace_record	288
dbase_delete_record	288
dbase_get_record	289
dbase_get_record_with_names	289
dbase_numfields	289
dbase_numrecords	289
XVII. Funciones dbm	291
dbmopen	293
dbmclose	293
dbmexists	293
dbmfetch	293
dbminsert	293
dbmreplace	294
dbmdelete	294
dbmfirstkey	294
dbmnextkey	294
dblist	295
XVIII. Funciones con directorios	297
chdir	299
dir	299
closedir	299
opendir	299
readdir	299
rewinddir	300
XIX. Funciones de DOM XML	301
xml doc	303
xml docfile	303
xml tree	303
XX. Error Handling and Logging Functions	305
error_log	307
error_reporting	308
restore_error_handler	308
set_error_handler	308
trigger_error	310
user_error	311

XXI. Funciones filePro	313
filepro	315
filepro_fieldname	315
filepro_fieldtype	315
filepro_fieldwidth	315
filepro_retrieve	315
filepro_fieldcount	315
filepro_rowcount	316
XXII. Funciones del sistema de ficheros	317
basename	319
chgrp	319
chmod	319
chown	320
clearstatcache	320
copy	320
delete	321
dirname	321
diskfreespace	321
fclose	321
feof	322
fgetc	322
fgetcsv	322
fgets	323
fgetss	323
file	324
file_exists	324
fileatime	324
filectime	324
filegroup	325
fileinode	325
filemtime	325
fileowner	325
fileperms	325
filesize	326
filetype	326
flock	326
fopen	327
fpassthru	328
fputs	328
fread	328
fseek	328
ftell	329
fwrite	329
set_file_buffer	329
is_dir	330
is_executable	330
is_file	330
is_link	330
is_readable	331
is_writeable	331
link	331
linkinfo	331
mkdir	332
pclose	332
popen	332
readfile	333

readlink	333
rename	333
rewind	333
rmdir	334
stat	334
lstat	335
symlink	335
tempnam	335
touch	336
umask	336
unlink	336
XXIII. Funciones Forms Data Format (Formato de Datos de Formularios)	339
fdf_open	341
fdf_close	341
fdf_create	341
fdf_save	342
fdf_get_value	342
fdf_set_value	342
fdf_next_field_name	342
fdf_set_ap	343
fdf_set_status	343
fdf_get_status	343
fdf_set_file	343
fdf_get_file	344
XXIV. Funciones FTP	345
ftp_connect	347
ftp_login	347
ftp_pwd	347
ftp_cdup	347
ftp_chdir	347
ftp_mkdir	347
ftp_rmdir	348
ftp_nlist	348
ftp_rawlist	348
ftp_systype	348
ftp_pasv	348
ftp_get	349
ftp_fget	349
ftp_put	349
ftp_fput	349
ftp_size	350
ftp_mdtm	350
ftp_rename	350
ftp_delete	350
ftp_quit	350
XXV. Function Handling functions	353
call_user_func	355
create_function	355
func_get_arg	357
func_get_args	357
func_num_args	358
function_exists	358
register_shutdown_function	359
XXVI. GNU Gettext	361
bindtextdomain	363
dcgettext	363

dgettext	363
gettext	363
textdomain	363
XXVII. GMP functions	365
gmp_init	367
gmp_intval	367
gmp_strval	367
gmp_add	368
gmp_sub	368
gmp_mul	368
gmp_div_q	368
gmp_div_r	368
gmp_div_qr	369
gmp_div	369
gmp_mod	369
gmp_divexact	370
gmp_cmp	370
gmp_neg	370
gmp_abs	370
gmp_sign	370
gmp_fact	370
gmp_sqrt	371
gmp_sqrtm	371
gmp_perfect_square	371
gmp_pow	371
gmp_powm	371
gmp_prob_prime	372
gmp_gcd	372
gmp_gcdext	372
gmp_invert	372
gmp_legendre	372
gmp_jacobi	373
gmp_random	373
gmp_and	373
gmp_or	373
gmp_xor	373
gmp_setbit	374
gmp_clrbit	374
gmp_scan0	374
gmp_scan1	374
gmp_popcount	374
gmp_hamdist	375
XXVIII. Funciones HTTP	377
header	379
setcookie	379
XXIX. Funciones para Hyperwave	381
hw_Array2Objrec	385
hw_Children	385
hw_ChildrenObj	385
hw_Close	385
hw_Connect	385
hw_Cp	386
hw_Deleteobject	386
hw_DocByAnchor	386
hw_DocByAnchorObj	386
hw_DocumentAttributes	386

hw_DocumentBodyTag	387
hw_DocumentContent	387
hw_DocumentSetContent	387
hw_DocumentSize	387
hw_ErrorMsg	388
hw_EditText	388
hw_Error	388
hw_Free_Document	388
hw_GetParents	388
hw_GetParentsObj	389
hw_GetChildColl	389
hw_GetChildCollObj	389
hw_GetRemote	389
hw_GetRemoteChildren	390
hw_GetSrcByDestObj	390
hw_GetObject	390
hw_GetAndLock	391
hw_GetText	391
hw_GetObjectByQuery	392
hw_GetObjectByQueryObj	392
hw_GetObjectByQueryColl	392
hw_GetObjectByQueryCollObj	392
hw_GetChildDocColl	393
hw_GetChildDocCollObj	393
hw_GetAnchors	393
hw_GetAnchorsObj	393
hw_Mv	393
hw_Identify	394
hw_InCollections	394
hw_Info	394
hw_InsColl	394
hw_InsDoc	395
hw_InsertDocument	395
hw_InsertObject	395
hw_mapid	395
hw_Modifyobject	396
hw_New_Document	397
hw_Objrec2Array	398
hw_OutputDocument	398
hw_pConnect	398
hw_PipeDocument	398
hw_Root	399
hw_Unlock	399
hw_Who	399
hw_Username	399
XXX. Funciones para ICAP - Internet Calendar Application Protocol	401
icap_open	403
icap_close	403
icap_fetch_event	403
icap_list_events	404
icap_store_event	404
icap_delete_event	405
icap_snooze	405
icap_list_alarms	405
XXXI. Funciones de imágenes	407
GetImageSize	409

ImageArc	409
ImageChar	409
ImageCharUp	410
ImageColorAllocate	410
ImageColorAt	410
ImageColorClosest	410
ImageColorExact	411
ImageColorResolve	411
ImageColorSet	411
ImageColorsForIndex	411
ImageColorsTotal	412
ImageColorTransparent	412
ImageCopyResized	412
ImageCreate	412
ImageCreateFromGif	412
ImageDashedLine	413
ImageDestroy	413
ImageFill	413
ImageFilledPolygon	414
ImageFilledRectangle	414
ImageFillToBorder	414
ImageFontHeight	414
ImageFontWidth	414
ImageGif	415
ImageInterlace	415
ImageLine	415
ImageLoadFont	415
ImagePolygon	416
ImagePSbbox	416
ImagePSEncodeFont	417
ImagePSFreeFont	417
ImagePSLoadFont	417
ImagePSText	418
ImageRectangle	418
ImageSetPixel	419
ImageString	419
ImageStringUp	419
ImageSX	419
ImageSY	419
ImageTTFbbox	420
ImageTTFText	420
XXXII. Funciones IMAP	423
imap_append	425
imap_base64	425
imap_body	425
imap_check	425
imap_close	426
imap_createmailbox	426
imap_delete	426
imap_deletemailbox	426
imap_expunge	426
imap_fetchbody	427
imap_fetchstructure	427
imap_header	428
imap_headers	430
imap_listmailbox	430

imap_getmailboxes.....	430
imap_listsubscribed.....	431
imap_getsubscribed.....	431
imap_mail_copy.....	431
imap_mail_move.....	432
imap_num_msg.....	432
imap_num_recent.....	432
imap_open.....	432
imap_ping.....	433
imap_renamemailbox.....	433
imap_reopen.....	433
imap_subscribe.....	434
imap_undelete.....	434
imap_unsubscribe.....	434
imap_qprint.....	434
imap_8bit.....	435
imap_binary.....	435
imap_scanmailbox.....	435
imap_mailboxmsginfo.....	435
imap_rfc822_write_address.....	436
imap_rfc822_parse_adrlist.....	436
imap_setflag_full.....	436
imap_clearflag_full.....	436
imap_sort.....	437
imap_fetchheader.....	437
imap_uid.....	438
imap_msgno.....	438
imap_search.....	438
imap_last_error.....	439
imap_errors.....	439
imap_alerts.....	440
imap_status.....	440
XXXIII. Funciones para Informix	441
ifx_connect.....	443
ifx_pconnect.....	443
ifx_close.....	443
ifx_query.....	444
ifx_prepare.....	445
ifx_do.....	445
ifx_error.....	446
ifx_errormsg.....	446
ifx_affected_rows.....	447
ifx_getsqlca.....	447
ifx_fetch_row.....	448
ifx_htmlltbl_result.....	449
ifx_fieldtypes.....	449
ifx_fieldproperties.....	450
ifx_num_fields.....	450
ifx_num_rows.....	450
ifx_free_result.....	450
ifx_create_char.....	451
ifx_free_char.....	451
ifx_update_char.....	451
ifx_get_char.....	451
ifx_create_blob.....	451
ifx_copy_blob.....	452

ifx_free_blob	452
ifx_get_blob	452
ifx_update_blob.....	452
ifx_blobinfile_mode	453
ifx_textasvarchar	453
ifx_byteasvarchar	453
ifx_nullformat.....	453
ifxus_create_slob.....	453
ifx_free_slob.....	454
ifxus_close_slob	454
ifxus_open_slob	454
ifxus_tell_slob	454
ifxus_seek_slob	454
ifxus_read_slob	455
ifxus_write_slob	455
XXXIV. Funciones InterBase	457
ibase_connect	459
ibase_pconnect	459
ibase_close	459
ibase_query.....	459
ibase_fetch_row.....	459
ibase_free_result.....	459
ibase_prepare.....	459
ibase_bind.....	459
ibase_execute.....	459
ibase_free_query	460
ibase_timefmt	460
XXXV. Ingres II functions	461
ingres_connect.....	463
ingres_pconnect.....	463
ingres_close	463
ingres_query	464
ingres_num_rows	465
ingres_num_fields	465
ingres_field_name	465
ingres_field_type	466
ingres_field_nullable	466
ingres_field_length	466
ingres_field_precision	466
ingres_field_scale	467
ingres_fetch_array	467
ingres_fetch_row	468
ingres_fetch_object	468
ingres_rollback	469
ingres_commit	469
ingres_autocommit	469
XXXVI. Funciones LDAP	471
ldap_add	475
ldap_mod_add	475
ldap_mod_del	475
ldap_mod_replace	476
ldap_bind	476
ldap_close	476
ldap_connect	476
ldap_count_entries	477
ldap_delete	477

ldap_dn2ufn	477
ldap_explode_dn	477
ldap_first_attribute	478
ldap_first_entry	478
ldap_free_result	478
ldap_get_attributes	478
ldap_get_dn	479
ldap_get_entries	479
ldap_get_values	480
ldap_get_values_len	481
ldap_list	481
ldap_modify	482
ldap_next_attribute	482
ldap_next_entry	482
ldap_read	483
ldap_search	483
ldap_unbind	484
ldap_err2str	484
ldap_errno	484
ldap_error	485
XXXVII. Funciones de Correo	487
mail	489
XXXVIII. Funciones matemáticas	491
Abs	493
Acos	493
Asin	493
Atan	493
Atan2	493
base_convert	494
BinDec	494
Ceil	494
Cos	494
DecBin	495
DecHex	495
DecOct	495
Exp	495
Floor	495
getrandmax	496
HexDec	496
Log	496
Log10	496
max	497
min	497
mt_rand	497
mt_srand	497
mt_getrandmax	498
number_format	498
OctDec	498
pi	499
pow	499
rand	499
round	499
Sin	499
Sqrt	500
srand	500
Tan	500

XXXIX. MCAL functions	501
mcal_open	503
mcal_close	503
mcal_fetch_event	503
mcal_list_events	504
mcal_append_event	504
mcal_store_event	504
mcal_delete_event	504
mcal_snooze	505
mcal_list_alarms	505
mcal_event_init	505
mcal_event_set_category	505
mcal_event_set_title	505
mcal_event_set_description	506
mcal_event_set_start	506
mcal_event_set_end	506
mcal_event_set_alarm	506
mcal_event_set_class	507
mcal_is_leap_year	507
mcal_days_in_month	507
mcal_date_valid	507
mcal_time_valid	507
mcal_day_of_week	508
mcal_day_of_year	508
mcal_date_compare	508
mcal_next_recurrence	508
mcal_event_set_recur_none	508
mcal_event_set_recur_daily	509
mcal_event_set_recur_weekly	509
mcal_event_set_recur_monthly_mday	509
mcal_event_set_recur_monthly_wday	509
mcal_event_set_recur_yearly	509
mcal_fetch_current_stream_event	510
mcal_event_add_attribute	510
XL. Funciones Criptográficas	511
mcrypt_get_cipher_name	513
mcrypt_get_block_size	513
mcrypt_get_key_size	513
mcrypt_create_iv	513
mcrypt_cbc	514
mcrypt_cfb	514
mcrypt_ecb	515
mcrypt_ofb	515
XLI. Funciones Hash	517
mhash_get_hash_name	519
mhash_get_block_size	519
mhash_count	519
mhash	520
XLII. Funciones de Microsoft SQL Server	521
mssql_close	523
mssql_connect	523
mssql_data_seek	523
mssql_fetch_array	523
mssql_fetch_field	524
mssql_fetch_object	524
mssql_fetch_row	524

mssql_field_seek	525
mssql_free_result	525
mssql_num_fields	525
mssql_num_rows	525
mssql_pconnect	526
mssql_query	526
mssql_result	526
mssql_select_db	527
XLIII. Miscelánea de funciones	529
connection_aborted	531
connection_status	531
connection_timeout	531
define	531
defined	532
die	532
eval	532
exit	533
get_browser	533
ignore_user_abort	534
iptcparse	534
leak	535
pack	535
serialize	536
sleep	537
uniqid	537
unpack	537
unserialize	538
usleep	538
XLIV. mnoGoSearch Functions	541
udm_alloc_agent	543
udm_set_agent_param	543
udm_add_search_limit	545
udm_clear_search_limits	546
udm_find	546
udm_get_res_param	546
udm_get_res_field	547
udm_load_ispell_data	547
udm_free_ispell_data	549
udm_free_res	549
udm_free_agent	549
udm_errno	550
udm_error	550
XLV. funciones mSQL	551
msql	553
msql_affected_rows	553
msql_close	553
msql_connect	553
msql_create_db	554
msql_createdb	554
msql_data_seek	554
msql_dbname	554
msql_drop_db	554
msql_dropdb	555
msql_error	555
msql_fetch_array	555
msql_fetch_field	555

msql_fetch_object	556
msql_fetch_row	556
msql_fieldname	557
msql_field_seek	557
msql_fieldtable	557
msql_fieldtype	557
msql_fieldflags	557
msql_fieldlen	558
msql_free_result	558
msql_freeresult	558
msql_list_fields	558
msql_listfields	558
msql_list_dbs	559
msql_listdbs	559
msql_list_tables	559
msql_listtables	559
msql_num_fields	559
msql_num_rows	559
msql_numfields	560
msql_numrows	560
msql_pconnect	560
msql_query	560
msql_regcase	561
msql_result	561
msql_select_db	561
msql_selectdb	561
msql_tablename	562
XLVI. Funciones MySQL	563
mysql_affected_rows	565
mysql_change_user	565
mysql_close	565
mysql_connect	566
mysql_create_db	566
mysql_data_seek	567
mysql_db_query	568
mysql_drop_db	568
mysql_errno	568
mysql_error	568
mysql_fetch_array	569
mysql_fetch_field	570
mysql_fetch_lengths	570
mysql_fetch_object	570
mysql_fetch_row	571
mysql_field_name	571
mysql_field_seek	572
mysql_field_table	572
mysql_field_type	572
mysql_field_flags	573
mysql_field_len	573
mysql_free_result	573
mysql_insert_id	573
mysql_list_fields	574
mysql_list_dbs	574
mysql_list_tables	574
mysql_num_fields	574
mysql_num_rows	575

mysql_pconnect.....	575
mysql_query.....	575
mysql_result.....	576
mysql_select_db.....	577
mysql_tablename.....	577
XLVII. Funciones de Red	579
checkdnsrr	581
closelog.....	581
debugger_off.....	581
debugger_on.....	581
fsockopen.....	581
gethostbyaddr.....	582
gethostbyname.....	582
gethostbynamel.....	582
getmxrr	583
getprotobynumber.....	583
getservbyname.....	583
getservbyport.....	584
openlog.....	584
pfsockopen.....	584
set_socket_blocking	584
syslog.....	584
XLVIII. ODBC functions	587
odbc_autocommit.....	589
odbc_binmode	589
odbc_close	589
odbc_close_all.....	590
odbc_commit.....	590
odbc_connect.....	590
odbc_cursor	591
odbc_do	591
odbc_exec	591
odbc_execute	591
odbc_fetch_into	592
odbc_fetch_row	592
odbc_field_name	592
odbc_field_type	592
odbc_field_len	593
odbc_free_result	593
odbc_longreadlen	593
odbc_num_fields	593
odbc_pconnect	594
odbc_prepare	594
odbc_num_rows	594
odbc_result	594
odbc_result_all	595
odbc_rollback	595
odbc_setopt	595
XLIX. Funciones de Oracle 8	597
OCIDefineByName	599
OCIBindByName	599
OCILogon.....	600
OCIPLogon	602
OCINLogon.....	602
OCILogOff	604

OCIExecute	604
OCICommit	604
OCIRollback	605
OCINewDescriptor	605
OCIRowCount	606
OCINumCols	607
OCIResult	607
OCIFetch	607
OCIFetchInto	608
OCIFetchStatement	608
OCIColumnIsNULL	609
OCIColumnSize	609
OCIServerVersion	610
OCIStatementType	610
OCINewCursor	611
OCIFreeStatement	612
OCIFreeCursor	612
OCIColumnName	613
OCIColumnType	613
OCIParse	614
OCIError	614
OCIInternalDebug	614
L. OpenSSL functions	617
openssl_free_key	619
openssl_get_privatekey	619
openssl_get_publickey	619
openssl_open	619
openssl_seal	620
openssl_sign	620
openssl_verify	621
LI. Funciones Oracle	623
Ora_Bind	625
Ora_Close	625
Ora_ColumnName	625
Ora_ColumnType	625
Ora_Commit	626
Ora_CommitOff	626
Ora_CommitOn	626
Ora_Error	627
Ora_ErrorCode	627
Ora_Exec	627
Ora_Fetch	627
Ora_GetColumn	628
Ora_Logoff	628
Ora_Logon	628
Ora_Open	628
Ora_Parse	629
Ora_Rollback	629
LII. Ovrimos SQL functions	631
ovrimos_connect	633
ovrimos_close	633
ovrimos_close_all	633
ovrimos_longreadlen	633
ovrimos_prepare	634
ovrimos_execute	634
ovrimos_cursor	635

ovrimos_exec	635
ovrimos_fetch_into	635
ovrimos_fetch_row	636
ovrimos_result	637
ovrimos_result_all	637
ovrimos_num_rows	638
ovrimos_num_fields	638
ovrimos_field_name	639
ovrimos_field_type	639
ovrimos_field_len	639
ovrimos_field_num	639
ovrimos_free_result	639
ovrimos_commit	640
ovrimos_rollback	640
LIII. Output Control Functions	641
flush	643
ob_start	643
ob_get_contents	643
ob_get_length	643
ob_end_flush	643
ob_end_clean	644
ob_implicit_flush	644
LIV. PDF functions	645
PDF_get_info	649
PDF_set_info	649
PDF_open	649
PDF_close	650
PDF_begin_page	650
PDF_end_page	650
PDF_show	651
PDF_show_boxed	651
PDF_show_xy	651
PDF_set_font	651
PDF_set_leading	652
PDF_set_parameter	652
PDF_get_parameter	652
PDF_set_value	652
PDF_get_value	653
PDF_set_text_rendering	653
PDF_set_horiz_scaling	653
PDF_set_text_rise	653
PDF_set_text_matrix	653
PDF_set_text_pos	654
PDF_set_char_spacing	654
PDF_set_word_spacing	654
PDF_skew	654
PDF_continue_text	654
PDF_stringwidth	655
PDF_save	655
PDF_restore	655
PDF_translate	655
PDF_scale	656
PDF_rotate	656
PDF_setflat	656
PDF_setlinejoin	657
PDF_setlinecap	657

PDF_setmiterlimit	657
PDF_setlinewidth	657
PDF_setdash	657
PDF_moveto	658
PDF_curveto	658
PDF_lineto	658
PDF_circle	658
PDF_arc	658
PDF_rect	659
PDF_closepath	659
PDF_stroke	659
PDF_closepath_stroke	659
PDF_fill	660
PDF_fill_stroke	660
PDF_closepath_fill_stroke	660
PDF_endpath	660
PDF_clip	660
PDF_setgray_fill	661
PDF_setgray_stroke	661
PDF_setgray	661
PDF_setrgbcolor_fill	661
PDF_setrgbcolor_stroke	661
PDF_setrgbcolor	662
PDF_add_outline	662
PDF_set_transition	662
PDF_set_duration	663
PDF_open_gif	663
PDF_open_png	663
PDF_open_memory_image	664
PDF_open_jpeg	664
PDF_close_image	664
PDF_place_image	665
PDF_put_image	665
PDF_execute_image	665
pdf_add_annotation	666
PDF_set_border_style	666
PDF_set_border_color	666
PDF_set_border_dash	666
LV. Verisign Payflow Pro functions	669
pfpro_init	671
pfpro_cleanup	671
pfpro_process	671
pfpro_process_raw	672
pfpro_version	673
LVI. opciones e información de PHP	675
extension_loaded	677
getenv	677
get_cfg_var	677
get_current_user	677
get_magic_quotes_gpc	678
get_magic_quotes_runtime	678
getlastmod	678
getmyinode	678
getmypid	679
getmyuid	679
getrusage	679

phpinfo	679
phpversion	680
php_logo_guid	680
putenv	680
set_magic_quotes_runtime	680
set_time_limit	681
zend_logo_guid	681
LVII. Funciones POSIX	683
posix_kill	685
posix_getpid	685
posix_getppid	685
posix_getuid	685
posix_geteuid	685
posix_getgid	685
posix_getegid	686
posix_setuid	686
posix_setgid	686
posix_getgroups	686
posix_getlogin	687
posix_getpgrp	687
posix_setsid	687
posix_setpgid	687
posix_getpgid	687
posix_getsid	688
posix_uname	688
posix_times	688
posix_ctermid	689
posix_ttyname	689
posix_isatty	689
posix_getcwd	689
posix_mkfifo	689
posix_getgrnam	689
posix_getgrgid	690
posix_getpwnam	690
posix_getpwuid	691
posix_getrlimit	692
LVIII. Funciones de PostgreSQL	693
pg_Close	695
pg_CmdTuples	695
pg_Connect	695
pg_DBname	695
pg_ErrorMessage	696
pg_Exec	696
pg_Fetch_Array	696
pg_Fetch_Object	697
pg_Fetch_Row	698
pg_FieldIsNull	699
pg_FieldName	699
pg_FieldNum	699
pg_FieldPrtLen	699
pg_FieldSize	700
pg_FieldType	700
pg_FreeResult	700
pg_GetLastOid	700
pg_Host	701
pg_loclose	701

pg_locreate	701
pg_loopen	701
pg_loread	701
pg_loreadall	702
pg_lounlink	702
pg_lowrite	702
pg_NumFields	702
pg_NumRows	702
pg_Options	703
pg_pConnect	703
pg_Port	703
pg_Result	703
pg_tty	704
LIX. Funciones de ejecución de programas	705
escapeshellcmd	707
exec	707
passthru	707
system	708
LX. Pspell Functions	709
pspell_new	711
pspell_check	711
pspell_suggest	712
LXI. GNU Readline	713
readline	715
readline_add_history	715
readline_clear_history	715
readline_completion_function	715
readline_info	716
readline_list_history	716
readline_read_history	716
readline_write_history	716
LXII. Funciones GNU Recode	717
recode_string	719
recode_file	719
LXIII. Funciones de expresiones regulares compatibles con Perl	721
preg_match	723
preg_match_all	723
preg_replace	724
preg_split	725
preg_quote	725
preg_grep	726
Modificadores de Patrones	726
Sintaxis de los Patrones	727
LXIV. Funciones para expresiones regulares	745
ereg	747
ereg_replace	747
eregi	748
eregi_replace	748
split	748
sql_regcase	749
LXV. Satellite CORBA client extension	751
OrbitObject	753
OrbitEnum	753
OrbitStruct	754
satellite_caught_exception	754
satellite_exception_id	755

satellite_exception_value	755
LXVI. Funciones Semáforo y de memoria compartida	757
sem_get.....	759
sem_acquire.....	759
sem_release	759
shm_attach.....	759
shm_detach.....	760
shm_remove.....	760
shm_put_var.....	760
shm_get_var.....	760
shm_remove_var	760
LXVII. SESAM database functions	763
sesam_connect.....	767
sesam_disconnect.....	767
sesam_settransaction	767
sesam_commit.....	768
sesam_rollback.....	769
sesam_execimm.....	769
sesam_query.....	770
sesam_num_fields	771
sesam_field_name	771
sesam_diagnostic	772
sesam_fetch_result	773
sesam_affected_rows	774
sesam_errormsg.....	775
sesam_field_array	775
sesam_fetch_row	777
sesam_fetch_array	779
sesam_seek_row	780
sesam_free_result	780
LXVIII. Funciones para la Gestión de Sesiones	783
session_start	787
session_destroy.....	787
session_name	787
session_module_name	788
session_save_path	788
session_id	788
session_register	789
session_unregister	789
session_is_registered	789
session_decode	789
session_encode	790
LXIX. Shared Memory Functions	791
shmop_open	793
shmop_read	793
shmop_write	793
shmop_size	794
shmop_delete.....	794
shmop_close	795
LXX. Shockwave Flash functions	797
swf_openfile	799
swf_closefile	799
swf_labelframe	799
swf_showframe	799
swf_setframe	799
swf_getframe	799

swf_mulcolor	800
swf_addcolor	800
swf_placeobject	800
swf_modifyobject	800
swf_removeobject	801
swf_nextid	801
swf_startdoaction	801
swf_actiongotoframe	801
swf_actiongeturl	801
swf_actionnextframe	802
swf_actionprevframe	802
swf_actionplay	802
swf_actionstop	802
swf_actiontogglequality	802
swf_actionwaitforframe	803
swf_actionsettarget	803
swf_actiongotolabel	803
swf_enddoaction	803
swf_defineline	803
swf_definerect	804
swf_definepoly	804
swf_startshape	804
swf_shapelinesolid	804
swf_shapefilloff	804
swf_shapefillsolid	805
swf_shapefillbitmapclip	805
swf_shapefillbitmaptile	805
swf_shapemoveto	805
swf_shapelineto	805
swf_shapecurveto	806
swf_shapecurveto3	806
swf_shapearc	806
swf_endshape	806
swf_definefont	806
swf_setfont	807
swf_fontsize	807
swf_fontslant	807
swf_fontracking	807
swf_getfontinfo	807
swf_definetext	808
swf_textwidth	808
swf_definebitmap	808
swf_getbitmapinfo	808
swf_startsymbol	808
swf_endsymbol	809
swf_startbutton	809
swf_addbuttonrecord	809
swf_oncondition	809
swf_endbutton	810
swf_viewport	810
swf_ortho	811
swf_ortho2	811
swf_perspective	811
swf_polarview	811
swf_lookat	812
swf_pushmatrix	812

swf_popmatrix	812
swf_scale	812
swf_translate	812
swf_rotate	813
swf_posround	813
LXXI. Funciones SNMP	815
snmpget	817
snmpset	817
snmpwalk	817
snmpwalkoid	818
snmp_get_quick_print	818
snmp_set_quick_print	818
LXXII. Socket functions	821
accept_connect	825
bind	825
connect	825
listen	826
socket	826
strerror	826
LXXIII. Funciones de cadenas	829
AddCSlashes	831
AddSlashes	831
bin2hex	831
Chop	831
Chr	832
chunk_split	832
convert_cyr_string	832
count_chars	833
crc32	833
crypt	833
echo	834
explode	834
get_html_translation_table	835
get_meta_tags	835
hebrev	836
hebrevc	836
htmlentities	836
htmlspecialchars	837
implode	837
join	837
levenshtein	838
ltrim	838
md5	838
Metaphone	838
nl2br	839
Ord	839
parse_str	839
print	840
printf	840
quoted_printable_decode	840
quotemeta	840
rtrim	841
sscanf	841
setlocale	842
similar_text	842
soundex	842

sprintf	843
strcasecmp	844
strchr	844
strcmp	844
strcspn	845
strip_tags	845
stripclashes	845
striplashes	846
stristr	846
strlen	846
strmatcmp	846
strnatcasecmp	847
strncmp	847
str_pad	848
strpos	848
strrchr	849
str_repeat	849
strrev	850
strrpos	850
strspn	850
strstr	850
strtok	851
strtolower	851
strtoupper	852
str_replace	852
strr	853
substr	853
substr_count	854
substr_replace	854
trim	855
ucfirst	855
ucwords	856
wordwrap	856
LXXIV. Funciones de Sybase	857
sybase_affected_rows	859
sybase_close	859
sybase_connect	859
sybase_data_seek	859
sybase_fetch_array	860
sybase_fetch_field	860
sybase_fetch_object	860
sybase_fetch_row	861
sybase_field_seek	861
sybase_free_result	861
sybase_num_fields	862
sybase_num_rows	862
sybase_pconnect	862
sybase_query	862
sybase_result	863
sybase_select_db	863
LXXV. Funciones URL	865
base64_decode	867
base64_encode	867
parse_url	867
urldecode	867
urlencode	868

LXXVI. Funciones sobre variables.....	869
doubleval	871
empty	871
gettype	871
intval	871
is_array	872
is_double	872
is_float	872
is_int	872
is_integer	872
is_long	873
is_object	873
is_real	873
is_string	873
isset	873
settype	874
strval	874
unset	874
LXXVII. Funciones WDDX.....	877
wddx_serialize_value	879
wddx_serialize_vars	879
wddx_packet_start	879
wddx_packet_end	880
wddx_add_vars	880
wddx_deserialize	880
LXXVIII. Funciones de intérprete XML.....	881
xml_parser_create	889
xml_set_object	889
xml_set_element_handler	890
xml_set_character_data_handler	891
xml_set_processing_instruction_handler	891
xml_set_default_handler	892
xml_set_unparsed_entity_decl_handler	892
xml_set_notation_decl_handler	893
xml_set_external_entity_ref_handler	894
xml_parse	895
xml_get_error_code	895
xml_error_string	896
xml_get_current_line_number	896
xml_get_current_column_number	896
xml_get_current_byte_index	897
xml_parser_free	897
xml_parser_set_option	897
xml_parser_get_option	898
utf8_decode	898
utf8_encode	898
LXXIX. XSLT functions.....	901
xslt_closelog	903
xslt_create	903
xslt_errno	903
xslt_error	903
xslt_fetch_result	903
xslt_free	904
xslt_openlog	904
xslt_output_begintransform	904
xslt_output_endtransform	904

xslt_output_process	904
xslt_run	904
xslt_set_sax_handler	905
xslt_transform	905
LXXX. YAZ	907
yaz_addinfo	909
yaz_close	909
yaz_connect	909
yaz_errno	909
yaz_error	909
yaz_hits	910
yaz_range	910
yaz_record	910
yaz_search	910
yaz_syntax	910
yaz_wait	911
LXXXI. NIS funciona	913
yp_get_default_domain	915
yp_order	915
yp_master	915
yp_match	916
yp_first	916
yp_next	916
yp_errno	917
yp_err_string	917
LXXXII. Funciones de Compresión	919
gzclose	921
gzeof	921
gzfile	921
gzgetc	921
gzgets	921
gzgetss	922
gzopen	922
gzpassthru	923
gzputs	923
gzread	923
gzrewind	923
gzseek	924
gztell	924
gzwrite	924
readgzfile	924
V. Apéndices	927
A. Migrando de PHP/FI 2.0 a PHP 3.0	927
Acerca de las incompatibilidades en PHP 3.0	929
Tags de inicio y fin	929
sintáxis de if..endif	929
sintáxis de while (mientras)	930
Tipos de expresiones	930
Cambios en los mensajes de error	931
Evaluación booleana por corto-circuito	931
Retorno de valores en funciones verdadero/falso	931
Otras incompatibilidades	931
B. Migrating from PHP 3.0 to PHP 4.0	933
What has changed in PHP 4.0	935
Parser behavior	935

Error reporting	935
Configuration changes	935
Additional warning messages	935
Initializers	936
empty("0")	936
Missing functions	936
Functions missing due to conceptual changes	936
Deprecate functions and extensions	936
Changed status for unset()	936
PHP 3.0 extension	937
Variable substitution in strings	937
Cookies	937
C. Desarrollo en PHP	939
Añadiendo funciones al PHP3	941
Prototipo de Función	941
Argumentos de Función	941
Argumentos de Función Variables	941
Usando los Argumentos de Función	941
Manejo de Memoria en las Funciones	942
Asignando Variables en la Tabla de Símbolos	943
Devolviendo valores simples	944
Devolviendo valores complejos	945
Usando la lista de recursos	946
Utilizando la tabla de recursos persistentes	947
Añadiendo directivas de configuración en tiempo de ejecución	948
Llamando a Funciones del Usuario	948
HashTable *tabla_funciones	948
pval *objeto	949
pval *nombre_func	949
pval *valret	949
int num_params	949
pval *params[]	949
Informando de errores	949
E_NOTICE	949
E_WARNING	949
E_ERROR	949
E_PARSE	950
E_CORE_ERROR	950
E_CORE_WARNING	950
D. El debugger de PHP	951
Usando el Debugger	953
Protocolo del debugger	953

Prefacio

PHP, acrónimo de "PHP: Hypertext Preprocessor", es un lenguaje interpretado de alto nivel embebido en páginas HTML. La mayoría de su sintaxis es similar a C, Java y Perl, con solamente un par de características PHP específicas. La meta de este lenguaje es permitir escribir a los creadores de páginas web, páginas dinámicas de una manera rápida y fácil.

Sobre este Manual

Este manual está escrito en SGML usando DocBook DTD (<http://www.ora.com/davenport/>) y DSSSL (<http://www.jclark.com/dsssl/>) (Document Style and Semantics Specification Language) para su creación. Las herramientas usadas para crear las versiones HTML, TeX y RTF son Jade (<http://www.jclark.com/jade/>), escrita por James Clark (<http://www.jclark.com/bio.htm>) y The Modular DocBook Stylesheets (<http://nwalsh.com/docbook/dsssl/>) escrita por Norman Walsh (<http://nwalsh.com/>). El marco de trabajo de la documentación de PHP fue creado por Stig Sæther Bakken (<mailto:stig@php.net>).

Sobre la traducción

La traducción del manual de PHP al español ha sido posible gracias a la colaboración de un gran número de traductores, que desinteresadamente han usado su tiempo para que todos podamos tener una versión en nuestra lengua de esta documentación.

(Aqui vendra la lista de colaboradores)

Parte I. Conceptos Básicos

Capítulo 1. Introducción

Qué es PHP?

PHP (acronimo de "PHP: Hypertext Preprocessor") es un lenguaje interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor.

Una respuesta corta y concisa, pero que significa realmente? Un ejemplo nos aclarará las cosas:

Ejemplo 1-1. Un ejemplo introductorio

```
<html>
  <head>
    <title>Ejemplo PHP</title>
  </head>
  <body>
    <?php echo "Hola, este es un ejemplo con PHP!"; ?>
  </body>
</html>
```

Podemos ver que no es lo mismo que un script CGI escrito en otro lenguaje de programación como Perl o C – En vez de escribir un programa con muchos comandos para crear una salida en HTML, escribimos el código HTML con cierto código PHP embebido (introducido) en el mismo, que producirá cierta salida (en nuestro ejemplo, producir un texto). El código PHP se incluye entre [etiquetas especiales de comienzo y final](#) que nos permitirán entrar y salir del modo PHP.

Lo que distingue a PHP de la tecnología Javascript, la cual se ejecuta en la máquina cliente, es que el código PHP es ejecutado en el servidor. Si tuviesemos un script similar al de nuestro ejemplo en nuestro servidor, el cliente sólamente recibiría el resultado de su ejecución en el servidor, sin ninguna posibilidad de determinar que código ha producido el resultado recibido. El servidor web puede ser incluso configurado para que procese todos los ficheros HTML con PHP.

Qué se puede hacer con PHP?

Al nivel más básico, PHP puede hacer cualquier cosa que se pueda hacer con un script CGI, como procesar la información de formularios, generar páginas con contenidos dinámicos, o mandar y recibir cookies.

Quizas la característica más potente y destacable de PHP es su soporte para una gran cantidad de bases de datos. Escribir un interfaz via web para una base de datos es una tarea simple con PHP. Las siguientes bases de datos están soportadas actualmente:

Adabas D	Ingres	Oracle (OCI7 and OCI8)
dBase	InterBase	PostgreSQL
Empress	FrontBase	Solid
FilePro	mSQL	Sybase
IBM DB2	MySQL	Velocis
Informix	ODBC	Unix dbm

PHP también soporta el uso de otros servicios que usen protocolos como IMAP, SNMP, NNTP, POP3, HTTP y derivados. Tambien se pueden abrir sockets de red directos (raw sockets) e interactuar con otros protocolos.

Corta historia de PHP

PHP fue concebido en otoño de 1994 por Rasmus Lerdorf (<mailto:rasmus@php.net>). Las primeras versiones no distribuidas al público fueron usadas en sus páginas web para mantener un control sobre quien consultaba su currículum. La primera versión disponible para el público a principios de 1995 fue conocida como "Herramientas para paginas web personales"(Personal Home Page Tools). Consistian en un analizador sintáctico muy simple que solo entendia unas cuantas macros y una serie de utilidades comunes en las páginas web de entonces, un libro de visitas, un contador y otras pequeñas cosas. El analizador sintactico fue reescrito a mediados de 1995 y fue nombrado PHP/FI version 2. FI viene de otro programa que Rasmus habia escrito y que procesaba los datos de formularios. Asi que combinó las "Herramientas

para páginas web personales", el "intérprete de formularios", añadió soporte para MySQL y PHP/FI vio la luz. PHP/FI creció a gran velocidad y la gente empezó a contribuir en el código.

Es difícil dar estadísticas exactas, pero se estima que a finales de 1996 PHP/FI se estaba usando al menos en 15.000 páginas web alrededor del mundo. A mediados de 1997 este número había crecido a más de 50.000. A mediados de 1997 el desarrollo del proyecto sufrió un profundo cambio, dejó de ser un proyecto personal de Rasmus, al cual habían ayudado un grupo de usuarios y se convirtió en un proyecto de grupo mucho más organizado. El analizador sintáctico se reescribió desde el principio por Zeev Suraski y Andi Gutmans y este nuevo analizador estableció las bases para PHP versión 3. Gran cantidad de código de PHP/FI fue portado a PHP3 y otra gran cantidad fue escrito completamente de nuevo.

Hoy en día (finales 1999), tanto PHP/FI como PHP3 se distribuyen en un gran número de productos comerciales tales como el servidor web "C2's StrongHold" y Redhat Linux. Una estimación conservativa basada en estadísticas de NetCraft (<http://www.netcraft.com/>) (ver también Estudio de NetCraft sobre servidores web (<http://www.netcraft.com/survey/>)), es que más de 1.000.000 de servidores alrededor del mundo usan PHP. Para hacernos una idea, este número es mayor que el número de servidores que utilizan el "Netscape's Enterprise server" en Internet.

A la vez que todo esto está pasando, el trabajo de desarrollo de la próxima generación de PHP está en marcha. Esta versión utiliza el potente motor de scripts Zend (<http://www.zend.com/>) para proporcionar altas prestaciones, así como soporta otros servidores web, además de Apache, que corren PHP como módulo nativo.

Capítulo 2. Instalación

Bajándose la última versión

El código fuente y las distribuciones binarias para algunas plataformas (incluído Windows) se pueden encontrar en <http://www.php.net/>.

Instalación en sistemas UNIX

Esta sección le guiará a través de la configuración e instalación del PHP. Conocimientos y software necesarios:

- Habilidades básicas en UNIX (ser capaz de manejar el "make" y un compilador de C)
- Un compilador ANSI de C
- Un servidor web

Instrucciones Rápidas de Instalación (Versión Módulo de Apache)

```

1. gunzip apache_1.3.x.tar.gz
2. tar xvf apache_1.3.x.tar
3. gunzip php-3.0.x.tar.gz
4. tar xvf php-3.0.x.tar
5. cd apache_1.3.x
6. ./configure --prefix=/www
7. cd .. /php-3.0.x
8. ./configure --with-mysql --with-apache=../apache_1.3.x --enable-track-vars
9. make
10. make install
11. cd .. /apache_1.3.x
12. ./configure --prefix=/www --activate-module=src/modules/php3/libphp3.a
13. make
14. make install

```

En lugar de este paso quizás prefiera simplemente copiar el binario httpd encima del binario existente. Si lo hace, asegúrese antes de cerrar su servidor.

```

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

```

Puede editar el archivo /usr/local/lib/php3.ini para ajustar opciones del PHP. Si prefiere tenerlo en otro sitio, utilice --with-config-file-path=/path en el paso 8.

17. Edite su archivo httpd.conf o srm.conf y añada:

```
AddType application/x-httpd-php3 .php3
```

Puede elegir la extensión que desee aquí. .php3 es simplemente nuestra sugerencia.

18. Utilice su método habitual para iniciar el servidor Apache (debe detener y reiniciar el servidor, no solamente hacerlo recargarse usando una señal HUP o USR1.)

Configuración

Hay dos maneras de configurar el PHP.

- Utilizando el script de "setup" que viene con el PHP. Este script le hace una serie de preguntas (casi como el script "install" del PHP/FI 2.0) y ejecuta el "configure" al final. Para ejecutar este script, escriba `./setup`.

Este script también creará un archivo llamado "do-conf", que contendrá las opciones pasadas a la configuración. Puede editar este archivo para cambiar algunas opciones sin tener que re-ejecutar el "setup". Escriba luego `./do-conf` para ejecutar la configuración con las nuevas opciones.

- Ejecutar el "configure" a mano. Para ver las opciones de que dispone, escriba `./configure --help`.

Los detalles sobre las distintas opciones de configuración son listados a continuación.

Módulo del Apache

Para configurar el PHP como módulo de Apache, responda "yes" a "Build as an Apache module?" (la opción `-with-apache=DIR` es la que lo configura) y especifique el directorio base de la distribución de Apache. Si ha desempacado el Apache en `/usr/local/www/apache_1.2.4`, este será su directorio base de la distribución de Apache. El directorio por defecto es `/usr/local/etc/httpd`.

Módulo fhttpd

Para configurar el PHP como módulo fhttpd, responda "yes" a "Build as an fhttpd module?" (la opción `-with-fhttpd=DIR` es la que lo configura) y especifique el directorio base del fuente del fhttpd. El directorio por defecto es `/usr/local/src/fhttpd`. Si está ejecutando fhttpd, configurar PHP como módulo le dará mejor rendimiento, más control y capacidad de ejecución remota.

CGI version

El valor por defecto es configurar el PHP como programa CGI. Si está ejecutando un servidor web para el que el PHP tiene soporte como módulo, debería elegir dicha solución por motivos de rendimiento. Sin embargo, la versión CGI permite a los usuarios del Apache el ejecutar distintas páginas con PHP bajo distintos identificadores de usuario. Por favor, asegúrese de haber leído el [capítulo sobre Seguridad](#) si va a ejecutar el PHP como CGI.

Opciones de soporte para Base de Datos

El PHP tiene soporte nativo para bastantes bases de datos (así como para ODBC):

Adabas D

`-with-adabas=DIR`

Compila con soporte para Adabas D. El parámetro es el directorio de instalación de Adabas D y por defecto vale `/usr/local/adabasd`.

Página de Adabas (<http://www.adabas.com/>)

dBase

`-with-dbase`

Habilita el soporte integrado para DBase. No se precisan librerías externas.

filePro

`-with-filepro`

Habilita el soporte integrado de sólo lectura para filePro. No se precisan librerías externas.

mSQL

`-with-msql=DIR`

Habilita el soporte para mSQL. El parámetro es el directorio de instalación de mSQL y por defecto vale `/usr/local/Hughes`. Este es el directorio por defecto de la distribución mSQL 2.0. **configure** detecta automáticamente qué versión de mSQL está ejecutándose y el PHP soporta tanto 1.0 como 2.0, pero si compila el PHP con mSQL 1.0 sólo podrá acceder a bases de datos de esa versión y viceversa.

Vea también Directivas de [Configuración de mSQL](#) en el [archivo de configuración](#).

Página de mSQL (<http://www.hughes.com.au>)

MySQL

`-with-mysql=DIR`

Habilita el soporte para MySQL. El parámetro es el directorio de instalación de MySQL y por defecto vale `/usr/local`. Este es el directorio de instalación de la distribución de MySQL.

Vea también Directivas de [Configuración de MySQL](#) en el [archivo de configuración](#).

Página de MySQL (<http://www.tcx.se>)

iODBC

`-with-iodbc=DIR`

Incluye soporte para iODBC. Esta característica se desarrolló inicialmente para el iODBC Driver Manager, un gestor de controlador de ODBC de redistribución libre que ese ejecuta bajo varios sabores de UNIX. El parámetro es el directorio de instalación de iODBC y por defecto vale `/usr/local`.

Página de FreeODBC (<http://users.ids.net/~bjepson/freeODBC/>) o página de iODBC (<http://www.iodbc.org>)

OpenLink ODBC

`-with-openlink=DIR`

Incluye soporte para OpenLink ODBC. El parámetro es el directorio de instalación de OpenLink ODBC y por defecto vale `/usr/local/openlink`.

Página de OpenLink Software (<http://www.openlinksw.com/>)

Oracle

`-with-oracle=DIR`

Incluye soporte para Oracle. Se ha probado y debería funcionar al menos con las versiones de la 7.0 a la 7.3. El parámetro es el directorio ORACLE_HOME. No necesita especificar este parámetro si su entorno de Oracle ya está ajustado.

Página de Oracle (<http://www.oracle.com>)

PostgreSQL

`-with-pgsql=DIR`

Incluye soporte para PostgreSQL. El parámetro es el directorio base de la instalación de PostgreSQL y por defecto vale `/usr/local/pgsql`.

Vea también Directivas de [Configuración de Postgres](#) en el [archivo de configuración](#).

Página de PostgreSQL (<http://www.postgreSQL.org/>)

Solid

`-with-solid=DIR`

Incluye soporte para Solid. El parámetro es el directorio de instalación y vale por defecto `/usr/local/solid`.

Página de Solid (<http://www.solidtech.com>)

Sybase

`-with-sybase=DIR`

Incluye soporte para Sybase. El parámetro es el directorio de instalación y vale por defecto `/home/sybase`.

Vea también Directivas de [Configuración de Sybase](#) en el [archivo de configuración](#).

Página de Sybase (<http://www.sybase.com>)

Sybase-CT

`-with-sybase-ct=DIR`

Incluye soporte para Sybase-CT. El parámetro es el directorio de instalación de Sybase-CT y por defecto vale `/home/sybase`.

Vea también Directivas de [Configuración de Sybase-CT](#) en el [archivo de configuración](#).

Velocis

`-with-velocis=DIR`

Incluye soporte para Velocis. El parámetro es el directorio de instalación de Velocis y vale por defecto `/usr/local/velocis`.

Página de Velocis (<http://www.raima.com>)

Una librería a medida de ODBC

`-with-custom-odbc=DIR`

Incluye soporte para una librería a medida arbitraria de ODBC. El parámetro es el directorio base y por defecto vale `/usr/local`.

Esta opción implica que se ha definido `CUSTOM_ODBC_LIBS` cuando se ejecutó el script de configuración. También deberá tener una cabecera `odbc.h` válida en algún lugar de su sendero (path) de inclusión. Si no tiene uno, créelo e incluya su cabecera específica desde ahí. Su cabecera puede requerir algunas definiciones extra, particularmente si es multiplataforma. Défínalas en `CFLAGS`.

Por ejemplo, puede usar Sybase SQL Anywhere bajo QNX como sigue: `CFLAGS=-DODBC_QNX LDFLAGS=-lunix CUSTOM_ODBC_LIBS="-lplib -lodbc" ./configure -with-custom-odbc=/usr/lib/sqlany50`

ODBC Unificado

`-disable-unified-odbc`

Deshabilita el módulo de ODBC Unificado, que es un interfaz común a todas las bases de datos con interfaces basados en ODBC, tales como Solid y Adabas D. También funciona para librerías normales de ODBC. Ha sido probado con iODBC, Solid, Adabas D y Sybase SQL Anywhere. Requiere que uno (y sólo uno) de estos módulos o el módulo de Velocis esté habilitado, o que se especifique una librería a medida de ODBC. Esta opción sólo se puede aplicar si alguna de estas opciones es usada: [-with-iodbc](#), [-with-solid](#), [-with-adabas](#), [-with-velocis](#), o [-with-custom-odbc](#).

Vea también Directivas de [Configuración de ODBC Unificado](#) en el [archivo de configuración](#).

LDAP

`-with-ldap=DIR`

Incluye soporte para LDAP (Lightweight Directory Access Protocol - Protocolo Ligero de Acceso a Directorios). El parámetro es el directorio base de instalación de LDAP, y por defecto vale `/usr/local/ldap`.

Puede encontrar más información sobre LDAP en RFC1777 (<ftp://ftp.isi.edu/in-notes/rfc1777.txt>) y en RFC1778 (<ftp://ftp.isi.edu/in-notes/rfc1778.txt>).

Otras opciones de configuración

`-with-mcrypt=DIR`

`-with-mcrypt`

Incluye soporte para la librería mcrypt. Vea la [documentación de mcrypt](#) para más información. Si utiliza el argumento opcional `DIR`, el PHP buscará `mcrypt.h` en `DIR/include`.

`-enable-sysvsem`

`-enable-sysvsem`

Incluye soporte para semáforos Sys V (soportados por muchos derivados Unix). Vea la [documentación sobre Semáforos y Memoria Compartida](#) para más información.

`-enable-sysvshm`

`-enable-sysvshm`

Incluye soporte para la memoria compartida Sys V (soportada por muchos derivados Unix). Vea la [documentación sobre Semáforos y Memoria Compartida](#) para más información.

-with-xml

`-with-xml`

Incluye soporte para un parser XML no validador que utiliza la librería expat (<http://www.jclark.com/xml/>) de James Clark. Vea la [referencia de funciones XML](#) para más detalles.

-enable-maintainer-mode

`-enable-maintainer-mode`

Activa avisos extra de dependencias y del compilador utilizados por algunos de los desarrolladores del PHP.

-with-system-regex

`-with-system-regex`

Utiliza la librería de expresiones regulares del sistema en lugar de la incluída. Si está compilando PHP como módulo de servidor, debe utilizar la misma librería cuando genere el PHP y cuando lo enlace con el servidor. Active esto si la librería del sistema proporciona características especiales que pueda necesitar. Se recomienda utilizar la librería incluída siempre que sea posible.

-with-config-file-path

`-with-config-file-path=DIR`

El path utilizado para buscar [el archivo de configuración](#) cuando arranca el PHP.

-with-exec-dir

`-with-exec-dir=DIR`

Sólo permite ejecutar programas en DIR cuando está en modo seguro. Por defecto vale `/usr/local/bin`. Esta opción sólo fija el valor por defecto. Puede ser cambiado posteriormente mediante la directiva `safe_mode_exec_dir` en el [fichero de configuración](#).

-enable-debug

`-enable-debug`

Habilita información de depuración adicional. Esto hace posible obtener información más detallada cuando hay problemas con el PHP. (Nótese que esto no tiene que ver con las facilidades de depuración o con la información disponible para los script PHP).

-enable-safe-mode

`-enable-safe-mode`

Habilita el "modo seguro" por defecto. Esto impone varias restricciones sobre lo que el PHP puede hacer, tales como abrir fichero sólo en el raíz de documentos. Lea el [capítulo de Seguridad](#) para más información. Los usuarios de CGI deberán siempre habilitar el modo seguro. Esta opción sólo fija el valor por defecto. Puede ser habilitado o deshabilitado posteriormente mediante la directiva [safe_mode](#) en el [archivo de configuración](#).

–enable-track-vars

`-enable-track-vars`

Hace que el PHP lleve el control de dónde proceden las variables GET/POST/cookie usando las matrices `HTTP_GET_VARS`, `HTTP_POST_VARS` y `HTTP_COOKIE_VARS`. Esta opción sólo fija el valor por defecto. Puede ser habilitado o deshabilitado posteriormente mediante la directiva `track_vars` en el [archivo de configuración](#).

–enable-magic-quotes

`-enable-magic-quotes`

Habilita las comillas mágicas por defecto. Esta opción sólo fija el valor por defecto. Puede ser habilitada o deshabilitada posteriormente mediante la directiva `magic_quotes_runtime` en el [archivo de configuración](#). Vea también las directivas `magic_quotes_gpc` y `magic_quotes_sybase`.

–enable-debugger

`-enable-debugger`

Habilita el soporte de depuración interno del PHP. Esta característica aún está en estado experimental. Vea también las directivas de [Configuración del Depurador](#) en el [archivo de configuración](#).

–enable-discard-path

`-enable-discard-path`

Si está habilitado, el ejecutable CGI del PHP se puede situar tranquilamente fuera del árbol de la web y la gente no podrá saltarse la seguridad del .htaccess. Lea la [sección en el capítulo de seguridad](#) sobre esta opción.

–enable-bcmath

`-enable-bcmath`

Habilita las funciones matemáticas de precisión arbitraria estilo **bc**. Vea también la opción `bcmath.scale` en el [archivo de configuración](#).

–enable-force-cgi-redirect

`-enable-force-cgi-redirect`

Habilita la comprobación de seguridad para redirecciones internas del servidor. Deberá usar esta opción si está ejecutando la versión CGI bajo Apache.

Cuando se utiliza el PHP como un ejecutable CGI, siempre comprueba primero si está siendo utilizado bajo redirección (por ejemplo bajo Apache, usando directivas Action). Esto asegura que el ejecutable del PHP no se puede usar para saltarse los mecanismos estándar de autenticación del servidor web llamando al ejecutable directamente, como en

`http://my.host/cgi-bin/php/secret/doc.html`. Este ejemplo accede al archivo `http://my.host/secret/doc.html` pero sin respetar ningún ajuste de seguridad del httpd para el directorio `/secret`.

No habilitando esta opción se deshabilita la comprobación y se permite el saltarse los ajustes de seguridad y autenticación del httpd. Haga esto sólo si el software de su servidor no puede indicar que se ha realizado una redirección segura y que todos sus archivos bajo la raíz de documentos y los directorios de los usuarios pueden ser accedidos por cualquiera.

Lea la [sección en el capítulo de seguridad](#) acerca de esta opción.

-disable-short-tags

`-disable-short-tags`

Deshabilita las etiquetas de PHP en formato corto `<? ?>`. Debe deshabilitar el formato corto si desea usar PHP con XML. Con el formato corto deshabilitado, la única etiqueta de código de PHP es `<?php ?>`. Esta opción sólo fija el valor por defecto. Puede ser habilitada o deshabilitada posteriormente mediante la directiva `short_open_tag` en el [archivo de configuración](#).

-enable-url-includes

`-enable-url-includes`

Hace posible ejecutar código en otros servidores HTTP o FTP directamente desde el PHP usando `include()`. Vea también la opción `include_path` en el [archivo de configuración](#).

-disable-syntax-hl

`-disable-syntax-hl`

Desconecta el resalte de sintaxis.

CPPFLAGS y LDFLAGS

Para hacer que la instalación de PHP busque los archivos de cabecera o de librería en distintos directorios, modifique las variables de entorno CPPFLAGS y LDFLAGS respectivamente. Si está utilizando un shell "sensible", podrá ejecutar `LDFLAGS=-L/my/lib/dir CPPFLAGS=-I/my/include/dir ./configure`

Construyendo

Cuando el PHP está configurado, ya está listo para construir el ejecutable CGI o la librería PERL. El comando **make** debería ocuparse de esto. Si fallara y no puede saber el motivo, vea la [sección de Problemas](#).

Probando

Si ha construido el PHP como un programa CGI, puede probar su funcionamiento tecleando **make test**. Siempre es buena idea probar su construcción. Así puede atrapar pronto los problemas del PHP en su plataforma sin tener que batallar con ellos luego.

Comprobando la velocidad

Si ha construido el PHP como un programa CGI, puede comprobar la velocidad de su código escribiendo **make bench**. Nótese que si el modo seguro está habilitado por defecto, el test no podrá finalizar si se toma más de los 30 segundos disponibles. Esto se debe a que la función `set_time_limit()` no se puede usar en modo seguro. Use el ajuste de

configuración `max_execution_time` para controlar este tiempo en sus propios script. `make bench` ignora el archivo de configuración.

Instalación en sistemas Windows 95/98/NT

Esta guía de instalación le ayudará a instalar y configurar el PHP en sus servidores web bajo Windows 9x/NT. Esta guía fue compilada por Bob Silva (mailto:bob_silva@mail.umesd.k12.or.us). La última revisión puede encontrarse en <http://www.umesd.k12.or.us/php/win32install.html>.

Esta guía proporciona soporte de instalación para:

- Personal Web Server (se recomienda la última versión)
- Internet Information Server 3 ó 4
- Apache 1.3.x
- Omni HTTPD 2.0b1

Pasos Generales de Instalación

Los siguientes pasos deben realizarse en todas las instalaciones antes de las instrucciones específicas de cada servidor.

- Extraiga el archivo de distribución a un directorio de su elección. "C:\PHP3\" es un buen comienzo.
- Copie el archivo 'php3.ini-dist' a su directorio '%WINDOWS%' y renómbrelo a 'php3.ini'. Su directorio '%WINDOWS%' es típicamente:
c:\windows para Windows 95/98
c:\winnt o c:\winnt40 para servidores NT
- Edite su archivo 'php3.ini':
 - Necesitará cambiar la opción 'extension_dir' para que apunte a su php-install-dir, o a donde quiera que haya puesto sus archivos 'php3_*.dll'. P.ej.: c:\php3
 - Si está utilizando Omni Httpd, no siga el siguiente paso. Fije el 'doc_root' para que apunte a la raíz web de sus servidores. P.ej.: c:\apache\htdocs o c:\webroot
 - Elija qué módulos desearía cargar cuando comience el PHP. Puede descomentar las líneas: 'extension=php3_*.dll' para cargar estos módulos. Algunos módulos requieren que tenga instaladas en su sistema librerías adicionales para que el módulo funcione correctamente. El FAQ (<http://www.php.net/FAQ.php>) de PHP tiene más información sobre dónde obtener librerías de soporte. También puede cargar un módulo dinámicamente en su script utilizando: **`dl("php_*.dll");`**
 - En el PWS y el IIS puede fijar el browscap.ini para que apunte a: 'c:\windows\system\inetsrv\browscap.ini' bajo Windows 95/98 y a 'c:\winnt\system32\inetsrv\browscap.ini' bajo NT Server. Se puede encontrar información adicional sobre el uso de la funcionalidad del browscap en el PHP en este servidor alternativo (<http://php.netvision.net.il/browser-id.php3>). Elija el botón "fuente" para verlo en acción.

Las DLL para las extensiones del PHP van precedidas de 'php3_'. Esto evita confusiones entre las extensiones del PHP y sus librerías de soporte.

Windows 95/98/NT y PWS/IIS 3

El método recomendado para configurar estos servidores es usar el archivo INF incluido con la distribución (php_iis_reg.inf). Quizás desee editar este archivo y asegurarse que las extensiones y directorios de instalación se ajustan a

su configuración. O puede seguir los pasos que siguen para hacerlo de forma manual.

AVISO: Estos pasos conllevan el trabajar directamente con el registro de windows. Un error aquí puede dejar su sistema en un estado inestable. Le recomendamos encarecidamente que haga una copia de seguridad del registro con antelación. El equipo de Desarrollo del PHP no se hará responsable si se daña su registro.

- Ejecute Regedit.
- Navegue hasta: `HKEY_LOCAL_MACHINE /System /CurrentControlSet /Services /W3Svc /Parameters /ScriptMap`.
- En el menú de edición elija: `New->String Value`.
- Escriba la extensión que desea usar para sus script PHP. P.ej.: `.php3`
- Haga doble click en el nuevo valor de cadena y escriba la ruta al `php.exe` en el campo del valor. P.ej.: `c:\php3\php.exe %s %s`. La parte '`%s %s`' son MUY importantes, pues el PHP no funcionará correctamente sin ella.
- Repita estos pasos para cada extensión que desee asociar con los scripts PHP.
- Ahora navegue hasta: `HKEY_CLASSES_ROOT`
- En el menú de edición elija: `New->Key`.
- Déle a la clave el nombre de la extensión que preparó en la sección anterior. P.ej.: `.php3`
- Marque la nueva clave y en el panel del lado derecho haga doble click en "default value"y escriba `phpfile`.
- Repita el último paso para cada extensión que haya preparado en la sección previa.
- Ahora cree otra `New->Key` bajo `HKEY_CLASSES_ROOT` y denomínela `phpfile`.
- Marque la nueva clave `phpfile` y haga doble click en el panel derecho sobre "default value"y escriba `PHP Script`.
- Pulse el botón derecho sobre la clave `phpfile` y seleccione `New->Key` y llámela `Shell`.
- Pulse el botón derecho sobre la clave `Shell` y elija `New->Key` y llámela `open`.
- Pulse el botón derecho sobre la clave `open` y elija `New->Key` y llámela `command`.
- Marque la nueva clave `command` y en el panel derecho haga doble click sobre "default value"y entre la ruta hasta el `php.exe`. P.ej.: `c:\php3\php.exe -q %1`. (no olvide el `%1`).
- Salga del Regedit.

Los usuarios de PWS e IIS3 tienen ahora un sistema completamente operativo. Los usuarios del IIS3 también pueden usar una curiosa herramienta (<http://www.genusa.com/iis/iiscfg.html>) de Steven Genusa para configurar sus mapeados de script.

Windows NT e IIS 4

Para instalar el PHP en un NT Server con IIS 4, siga estas instrucciones:

- En el Controlador de Servicios de Internet (MMC), elija el sitio Web o el directorio de comienzo de una aplicación.
- Abra las propiedades del directorio (haciendo click derecho y eligiendo propiedades) y luego pulse sobre la pestaña Carpeta Inicial, Directorio Virtual o Directorio.
- Pulse el botón Configuración y luego pulse sobre la pestaña Mapas de Aplicación.
- Pulse en Añadir, y en la caja Programa, escriba: `c:\path-to-php-dir\php.exe %s %s`. DEBE mantener los `%s %s` al final, pues el PHP no funcionará correctamente si se equivoca al hacerlo.
- En la caja Extensión, escriba la extensión de fichero que desea asociar a los script de PHP. Debe repetir los pasos 5 y 6 para cada extensión que desee asociar con los scripts PHP (`.php3` y `.phtml` son habituales).

- Ajuste la seguridad apropiada (esto se realiza en el Controlador de Servicio de Internet (ISM)), y si su NT Server usa el sistema de archivos NTFS, añada derechos de ejecución para I_USR_ al directorio que contenga el php.exe.

Windows 9x/NT y Apache 1.3.x

Debe editar sus archivos srm.conf o httpd.conf para configurar el Apache para que trabaje con el ejecutable CGI del PHP.

Aunque puede haber algunas variaciones al configurar PHP bajo Apache, esta es lo suficientemente simple para ser usada por el novato. Por favor, consulte la Documentación del Apache para saber de las siguientes directivas de configuración.

- ScriptAlias /php3/ "c:/ruta-al-dir-del-php/"
- AddType application/x-httdp-php3 .php3
- AddType application/x-httdp-php3 .phtml
- Action application/x-httdp-php3 "/php3/php.exe"

Para utilizar la capacidad de marcado del código fuente, cree simplemente un script de PHP y pegue este código en él:
`<?php show_source("script_original_php.php3"); ?>`. Sustituya script_original_php.php3 por el nombre del archivo del que desea visualizar el código fuente (esta es la única forma de hacerlo). *Nota:* Bajo Win-Apache todas las barras invertidas de una ruta tal como: "c:\directory\file.ext", deben ser convertidas a barras hacia adelante.

Omni HTTPd 2.0b1 para Windows

Esta ha resultado ser la configuración más sencilla:

Paso 1: Instale el servidor Omni

Paso 2: Pulse el botón derecho sobre el icono azul del OmniHTTPd que está en la barrita del sistema y elija Propiedades

Paso 3: Pulse sobre Web Server Global Settings

Paso 4: En la pestaña 'External', escriba: virtual = .php3 | actual = c:\ruta-al-dir-del-php\php.exe

Paso 5: En la pestaña Mime, escriba: virtual = wwwserver/stdcgi | actual = .php3

Paso 6: Pulse en OK

Repita los pasos 2 a 6 para cada extensión que desee asociar al PHP.

Módulos del PHP

Tabla 2-1. Módulos del PHP

php3_calendar.dll	Funciones de conversión de calendario
php3_crypt.dll	Funciones de criptografía
php3_dbase.dll	Funciones para DBase
php3_dbm.dll	Emulación GDBM con la librería Berkeley DB2
php3_filepro.dll	Acceso SÓLO LECTURA a bases de datos filepro
php3_gd.dll	Funciones de librería GD para manipular GIF
php3_hyperwave.dll	Funciones de HyperWave
php3_imap4r2.dll	Funciones de IMAP 4
php3_ldap.dll	Funciones de LDAP

php3_msq1.dll	Cliente de mSQL 1
php3_msq12.dll	Cliente de mSQL 2
php3_mssql.dll	Cliente de MSSQL client (requiere las librerías de MSSQL DB)
php3_mysql.dll	Funciones de MySQL
php3_nsmail.dll	Funciones de correo de Netscape
php3_oci73.dll	Funciones de Oracle
php3_snmp.dll	Funciones get y walk de SNMP (sólo en NT!)
php3_zlib.dll	Funciones de ZLib

¿Problemas?

Lea las PMF (FAQ)

Algunos problemas son más comunes que otros. Los más comunes están listados en las PMF (Preguntas Más Frecuentes) del PHP, que están en <http://www.php.net/FAQ.php>

Informes de error

Si cree que ha encontrado un error en el PHP, por favor infórmenos. Los desarrolladores del PHP probablemente no tengan conocimiento del mismo, y salvo si informa del mismo, pocas probabilidades habrá de que lo solucionen. Puede informar de los errores usando el sistema de rastreo de errores en <http://www.php.net/bugs.php>.

Otros problemas

Si aún se encuentra atascado, alguien de la lista de correos del PHP puede ser capaz de ayudarle. Deberá buscar primero en los archivos, por si acaso alguien ya ha respondido a otra persona que tuvo el mismo problema que usted. Los archivos están disponibles desde la página de soporte en <http://www.php.net/>. Para suscribirse a la lista de correo de PHP, envíe un correo vacío a php-general-subscribe@lists.php.net (<mailto:php-general-subscribe@lists.php.net>). La dirección de la lista de correo es php-general@lists.php.net.

Si desea ayuda sobre la lista de correo, intente ser preciso y de los detalles necesarios sobre su entorno (qué sistema operativo, qué versión de PHP, qué servidor web, si está ejecutando el PHP como CGI o como módulo de servidor, etc.) y también código suficiente para que otros puedan reproducir y comprobar su problema.

Capítulo 3. Configuración

El archivo de configuración

El archivo de configuración (llamado `php3.ini` en PHP 3.0, y simplemente `php.ini` a partir del PHP 4.0) es leído cuando arranca el PHP. Para las versiones de PHP como módulo de servidor esto sólo ocurre una vez al arrancar el servidor web. Para la versión CGI, esto ocurre en cada llamada.

Cuando se utiliza PHP como módulo Apache, también puede cambiar los ajustes de configuración utilizando directivas en los archivos de configuración del Apache y en los `.htaccess`.

Con el PHP 3.0 hay directivas Apache que se corresponden a cada uno de los ajustes de configuración del `php3.ini`, con la excepción que su nombre va precedido de "php3_".

Con el PHP 4.0 sólo hay unas pocas directivas de Apache que le permiten cambiar los ajustes de configuración del PHP.

`php_value nombre valor`

Fija el valor de la variable especificada.

`php_flag nombre on/off`

Fija una opción de configuración de tipo Boolean.

`php_admin_value nombre valor`

Fija el valor de la variable especificada. Los ajustes de configuración de tipo "Admin" sólo se pueden fijar desde los archivos principales de configuración del Apache, y no desde los `.htaccess`.

`php_admin_flag nombre on/off`

Fija una opción de configuración de tipo Boolean.

Puede ver los ajustes de los valores de configuración en la salida de `phpinfo()`. También puede acceder a los valores individuales de los ajustes de configuración utilizando `get_cfg_var()`.

Directivas Generales de Configuración

`asp_tags boolean`

Permite el uso de las etiquetas estilo ASP `<% %>` además de las habituales etiquetas `<?php ?>`. También se incluye el atajo para imprimir variables `<%= $valor %>`. Para más información, vea [Escapando del HTML](#).

Nota: El soporte para etiquetas estilo ASP se añadió en la 3.0.4.

`auto_append_file string`

Especifica el nombre de un archivo que es troceado automáticamente después del archivo principal. El archivo se incluye como si fuese llamado mediante la función `include()`, así que se utiliza `include_path`.

El valor especial `none` desconecta la adición automática de archivos.

Nota: Si el script es terminado con `exit()`, *no* tendrá lugar la adición automática.

`auto_prepend_file string`

Especifica el nombre de un archivo que es troceado automáticamente antes del archivo principal. Specifies the name of a file that is automatically parsed before the main file. El archivo se incluye como si fuese llamado mediante la función **include()**, así que se utiliza **include_path**.

El valor especial `none` desconecta la adición automática de archivos.

cgi_ext string

display_errors boolean

Determina si los errores se visualizan en pantalla como parte de la salida en HTML o no.

doc_root string

"Directorio raíz" del PHP en el servidor. Sólo se usa si no está vacío. Si el PHP se configura con **safe mode**, no se sirven archivos fuera de este directorio.

engine boolean

Esta directiva sólo es realmente útil en la versión de PHP como módulo Apache. Se utiliza por sitios que desean habilitar la ejecución del PHP directorio por directorio o en base a cada servidor virtual. Poniendo **php3_engine off** en los sitios apropiados del archivo `httpd.conf`, se puede habilitar o deshabilitar el PHP.

error_log string

Nombre del fichero para registrar los errores de un script. Si se utiliza el valor especial `syslog`, los errores se envían al registro de errores del sistema. En UNIX se refiere a `syslog(3)` y en Windows NT al registro de eventos. El registro de errores del sistema no es soportado bajo Windows 95.

error_reporting integer

Fija el nivel de informe de errores. El parámetro es un entero que representa un campo de bits. Sume los valores de los niveles de informe de error que desea.

Tabla 3-1. Niveles de Informe de Errores

valor de bit	informe habilitado
1	errores normales
2	avisos normales
4	errores del troceador (parser)
8	avisos de estilo no críticos

El valor por defecto para esta directiva es 7 (se muestran los errores normales, avisos normales y errores de parser).

open_basedir string

Limita los archivos que se pueden abrir por el PHP al árbol de directorios especificado.

Cuando un script intenta abrir un archivo con, por ejemplo, `fopen` o `gzopen`, se comprueba su localización. Si el fichero está fuera del árbol de directorios especificado, PHP se negará a abrirlo. Todos los enlaces simbólicos son resueltos, de modo que no es posible evitar esta limitación usando uno de ellos.

El valor especial `.` indica que el directorio base será aquel en el que reside el script.

Bajo Windows, separe los directorios mediante punto y coma. En el resto de sistemas, sepárelos con dos puntos `:`. Como módulo de Apache, los senderos para `open_basedir` de los directorios padre se heredan ahora automáticamente.

Nota: El soporte para directorios múltiples se añadió en la 3.0.7.

El valor por defecto es permitir abrir todos los archivos.

gpc_order string

Fija el orden de troceo de variables GET/POST/COOKIE. El valor por defecto de esta directiva es "GPC". Fijándola, por ejemplo, a "GP", hará que el PHP ignore por completo las cookies y que sobreescriba las variables recibidas por GET con las que tengan el mismo nombre y vengan por POST.

ignore_user_abort string

Por defecto está a on. Si se cambia a off, los script terminarán tan pronto como intenten enviar algo después de que un cliente ha roto la conexión. **ignore_user_abort()**.

include_path string

Especifica una lista de directorios en los que las funciones **require()**, **include()** y **fopen_with_path()** buscan los archivos. El formato es similar a la variable de entorno de sistema PATH: una lista de directorios separados por dos puntos en UNIX o por punto y coma en Windows.

Ejemplo 3-1. include_path en UNIX

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

Ejemplo 3-2. include_path en Windows

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

El valor por defecto para esta directiva es . (sólo el directorio actual).

isapi_ext string*log_errors* boolean

Dice si los mensajes de error de los script deben ser registrados o no en el registro del servidor. Esta opción, por tanto, es específica del mismo.

magic_quotes_gpc boolean

Fija el estado *magic_quotes* para operaciones GPC (Get/Post/Cookie). Si *magic_quotes* vale on, todas las ' (comilla sencilla), " (comilla doble), \ (barra invertida) y los NUL son automáticamente marcados con una barra invertida. Si además *magic_quotes_sybase* vale on, la comilla sencilla es marcada con otra comilla sencilla en lugar de la barra invertida.

magic_quotes_runtime boolean

Si se habilita *magic_quotes_runtime*, muchas de las funciones que devuelven datos de algún tipo de fuente externa incluyendo bases de datos y archivos de texto devolverán las comillas marcadas con una barra invertida. Si también está activo *magic_quotes_sybase*, la comilla simple es marcada con una comilla simple en lugar de la barra invertida.

magic_quotes_sybase boolean

Si *magic_quotes_sybase* está a on, la comilla simple es marcada con una comilla simple en lugar de la barra invertida cuando están habilitados *magic_quotes_gpc* o *magic_quotes_runtime*.

max_execution_time integer

Fija el tiempo máximo en segundos que se le permite usar a un script antes de ser finalizado por el intérprete. Así se evita que scripts mal escritos puedan bloquear el servidor.

memory_limit integer

Fija el tamaño máximo de memoria en bytes que se permite reclamar a un script. Así se evita que script mal escritos se coman toda la memoria disponible de un servidor.

nsapi_ext string

short_open_tag boolean

Indica si se debe permitir el formato corto (`<? ?>`) de la etiqueta de apertura del PHP. Si desea utilizar PHP en combinación con XML, deberá desactivar esta opción. Si está desactivada, deberá utilizar el formato largo de la etiqueta de apertura (`<?php ?>`).

sql.safe_mode boolean*track_errors* boolean

Si está habilitada, el último mensaje de error estará siempre presente en la variable global `$php_errormsg`.

track_vars boolean

Si está activada, la información de entrada de GET, POST y de las cookies se puede encontrar en las matrices asociativas `$HTTP_GET_VARS`, `$HTTP_POST_VARS` y `$HTTP_COOKIE_VARS` respectivamente.

upload_tmp_dir string

El directorio temporal utilizado para almacenar archivos cuando se envían al servidor. Debe tener permiso de escritura para el usuario bajo el que corra el PHP.

user_dir string

El nombre base del directorio utilizado bajo el directorio inicial de un usuario para los archivos PHP. Por ejemplo: `páginas_html`.

warn_plus_overloading boolean

Si está activada, esta opción hace que el PHP muestre un aviso cuando el operador suma (+) se utiliza en cadenas. Así es más fácil encontrar scripts que necesitan ser reescritos utilizando en su lugar el concatenador de cadenas (.).

Directivas de Configuración de Correo

SMTP string

Nombre DNS o dirección IP del servidor de SMTP que el PHP bajo Windows deberá usar para enviar correo con la función `mail()`.

sendmail_from string

La dirección del remitente ("De : ") para los correos enviados desde PHP bajo Windows.

sendmail_path string

Localización del programa **sendmail**. Generalmente `/usr/sbin/sendmail` o `/usr/lib/sendmail`. **configure** intenta localizarle este archivo lo mejor que puede y fijar un valor por defecto, pero en caso de fallo, lo puede usted fijar aquí.

Los sistemas que no usan sendmail deberán fijar esta directiva al nombre del programa alternativo que ofrezca su sistema de correo, si es que existe. Por ejemplo, los usuarios del Qmail (<http://www.qmail.org/>) pueden fijarlo normalmente a `/var/qmail/bin/sendmail`.

Directivas de Configuración de Modo Seguro

safe_mode boolean

Para activar el modo seguro del PHP. Lea el [Capítulo de seguridad](#) para más información.

safe_mode_exec_dir string

Si el PHP se utiliza en modo seguro, la función **system()** y el resto de funciones que ejecutan programas del sistema se niegan a ejecutar programas que no estén en este directorio.

Directivas de Configuración del Debugger

debugger.host string

Nombre DNS o dirección IP del servidor usado por el debugger.

debugger.port string

Número de puerto usado por el debugger.

debugger.enabled boolean

Indica si el debugger está habilitado o no.

Directivas de Carga de Extensiones

enable_dl boolean

Esta directiva sólo es útil en la versión del PHP como módulo del Apache. Puede habilitar o deshabilitar para un servidor virtual o para un directorio la carga dinámica de extensiones de PHP mediante **dl()**.

La razón principal para deshabilitar la carga dinámica es la seguridad. Con la carga dinámica es posible ignorar las restricciones **safe_mode** y **open_basedir**.

El valor por defecto es permitir la carga dinámica, excepto cuando se usa el modo seguro. En modo seguro, siempre es imposible usar **dl()**.

extension_dir string

En qué directorio debe buscar el PHP las extensiones cargables dinámicamente.

extension string

Qué extensiones dinámicas debe cargar el PHP cuando arranca.

Directivas de Configuración de MySQL

mysql.allow_persistent boolean

Si permitir o no conexiones MySQL persistentes.

mysql.default_host string

El servidor por defecto para utilizar cuando se conecte al servidor de bases de datos si no se especifica otro distinto.

mysql.default_user string

El nombre de usuario por defecto para utilizar cuando se conecta al servidor de base de datos si no se especifica otro.

mysql.default_password string

La clave por defecto para utilizar cuando se conecta al servidor de base de datos si no se especifica otro.

mysql.max_persistent integer

El número máximo de conexiones persistentes de MySQL por proceso.

mysql.max_links integer

El número máximo de conexiones de MySQL por proceso, incluyendo las persistentes.

Directivas de Configuración de mSQL

msql.allow_persistent boolean

Si se permiten o no conexiones persistentes de mSQL.

msql.max_persistent integer

El número máximo de conexiones persistentes mSQL por proceso.

msql.max_links integer

El número máximo de conexiones de mSQL por proceso, incluyendo las persistentes.

Directivas de Configuración de Postgres

pgsql.allow_persistent boolean

Si se permiten o no conexiones persistentes de Postgres.

pgsql.max_persistent integer

El número máximo de conexiones persistentes Postgres por proceso.

pgsql.max_links integer

El número máximo de conexiones de Postgres por proceso, incluyendo las persistentes.

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 necessary. 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.

Directivas de Configuración de Sybase

sybase.allow_persistent boolean

Si se permiten o no conexiones persistentes de Sybase.

sybase.max_persistent integer

El número máximo de conexiones persistentes Sybase por proceso.

sybase.max_links integer

El número máximo de conexiones de Sybase por proceso, incluyendo las persistentes.

Directivas de Configuración de Sybase-CT

sybct.allow_persistent boolean

Si se permiten o no conexiones persistentes de Sybase-CT. El valor por defecto es on.

sybct.max_persistent integer

El número máximo de conexiones persistentes Sybase-CT por proceso. El valor por defecto es -1, que significa ilimitadas.

sybct.max_links integer

El número máximo de conexiones de Sybase-CT por proceso, incluyendo las persistentes. El valor por defecto es -1, que significa ilimitadas.

sybct.min_server_severity integer

Los mensajes de servidor con gravedad mayor o igual que *sybct.min_server_severity* serán reportados como avisos. Este valor también se puede cambiar desde un script usando la función **sybase_min_server_severity()**. El valor por defecto es 10, que reporta los errores de información con gravedad o mayores.

sybct.min_client_severity integer

Los mensajes de librería de cliente con gravedad mayor o igual que *sybct.min_client_severity* serán reportados como avisos. Este valor también se puede cambiar desde un script usando la función **sybase_min_client_severity()**. El valor por defecto es 10, que desconecta los avisos.

sybct.login_timeout integer

El número máximo de segundos de espera por un intento de conexión con éxito antes de indicar un fallo. Nótese que si se ha excedido *max_execution_time* cuando finaliza la espera de un intento de conexión, el script será finalizado antes de que se pueda tomar una acción en caso de fallo. El valor por defecto es 1 minuto.

sybct.timeout integer

El número máximo de segundos de espera por una operación de consulta o select_db con éxito antes de indicar un fallo. Nótese que si se ha excedido *max_execution_time* cuando finaliza la espera de un intento de conexión, el script será finalizado antes de que se pueda tomar una acción en caso de fallo. El valor por defecto es sin límite.

sybct.hostname string

El nombre de la máquina desde la que dice estar conectado, para que se visualice con **sp_who()**. El valor por defecto es "none".

Directivas de Configuración de Informix

ifx.allow_persistent boolean

Si se permiten o no conexiones persistentes de Informix.

ifx.max_persistent integer

El número máximo de conexiones persistentes de Informix por proceso.

ifx.max_links integer

El número máximo de conexiones Informix por proceso, incluyendo las persistentes.

ifx.default_host string

El servidor por defecto al que conectarse si no se especifica uno en **ifx_connect()** o en **ifx_pconnect()**.

ifx.default_user string

El id de usuario por defecto para utilizar si no se especifica uno en **ifx_connect()** o en **ifx_pconnect()**.

ifx.default_password string

La clave por defecto para utilizar si no se especifica uno en **ifx_connect()** o en **ifx_pconnect()**.

ifx.blobinfile boolean

Fíjelo a true si desea recibir las columnas blob (objetos binarios grandes) en un archivo, y a false si las desea en memoria. Puede cambiar el ajuste en tiempo de ejecución utilizando **ifx_blobinfile_mode()**.

ifx.textasvarchar boolean

Fíjelo a true si desea recibir las columnas TEXT como cadenas normales en las instrucciones select, y a false si quiere usar parámetros de identificador de blobs. Puede cambiar el ajuste en tiempo de ejecución utilizando **ifx_textasvarchar()**.

ifx.byteasvarchar boolean

Fíjelo a true si desea devolver las columnas BYTE como cadenas normales en las instrucciones select, y a false si quiere usar parámetros de identificador de blobs. Puede cambiar el ajuste en tiempo de ejecución utilizando **ifx_byteasvarchar()**.

ifx.charasvarchar boolean

Fíjelo a true si desea suprimir los espacios a la derecha de las columnas CHAR cuando las solicita.

ifx.nullformat boolean

Fíjelo a true si desea que las columnas NULL (nulas) se devuelvan como la cadena literal "NULL", y a false si desea que se devuelvan como la cadena vacía "". Puede cambiar el ajuste en tiempo de ejecución utilizando **ifx_nullformat()**.

Directivas de Configuración de Matemática BC

bcmath.scale integer

Número de dígitos decimales para todas las funciones de bcmath.

Directivas de Configuración de Capacidades de los Navegadores

browscap string

Nombre del archivo de capacidades del navegador. Vea también **get_browser()**.

Directivas Unificadas de Configuración de ODBC

uodbc.default_db string

Fuentes de datos ODBC a utilizar si no se especifica una en **odbc_connect()** o en **odbc_pconnect()**.

uodbc.default_user string

Nombre de usuario si no se especifica uno en **odbc_connect()** o en **odbc_pconnect()**.

uodbc.default_pw string

Clave para usar si no se especifica una en **odbc_connect()** o en **odbc_pconnect()**.

uodbc.allow_persistent boolean

Si se permiten o no conexiones persistentes de ODBC.

uodbc.max_persistent integer

El número máximo de conexiones persistentes de ODBC por proceso.

uodbc.max_links integer

El número máximo de conexiones ODBC por proceso, incluyendo las persistentes.

Capítulo 4. Seguridad

PHP es un potente lenguaje y el interprete, tanto incluido en el servidor web como modulo o ejecutado como un binario CGI, puede acceder a ficheros, ejecutar comandos y abrir comunicaciones de red en el servidor. Todas estas características hacen que lo que se ejecute en el servidor web sea inseguro por defecto. PHP ha sido diseñado específicamente, para ser un lenguaje más seguro para escribir programas CGI, que Perl o C y con la correcta selección de las opciones de configuración del tiempo de compilación y ejecución se consigue la exacta combinación de libertad y seguridad que se necesita.

Ya que existen diferentes modos de utilizar PHP, existen multitud de opciones de configuración que permiten controlar su funcionamiento. Una gran selección de opciones garantiza que se pueda usar PHP para diferentes usos, pero también significa que existen combinaciones de estas opciones y configuraciones del servidor que producen instalaciones inseguras. Este capítulo explica las diferentes combinaciones de opciones de configuración y las situaciones donde pueden ser usadas de manera segura.

Binarios CGI

Posibles ataques

Usando PHP como un binario CGI es una opción para instalaciones que por cualquier causa no quieren integrar PHP como modulo en el software servidor (p.ej: Apache), o usarán PHP con diferentes clases de CGI wrappers para crear entornos chroot y setuid seguros para los scripts. Esta configuración implica generalmente instalar el binario ejecutable de PHP en el directorio cgi-bin del servidor web. El documento del CERT CA-96.11 (http://www.cert.org/advisories/CA-96.11.interpreters_in_cgi_bin_dir.html) recomienda no instalar interpretadores en cgi-bin. Aunque el binario PHP puede ser usado como interpretador independiente, PHP está diseñado para prevenir los ataques que esta configuración hace posible.

- Accediendo a ficheros del sistema: `http://my.host/cgi-bin/php?/etc/passwd`

La información introducida después del signo de interrogación (?) es transferida como argumento de la línea de comando al intérprete por el interfaz del CGI. Normalmente los interpretadores abren y ejecutan el fichero especificado como el primer argumento en la línea de comando.

Cuando se ejecuta como un CGI script, PHP rechaza interpretar los argumentos de la línea de comando.

- Accediendo cualquier documento web en el servidor: `http://my.host/cgi-bin/php/secret/doc.html`

La información con el camino (path) de la URL después del nombre del binario PHP, /secret/doc.html es usada convencionalmente para especificar el nombre del fichero que será abierto e interpretado por el programa CGI. Normalmente, algunas directivas del servidor web (Apache:Action) son usadas para redirigir peticiones de documentos como `http://my.host/secret/script.php3` al intérprete PHP. Con esta configuración, el servidor web comprueba primero los permisos de acceso al directorio /secret, y después crea la petición redirigida `http://my.host/cgi-bin/php/secret/script.php3`. Desafortunadamente, si la petición es hecha de esta forma en un principio, el servidor web no comprueba los permisos de acceso del fichero /secret/script.php3, sino solamente del fichero /cgi-bin/php. De esta manera cualquier usuario que pueda acceder /cgi-bin/php también puede acceder a cualquier documento protegido en el servidor web.

En PHP, a la hora de compilar, la opción de configuración `enable-force-cgi-redirect` y las directivas de configuración a la hora de ejecutar `doc_root` y `user_dir` pueden ser usadas para prevenir este ataque, si el árbol de documentos del servidor tiene cualquier directorio con acceso restringido. Ver más adelante la explicación de las diferentes combinaciones.

Caso 1: solamente se sirven ficheros públicos

Si tu servidor no contiene información que esté protegida con clave o acceso de control de IPs, no se necesitan estas opciones de configuración. Si tu servidor web no permite realizar redirecciones, o el servidor no tiene modo de comunicar al binario PHP que la petición es una petición segura redirigida, puedes especificar la opción

[-disable-force-cgi-redirect](#) en el script de configuracion. De todas maneras, teneis que aseguraros que vuestros scripts PHP no confíen en la manera al llamar al script, ni de forma directa `http://my.host/cgi-bin/php/dir/script.php3` o por redireccion `http://my.host/dir/script.php3`.

Redireccionamiento puede ser configurado en Apache usando las directivas AddHandler y Action (ver mas abajo).

Caso 2: usando [-enable-force-cgi-redirect](#)

Esta opcion a la hora de compilar previene que alguien llame PHP directamente con una url como la siguiente `http://my.host/cgi-bin/php/secretdir/script.php3`. PHP solamente analizara en este modo si ha pasado por una regla de redireccionamiento en el servidor.

Normalmente la redireccion en la configuracion de Apache es hecha con la siguientes directivas:

```
Action php3-script /cgi-bin/php
AddHandler php3-script .php3
```

Esta opcion ha sido solo comprobada con el servidor web Apache, y depende de Apache para fijar la variable de entorno CGI no estandar REDIRECT_STATUS en las peticiones de redireccionamiento. Si tu servidor web no soporta ningun modo para informar si una peticion es directa o redireccionada, no podeis usar esta opcion y debereis usar alguno de los otros modos de ejecucion de la version CGI documentados aqui.

Caso 3: Usando `doc_root` or `user_dir`

Incluir contenidos activos, como script y ejecutables, en el directorio de documentos del servidor web, es algunas veces considerada una practica insegura. Si por algun fallo de configuracion, los scripts no son ejecutados pero mostrados como documentos HTML, cualquiera podra conseguir codigo registrado o informacion de seguridad, como p.ej: claves de acceso. Por ello, muchos administradores prefieren utilizar otra estructura de directorios que contenga solamente los scripts, los cuales seran solamente accesibles via PHP CGI, y por ello siempre seran interpretados y no mostrados.

Habra que tener en cuenta que si el metodo que asegura que las peticiones no son redireccionadas, como hemos descrito en la seccion anterior, no esta disponible, sera necesario configurar un script `doc_root` que sea diferente del "web document root".

Podeis definir el script PHP "document root" con la directiva de configuracion [doc_root](#) en el [fichero de configuracion](#), o definir la variable de entorno `PHP_DOCUMENT_ROOT`. Si esta definida, la version CGI de PHP siempre obtendra el nombre del fichero a abrir con `doc_root` y el camino (path) utilizado en la peticion, asi podeis estar seguros que ningun script sera ejecutado fuera de este directorio (excepto para `user_dir`, ver a continuacion)

Otra opcion que se puede usar aqui es `user_dir`. Cuando `user_dir` no esta definido, lo unico que controla la apertura del fichero es `doc_root`. Si intentamos abrir una url tal como esta `http://my.host/~user/doc.php3` no se abrirá un fichero en el directorio de usuarios, en su lugar se abrirá un fichero llamado `~user/doc.php3` en el directorio `doc_root`. (si, un directorio que empieza por tilde [~]).

Si `user_dir` esta definido por ejemplo como `public_php`, una peticion tal como `http://my.host/~user/doc.php3`, abrirá un fichero llamado `doc.php3` en el directorio llamado `public_php` del directorio "home" del usuario. Si el directorio del usuario es `/home/user`, el fichero ejecutado sera `/home/user/public_php/doc.php3`.

La expansion de `user_dir` ocurre sin tener en cuenta la configuracion de `doc_root`, de este modo se puede controlar los accesos al directorio principal (document root) y al directorio de usuario separadamente.

Caso 4: Analizador PHP fuera del arbol web.

Una opcion muy segura es poner el analizador binario PHP, en algun lugar fuera del arbol de ficheros web. Por ejemplo en `/usr/local/bin`. La unica pega real de esta opcion es que habra que poner una linea similar a:

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

como primera linea en cualquier fichero que contenga codigo PHP. Tambien sera necesario asignar al fichero permisos de ejecucion. De esta manera, es tratado de la misma manera que cualquier otro CGI script escrito en Perl o sh o otro lenguaje utilizado para scripts y que utilicen el mecanismo #! para ejecutarse.

Para conseguir que PHP maneje correctamente con esta configuracion, la informacion de PATH_INFO y PATH_TRANSLATED, el analizador PHP deberia ser compilado con la opcion de configuracion [-enable-discard-path](#).

Modulo Apache

Cuando PHP es usado como modulo Apache, hereda los permisos de usuario de Apache (normalmente "nobody")

Parte II. Referencia del Lenguaje

Capítulo 5. Sintaxis básica

Saliendo de HTML

Hay cuatro formas de salir de HTML y entrar en el "modo de código PHP":

Ejemplo 5-1. Formas de salir de HTML

```
1. <? echo ("esta es la más simple, una instrucción de procesado SGML\n"); ?>
2. <?php echo("si quiere servir documentos XML, haga esto\n"); ?>
3. <script language="php">
   echo ("a algunos editores (como FrontPage) no les
         gustan las instrucciones de procesado");
</script>
4. <% echo ("Puedes también usar etiquetas tipo ASP"); %>
<%= $variable; # Esto es una forma abreviada de "<%echo .." %>
```

La primera forma sólo está disponible si se han habilitado las etiquetas cortas. Esto se puede hacer a través de la función **short_tags()**, habilitando la opción de configuración **short_open_tag** en el archivo de configuración de PHP, o compilando PHP con la opción **-enable-short-tags** en **configure**.

La cuarta manera está disponible sólo si se han habilitado las etiquetas tipo ASP usando la opción de configuración **asp_tags**.

Nota: El soporte para las etiquetas tipo ASP se añadió en 3.0.4.

La etiqueta de cierre de un bloque incluirá el carácter de nueva línea final si hay uno presente.

Separación de instrucciones

Las instrucciones se separan igual que en C o perl - terminando cada sentencia con un punto y coma.

La etiqueta de cierre (?>) también implica el fin de la sentencia, así lo siguiente es equivalente:

```
<?php
    echo "Esto es una prueba";
?>

<?php echo "Esto es una prueba" ?>
```

Comentarios

PHP soporta comentarios tipo 'C', 'C++' y shell de Unix. Por ejemplo:

```
<?php
    echo "Esto es una prueba"; // Esto es un comentario tipo c++ para una línea
    /* Esto es un comentario multilínea
       otra línea más de comentario*/
    echo "Esto es aún otra prueba";
    echo "Una Prueba Final"; # Este es un comentario tipo shell
```

```
?>
```

El tipo de comentario de "una línea" sólo comenta, en realidad, hasta el fin de la línea o el bloque actual de código PHP, lo que venga primero.

```
<h1>Esto es un <?# echo "simple";?> ejemplo.</h1>
<p>La cabecera de arriba dirá 'Esto es un ejemplo'.
```

Se debería tener cuidado para no anidar comentarios de tipo 'C', lo cual puede ocurrir cuando se comentan grandes bloques.

```
<?php
/*
    echo "Esto es una prueba"; /* Este comentario causará un problema */
*/
?>
```

Capítulo 6. Types

PHP soporta los siguientes tipos:

- [array](#)
- [números en punto flotante](#)
- [entero](#)
- [objeto](#)
- [cadena](#)

El tipo de una variable normalmente no lo indica el programador; en su lugar, lo decide PHP en tiempo de ejecución dependiendo del contexto en el que se utilice esa variable.

Si se quisiese obligar a que una variable se convierta a un tipo concreto, se podría [forzar](#) la variable o usar la función [settype\(\)](#) para ello.

Nótese que una variable se puede comportar de formas diferentes en ciertas situaciones, dependiendo de qué tipo sea en ese momento. Para más información, vea la sección [Conversión de Tipos](#).

Enteros

Los enteros se puede especificar usando una de las siguientes sintaxis:

```
$a = 1234; # número decimal
$a = -123; # un número negativo
$a = 0123; # número octal (equivalente al 83 decimal)
$a = 0x12; # número hexadecimal (equivalente al 18 decimal)
```

Números en punto flotante

Los números en punto flotante ("double") se pueden especificar utilizando cualquiera de las siguientes sintaxis:

```
$a = 1.234; $a = 1.2e3;
```

Cadenas

Las cadenas de caracteres se pueden especificar usando uno de dos tipos de delimitadores.

Si la cadena está encerrada entre dobles comillas (""), las variables que estén dentro de la cadena serán expandidas (sujetas a ciertas limitaciones de interpretación). Como en C y en Perl, el carácter de barra invertida ("\") se puede usar para especificar caracteres especiales:

Tabla 6-1. Caracteres protegidos

secuencia	significado
\n	Nueva Línea
\r	Retorno de carro
\t	Tabulación horizontal
\\	Barra invertida

secuencia	significado
\\$	Signo del dólar
\ "	Comillas dobles
\[0-7]{1,3}	la secuencia de caracteres que coincide con la expresión regular es un carácter en notación octal
\x[0-9A-Fa-f]{1,2}	la secuencia de caracteres que coincide con la expresión regular es un carácter en notación hexadecimal

Se puede proteger cualquier otro carácter, pero se producirá una advertencia en el nivel de depuración más alto.

La segunda forma de delimitar una cadena de caracteres usa el carácter de comilla simple (""). Cuando una cadena va encerrada entre comillas simples, los únicos caracteres de escape que serán comprendidos son "\\\"y "\\". Esto es por convenio, así que se pueden tener comillas simples y barras invertidas en una cadena entre comillas simples. Las variables no se expandirán dentro de una cadena entre comillas simples.

Otra forma de delimitar cadenas es usando la sintaxis de documento incrustado (">>"). Se debe proporcionar un identificador después de >>, después la cadena, y después el mismo identificador para cerrar el entrecorbillado.

Ejemplo 6-1. He aquí un ejemplo de entrecorbillado de cadenas con sintaxis de documento incrustado

```
$str = >>EOD
Ejemplo de cadena
Expandiendo múltiples líneas
usando sintaxis de documento incrustado.
EOD;
```

Nota: La sintaxis de documento incrustado fue añadida en PHP 4.

Las cadenas se pueden concatenar usando el operador '.' (punto). Nótese que el operador '+' (suma) no sirve para esto. Por favor mire [Operadores de cadena](#) para más información.

Se puede acceder a los caracteres dentro de una cadena tratándola como un array de caracteres indexado numéricamente, usando una sintaxis similar a la de C. Vea un ejemplo más abajo.

Ejemplo 6-2. Algunos ejemplos de cadenas

```
<?php
/* Asignando una cadena. */
$str = "Esto es una cadena";

/* Añadiendo a la cadena. */
$str = $str . " con algo más de texto";

/* Otra forma de añadir, incluye un carácter de nueva línea protegido. */
$str .= " Y un carácter de nueva línea al final.\n";

/* Esta cadena terminará siendo '<p>Número: 9</p>' */
$num = 9;
$str = "<p>Número: $num</p>";

/* Esta será '<p>Número: $num</p>' */
$num = 9;
$str = '<p>Número: $num</p>';

/* Obtener el primer carácter de una cadena */

```

```
$str = 'Esto es una prueba.';
$first = $str[0];

/* Obtener el último carácter de una cadena. */
$str = 'Esto es aún una prueba.';
$last = $str[strlen($str)-1];
?>
```

Conversión de cadenas

Cuando una cadena se evalúa como un valor numérico, el valor resultante y el tipo se determinan como sigue.

La cadena se evaluará como un doble si contiene cualquiera de los caracteres '.', 'e', o 'E'. En caso contrario, se evaluará como un entero.

El valor viene dado por la porción inicial de la cadena. Si la cadena comienza con datos de valor numérico, este será el valor usado. En caso contrario, el valor será 0 (cero). Los datos numéricos válidos son un signo opcional, seguido por uno o más dígitos (que opcionalmente contengan un punto decimal), seguidos por un exponente opcional. El exponente es una 'e' o una 'E' seguidos por uno o más dígitos.

Cuando la primera expresión es una cadena, el tipo de la variable dependerá de la segunda expresión.

```
$foo = 1 + "10.5";           // $foo es doble (11.5)
$foo = 1 + "-1.3e3";         // $foo es doble (-1299)
$foo = 1 + "bob-1.3e3";      // $foo es entero (1)
$foo = 1 + "bob3";           // $foo es entero (1)
$foo = 1 + "10 Cerditos";    // $foo es entero (11)
$foo = 1 + "10 Cerditos";   // $foo es entero (11)
$foo = "10.0 cerdos " + 1;   // $foo es entero (11)
$foo = "10.0 cerdos " + 1.0; // $foo es double (11)
```

Para más información sobre esta conversión, mire en la página del manual de Unix `strtod(3)`.

Si quisiera probar cualquiera de los ejemplos de esta sección, puede cortar y pegar los ejemplos e insertar la siguiente línea para ver por sí mismo lo que va ocurriendo:

```
echo "\$foo==\$foo; el tipo es " . gettype( $foo ) . "<br>\n";
```

Arrays

Los arrays actualmente actúan tanto como tablas hash (arrays asociativos) como arrays indexados (vectores).

Arrays unidimensionales

PHP soporta tanto arrays escalares como asociativos. De hecho, no hay diferencias entre los dos. Se puede crear una array usando las funciones `list()` o `array()`, o se puede asignar el valor de cada elemento del array de manera explícita.

```
$a[0] = "abc";
$a[1] = "def";
$b["foo"] = 13;
```

También se puede crear un array simplemente añadiendo valores al array. Cuando se asigna un valor a una variable array usando corchetes vacíos, el valor se añadirá al final del array.

```
$a[] = "hola"; // $a[2] == "hola"
$a[] = "mundo"; // $a[3] == "mundo"
```

Los arrays se pueden ordenar usando las funciones **asort()**, **arsort()**, **ksort()**, **rsort()**, **sort()**, **uasort()**, **usort()**, y **uksort()** dependiendo del tipo de ordenación que se desee.

Se puede contar el número de elementos de un array usando la función **count()**.

Se puede recorrer un array usando las funciones **next()** y **prev()**. Otra forma habitual de recorrer un array es usando la función **each()**.

Arrays Multidimensionales

Los arrays multidimensionales son bastante simples actualmente. Para cada dimensión del array, se puede añadir otro valor [clave] al final:

```
$a[1]      = $f;           # ejemplos de una sola dimensión
$a["foo"]  = $f;

$a[1][0]   = $f;           # bidimensional
$a["foo"][2] = $f;          # (se pueden mezclar índices numéricos y asociativos)
$a[3]["bar"] = $f;          # (se pueden mezclar índices numéricos y asociativos)

$a["foo"][4]["bar"][0] = $f;  # tetradimensional!
```

En PHP3 no es posible referirse a arrays multidimensionales directamente dentro de cadenas. Por ejemplo, lo siguiente no tendrá el resultado deseado:

```
$a[3]['bar'] = 'Bob';
echo "Esto no va a funcionar: $a[3][bar]";
```

En PHP3, lo anterior tendrá la salida `Esto no va a funcionar: Array[bar]`. De todas formas, el operador de concatenación de cadenas se puede usar para solucionar esto:

```
$a[3]['bar'] = 'Bob';
echo "Esto no va a funcionar: " . $a[3][bar];
```

En PHP4, sin embargo, todo el problema se puede circunvenir encerrando la referencia al array (dentro de la cadena) entre llaves:

```
$a[3]['bar'] = 'Bob';
echo "Esto va a funcionar: {$a[3][bar]}";
```

Se pueden "rellenar" arrays multidimensionales de muchas formas, pero la más difícil de comprender es cómo usar el comando **array()** para arrays asociativos. Estos dos trozos de código llenarán el array unidimensional de la misma manera:

```
# Ejemplo 1:
```

```
$a["color"] = "rojo";
$a["sabor"] = "dulce";
$a["forma"] = "redondeada";
$a["nombre"] = "manzana";
$a[3] = 4;

# Example 2:
$a = array(
    "color" => "rojo",
    "sabor" => "dulce",
    "forma" => "redondeada",
    "nombre" => "manzana",
    3        => 4
);

```

La función **array()** se puede anidar para arrays multidimensionales:

```
<?
$a = array(
    "manzana" => array(
        "color" => "rojo",
        "sabor" => "dulce",
        "forma" => "redondeada"
    ),
    "naranja" => array(
        "color" => "naranja",
        "sabor" => "ácido",
        "forma" => "redondeada"
    ),
    "plátano" => array(
        "color" => "amarillo",
        "sabor" => "paste-y",
        "forma" => "aplatanada"
    )
);

echo $a["manzana"]["sabor"];      # devolverá "dulce"
?>
```

Objetos

Inicialización de Objetos

Para inicializar un objeto, se usa la sentencia `new` para instanciar el objeto a una variable.

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

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

Type juggling

PHP no requiere (o soporta) la declaración explícita del tipo en la declaración de variables; el tipo de una variable se determina por el contexto en el que se usa esa variable. Esto quiere decir que si se asigna un valor de cadena a la variable `var`, `var` se convierte en una cadena. Si después se asigna un valor entero a la variable `var`, se convierte en una variable entera.

Un ejemplo de conversión de tipo automática en PHP3 es el operador suma '+'. Si cualquiera de los operandos es un doble, entonces todos los operandos se evalúan como dobles, y el resultado será un doble. En caso contrario, los operandos se interpretarán como enteros, y el resultado será también un entero. Nótese que esto NO cambia los tipos de los operandos propiamente dichos; el único cambio está en cómo se evalúan los operandos.

```
$foo = "0"; // $foo es una cadena (ASCII 48)
$foo++; // $foo es la cadena "1" (ASCII 49)
$foo += 1; // $foo ahora es un entero (2)
$foo = $foo + 1.3; // $foo ahora es un doble (3.3)
$foo = 5 + "10 Cerditos Pequeñitos"; // $foo es entero (15)
$foo = 5 + "10 Cerditos"; // $foo es entero (15)
```

Si los últimos dos ejemplos anteriores parecen confusos, vea [Conversión de cadenas](#).

Si se desea obligar a que una variable sea evaluada con un tipo concreto, mire la sección [Forzado de tipos](#). Si se desea cambiar el tipo de una variable, vea la función `settype()`.

Si quisiese probar cualquiera de los ejemplos de esta sección, puede cortar y pegar los ejemplos e insertar la siguiente línea para ver por sí mismo lo que va ocurriendo:

```
echo "\$foo==\$foo; el tipo es " . gettype( $foo ) . "<br>\n";
```

Nota: La posibilidad de una conversión automática a array no está definida actualmente.

```
$a = 1; // $a es un entero
$a[0] = "f"; // $a se convierte en un array, en el que $a[0] vale "f"
```

Aunque el ejemplo anterior puede parecer que claramente debería resultar en que `$a` se convierta en un array, el primer elemento del cual es 'f', consideremos esto:

```
$a = "1"; // $a es una cadena
$a[0] = "f"; // ¿Qué pasa con los índices de las cadenas? ¿Qué ocurre?
```

Dado que PHP soporta indexación en las cadenas vía offsets usando la misma sintaxis que la indexación de arrays, el ejemplo anterior nos conduce a un problema: ¿debería convertirse `$a` en un array cuyo primer elemento sea "f", o debería convertirse "f" en el primer carácter de la cadena `$a`?

Por esta razón, tanto en PHP 3.0.12 como en PHP 4.0b3-RC4, el resultado de esta conversión automática se considera que no está definido. Los parches se están discutiendo, de todas formas.

Forzado de tipos

El forzado de tipos en PHP funciona como en C: el nombre del tipo deseado se escribe entre paréntesis antes de la variable a la que se pretende forzar.

```
$foo = 10;      // $foo es un entero
$bar = (double) $foo;    // $bar es un doble
```

Los forzados de tipo permitidos son:

- (int), (integer) - fuerza a entero (integer)
- (real), (double), (float) - fuerza a doble (double)
- (string) - fuerza a cadena (string)
- (array) - fuerza a array (array)
- (object) - fuerza a objeto (object)

Nótese que las tabulaciones y espacios se permiten dentro de los paréntesis, así que los siguientes ejemplos son funcionalmente equivalentes:

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

Puede no ser obvio que ocurrirá cuando se fuerce entre ciertos tipos. Por ejemplo, lo siguiente debería ser tenido en cuenta.

Cuando se fuerza el cambio de un escalar o una variable de cadena a un array, la variable se convertirá en el primer elemento del array:

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

Cuando se fuerza el tipo de una variable escalar o de una cadena a un objeto, la variable se convertirá en un atributo del objeto; el nombre del atributo será 'scalar':

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


Capítulo 7. Variables

Conceptos Básicos

En PHP las variables se representan como un signo de dólar seguido por el nombre de la variable. El nombre de la variable es sensible a minúsculas y mayúsculas.

```
$var = "Bob";
$Var = "Joe";
echo "$var, $Var"; // produce la salida "Bob, Joe"
```

En PHP3, las variables siempre se asignan por valor. Esto significa que cuando se asigna una expresión a una variable, el valor íntegro de la expresión original se copia en la variable de destino. Esto quiere decir que, por ejemplo, después de asignar el valor de una variable a otra, los cambios que se efectúen a una de esas variables no afectará a la otra. Para más información sobre este tipo de asignación, vea [Expresiones](#).

PHP4 ofrece otra forma de asignar valores a las variables: *asignar por referencia*. Esto significa que la nueva variable simplemente referencia (en otras palabras, "se convierte en un alias de" o "apunta a") la variable original. Los cambios a la nueva variable afectan a la original, y viceversa. Esto también significa que no se produce una copia de valores; por tanto, la asignación ocurre más rápidamente. De cualquier forma, cualquier incremento de velocidad se notará sólo en los bucles críticos cuando se asignen grandes arrays u objetos.

Para asignar por referencia, simplemente se antepone un ampersand (&) al comienzo de la variable cuyo valor se está asignando (la variable fuente). Por ejemplo, el siguiente trozo de código produce la salida 'Mi nombre es Bob' dos veces:

```
<?php
$foo = 'Bob';           // Asigna el valor 'Bob' a $foo
$bar = &$foo;           // Referencia $foo vía $bar.
$bar = "Mi nombre es $bar"; // Modifica $bar...
echo $foo;             // $foo también se modifica.
echo $bar;
?>
```

Algo importante a tener en cuenta es que sólo las variables con nombre pueden ser asignadas por referencia.

```
<?php
$foo = 25;
$bar = &$foo;           // Esta es una asignación válida.
$bar = &(24 * 7);      // Inválida; referencia una expresión sin nombre.

function test() {
    return 25;
}

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

Variables predefinidas

PHP proporciona una gran cantidad de variables predefinidas a cualquier script que se ejecute. De todas formas, muchas de esas variables no pueden estar completamente documentadas ya que dependen de sobre qué servidor se esté ejecutando, la versión y configuración de dicho servidor, y otros factores. Algunas de estas variables no estarán disponibles cuando se ejecute PHP desde la línea de comandos.

A pesar de estos factores, aquí tenemos una lista de variables predefinidas disponibles en una instalación por defecto de PHP 3 corriendo como modulo de un Apache (<http://www.apache.org/>) 1.3.6 con su configuración también por defecto.

Para una lista de variables predefinidas (y muchas más información útil), por favor, vea (y use) **phpinfo()**.

Nota: Esta lista no es exhaustiva ni pretende serlo. Simplemente es una guía de qué tipo de variables predefinidas se puede esperar tener disponibles en un script.

Variables de Apache

Estas variables son creadas por el servidor web Apache (<http://www.apache.org/>). Si se está utilizando otro servidor web, no hay garantía de que proporcione las mismas variables; pueden faltar algunas, o proporcionar otras no listadas aquí. Dicho esto, también están presentes las variables de la especificación CGI 1.1 (<http://hoohoo.ncsa.uiuc.edu/cgi/env.html>), por lo que también se deben tener en cuenta.

Tenga en cuenta que unas pocas, como mucho, de estas variables van a estar disponibles (o simplemente tener sentido) si se ejecuta PHP desde la línea de comandos.

GATEWAY_INTERFACE

Qué revisión de la especificación CGI está usando el servidor; por ejemplo 'CGI/1.1'.

SERVER_NAME

El nombre del equipo servidor en el que se está ejecutando el script. Si el script se está ejecutando en un servidor virtual, este será el valor definido para dicho servidor virtual.

SERVER_SOFTWARE

Una cadena de identificación del servidor, que aparece en las cabeceras al responderse a las peticiones.

SERVER_PROTOCOL

Nombre y revisión del protocolo a través del que se solicitó la página; p.ej. 'HTTP/1.0';

REQUEST_METHOD

Qué método de petición se usó para acceder a la página; p.ej. 'GET', 'HEAD', 'POST', 'PUT'.

QUERY_STRING

La cadena de la petición, si la hubo, mediante la que se accedió a la página.

DOCUMENT_ROOT

El directorio raíz del documento bajo el que se ejecuta el script, tal y como está definido en el fichero de configuración del servidor.

HTTP_ACCEPT

Los contenidos de la cabecera Accept : de la petición actual, si hay alguna.

HTTP_ACCEPT_CHARSET

Los contenidos de la cabecera Accept-Charset : de la petición actual, si hay alguna. Por ejemplo: 'iso-8859-1,*;utf-8'.

HTTP_ENCODING

Los contenidos de la cabecera Accept-Encoding : de la petición actual, si la hay. Por ejemplo: 'gzip'.

HTTP_ACCEPT_LANGUAGE

Los contenidos de la cabecera Accept-Language : de la petición actual, si hay alguna. Por ejemplo: 'en'.

HTTP_CONNECTION

Los contenidos de la cabecera Connection : de la petición actual, si hay alguna. Por ejemplo: 'Keep-Alive'.

HTTP_HOST

Los contenidos de la cabecera `Host` : de la petición actual, si hay alguna.

HTTP_REFERER

La dirección de la página (si la hay) desde la que el navegador saltó a la página actual. Esto lo establece el navegador del usuario; no todos los navegadores lo hacen.

HTTP_USER_AGENT

Los contenidos de la cabecera `User-Agent` : de la petición actual, si hay alguna. Indica el navegador que se está utilizando para ver la página actual; p.ej. `Mozilla/4.5 [en] (X11; U; Linux 2.2.9 i586)`. Entre otras cosas, se puede usar este valor con `get_browser()` para adaptar la funcionalidad de la página a las posibilidades del navegador del usuario.

REMOTE_ADDR

La dirección IP desde la que el usuario está viendo la página actual.

REMOTE_PORT

El puerto que se está utilizando en la máquina del usuario para comunicarse con el servidor web.

SCRIPT_FILENAME

La vía de acceso absoluta del script que se está ejecutando.

SERVER_ADMIN

El valor que se haya dado a la directiva `SERVER_ADMIN` (en Apache) en el fichero de configuración del servidor web. Si el script se está ejecutando en un servidor virtual, será el valor definido para dicho servidor virtual.

SERVER_PORT

El puerto del equipo servidor que está usando el servidor web para la comunicación. Para configuraciones por defecto, será '80'; al usar SSL, por ejemplo, cambiará al puerto que se haya definido como seguro para HTTP.

SERVER_SIGNATURE

Una cadena que contiene la versión del servidor y el nombre del servidor virtual que es añadida a las páginas generadas por el servidor, si está característica está activa.

PATH_TRANSLATED

Vía de acceso basada en el sistema de ficheros- (no el directorio raíz del documento-) del script en cuestión, después de que el servidor haya hecho la conversión virtual-a-real.

SCRIPT_NAME

Contiene la vía de acceso del script actual. Es útil para páginas que necesitan apuntar a sí mismas.

REQUEST_URI

La URI que se dió para acceder a esta página; por ejemplo, '/index.html'.

Variables de entorno

Estas variables se importan en el espacio de nombres global de PHP desde el entorno en el que se esté ejecutando el intérprete PHP. Muchas son proporcionadas por el intérprete de comandos en el que se está ejecutando PHP, y dado que a sistemas diferentes les gusta ejecutar diferentes tipos de intérpretes de comandos, es imposible hacer una lista definitiva. Por favor, mire la documentación de su intérprete de comandos para ver una lista de las variables de entorno definidas.

Otras variables de entorno son las de CGI, que están ahí sin importar si PHP se está ejecutando como un módulo del servidor o como un intérprete CGI.

Variables de PHP

Estas variables son creadas por el propio PHP.

argv

Array de argumentos pasados al script. Cuando el script se ejecuta desde la línea de comandos, esto da un acceso, al estilo de C, a los parámetros pasados en línea de comandos. Cuando se le llama mediante el método GET, contendrá la cadena de la petición.

argc

Contiene el número de parámetros de la línea de comandos pasados al script (si se ejecuta desde la línea de comandos).

PHP_SELF

El nombre del fichero que contiene el script que se está ejecutando, relativo al directorio raíz de los documentos. Si PHP se está ejecutando como intérprete de línea de comandos, esta variable no está disponible.

HTTP_COOKIE_VARS

Un array asociativo de variables pasadas al script actual mediante cookies HTTP. Sólo está disponible si el seguimiento de variables ha sido activado mediante la directiva de configuración [track_vars](#) o la directiva `<?php_track_vars?>`.

HTTP_GET_VARS

Un array asociativo de variables pasadas al script actual mediante el método HTTP GET. Sólo está disponible si –variable tracking– ha sido activado mediante la directiva de configuración [track_vars](#) o la directiva `<?php_track_vars?>`.

HTTP_POST_VARS

Un array asociativo de variables pasadas al script actual mediante el método HTTP POST. Sólo está disponible si –variable tracking– ha sido activado mediante la directiva de configuración [track_vars](#) o la directiva `<?php_track_vars?>`.

Ambito de las variables

El ámbito de una variable es el contexto dentro del que la variable está definida. La mayor parte de las variables PHP sólo tienen un ámbito simple. Este ámbito simple también abarca los ficheros incluidos y los requeridos. Por ejemplo:

```
$a = 1;
include "b.inc";
```

Aquí, la variable \$a dentro del script incluido b.inc. De todas formas, dentro de las funciones definidas por el usuario aparece un ámbito local a la función. Cualquier variables que se use dentro de una función está, por defecto, limitada al ámbito local de la función. Por ejemplo:

```
$a = 1; /* ámbito global */

Function Test () {
    echo $a; /* referencia a una variable de ámbito local */
}

Test ();
```

Este script no producirá salida, ya que la orden echo utiliza una versión local de la variable \$a, a la que no se ha asignado ningún valor en su ámbito. Puede que usted note que hay una pequeña diferencia con el lenguaje C, en el que las variables globales están disponibles automáticamente dentro de la función a menos que sean expresamente sobreescritas por una definición local. Esto puede causar algunos problemas, ya que la gente puede cambiar variables globales inadvertidamente. En PHP, las variables globales deben ser declaradas globales dentro de la función si van a ser utilizadas dentro de dicha función. Veamos un ejemplo:

```
$a = 1;
$b = 2;

Function Sum () {
    global $a, $b;

    $b = $a + $b;
}

Sum ();
echo $b;
```

El script anterior producirá la salida "3". Al declarar \$a y \$b globales dentro de la función, todas las referencias a tales variables se referirán a la versión global. No hay límite al número de variables globales que se pueden manipular dentro de una función.

Un segundo método para acceder a las variables desde un ámbito global es usando el array \$GLOBALS propio de PHP3. El ejemplo anterior se puede reescribir así:

```
$a = 1;
$b = 2;

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

Sum ();
echo $b;
```

El array \$GLOBALS es un array asociativo con el nombre de la variable global como clave y los contenidos de dicha variable como el valor del elemento del array.

Otra característica importante del ámbito de las variables es la variable *static*. Una variable estática existe sólo en el ámbito local de la función, pero no pierde su valor cuando la ejecución del programa abandona este ámbito. Consideremos el siguiente ejemplo:

```
Function Test () {
    $a = 0;
    echo $a;
    $a++;
}
```

Esta función tiene poca utilidad ya que cada vez que es llamada asigna a \$a el valor 0 y representa un "0". La sentencia \$a++, que incrementa la variable, no sirve para nada, ya que en cuanto la función termina la variable \$a desaparece. Para hacer una función útil para contar, que no pierda la pista del valor actual del conteo, la variable \$a debe declararse como estática:

```
Function Test () {
    static $a = 0;
    echo $a;
    $a++;
}
```

Ahora, cada vez que se llame a la función Test(), se representará el valor de \$a y se incrementará.

Las variables estáticas también proporcionan una forma de manejar funciones recursivas. Una función recursiva es la que se llama a sí misma. Se debe tener cuidado al escribir una función recursiva, ya que puede ocurrir que se llame a sí misma indefinidamente. Hay que asegurarse de implementar una forma adecuada de terminar la recursión. La siguiente función cuenta recursivamente hasta 10, usando la variable estática \$count para saber cuándo parar:

```
Function Test () {
    static $count = 0;

    $count++;
    echo $count;
    if ($count < 10) {
        Test ();
    }
    $count--;
}
```

Variables variables

A veces es conveniente tener nombres de variables variables. Dicho de otro modo, son nombres de variables que se pueden establecer y usar dinámicamente. Una variable normal se establece con una sentencia como:

```
$a = "hello";
```

Una variable variable toma el valor de una variable y lo trata como el nombre de una variable. En el ejemplo anterior, *hello*, se puede usar como el nombre de una variable utilizando dos signos de dólar. p.ej.

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

En este momento se han definido y almacenado dos variables en el árbol de símbolos de PHP: \$a, que contiene "hello", y \$hello, que contiene "world". Es más, esta sentencia:

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

produce el mismo resultado que:

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

p.ej. ambas producen el resultado: *hello world*.

Para usar variables variables con arrays, hay que resolver un problema de ambigüedad. Si se escribe \$\$a[1] el intérprete necesita saber si nos referimos a utilizar \$a[1] como una variable, o si se pretendía utilizar \$\$a como variable y el índice [1] como índice de dicha variable. La sintaxis para resolver esta ambigüedad es: \${\$a[1]} para el primer caso y \${\${a}[1]} para el segundo.

Variables externas a PHP

Formularios HTML (GET y POST)

Cuando se envía un formulario a un script PHP, las variables de dicho formulario pasan a estar automáticamente disponibles en el script gracias a PHP. Por ejemplo, consideremos el siguiente formulario:

Ejemplo 7-1. Variables de formulario simples

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

Cuando es enviado, PHP creará la variable \$name, que contendrá lo que sea que se introdujo en el campo *Name*: del formulario.

PHP también maneja arrays en el contexto de variables de formularios, pero sólo en una dimensión. Se puede, por ejemplo, agrupar juntas variables relacionadas, o usar esta característica para recuperar valores de un campo select input múltiple:

Ejemplo 7-2. Variables de formulario más complejas

```
<form action="array.php" method="post">
    Name: <input type="text" name="personal[name]"><br>
    Email: <input type="text" name="personal[email]"><br>
    Beer: <br>
    <select multiple name="beer[ ]">
        <option value="warthog">Warthog
        <option value="guinness">Guinness
        <option value="stuttgarter">Stuttgarter Schwabenbräu
    </select>
    <input type="submit">
</form>
```

Si la posibilidad de PHP de track_vars está activada, ya sea mediante la opción de configuración `track_vars` o mediante la directiva `<?php_track_vars?>`, las variables enviadas con los métodos POST o GET también se encontrarán en los arrays asociativos globales `$HTTP_POST_VARS` y `$HTTP_GET_VARS`.

IMAGE SUBMIT variable names

Cuando se envía un formulario, es posible usar una imagen en vez del botón submit estándar con una etiqueta como:

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

Cuando el usuario hace click en cualquier parte de la imagen, el formulario que la acompaña se transmitirá al servidor con dos variables adicionales, `sub_x` y `sub_y`. Estas contienen las coordenadas del click del usuario dentro de la imagen. Los más experimentados puede notar que los nombres de variable enviados por el navegador contienen un guion en vez de un subrayado (guion bajo), pero PHP convierte el guion en subrayado automáticamente.

Cookies HTTP

PHP soporta cookies de HTTP de forma transparente tal y como están definidas en en las Netscape's Spec (http://www.netscape.com/newsref/std/cookie_spec.html). Las cookies son un mecanismo para almacenar datos en el navegador y así rastrear o identificar a usuarios que vuelven. Se pueden crear cookies usando la función `SetCookie()`. Las cookies son parte de la cabecera HTTP, así que se debe llamar a la función `SetCookie` antes de que se envíe cualquier salida al navegador. Es la misma restricción que para la función `Header()`. Cualquier cookie que se reciba procedente del cliente será convertida automáticamente en una variable de PHP como con los datos en los métodos GET y POST.

Si se quieren asignar múltiples valores a una sola cookie, basta con añadir `[]` al nombre de la. Por ejemplo:

```
SetCookie ("MyCookie[]", "Testing", time() + 3600);
```

Nótese que una cookie reemplazará a una cookie anterior que tuviese el mismo nombre en el navegador a menos que el camino (path) o el dominio fuesen diferentes. Así, para una aplicación de carro de la compra se podría querer mantener un contador e ir pasándolo. P.ej.

Ejemplo 7-3. SetCookie Example

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

Variables de entorno

PHP hace accesibles las variables de entorno automáticamente tratándolas como variables normales.

```
echo $HOME; /* Shows the HOME environment variable, if set. */
```

Dado que la información que llega vía mecanismos GET, POST y Cookie crean automáticamente variables de PHP, algunas veces es mejor leer variables del entorno explícitamente para asegurarse de que se está trabajando con la versión correcta. La función **getenv()** se puede usar para ello. También se puede asignar un valor a una variable de entorno con la función **putenv()**.

Puntos en los nombres de variables de entrada

Típicamente, PHP no altera los nombres de las variables cuando se pasan a un script. De todas formas, hay que notar que el punto no es un carácter válido en el nombre de una variable PHP. Por esta razón, mire esto:

```
$varname.ext; /* nombre de variable no válido */
```

Lo que el intérprete ve es el nombre de una variable \$varname, seguido por el operador de concatenación, y seguido por la prueba (es decir, una cadena sin entrecomillar que no coincide con ninguna palabra clave o reservada conocida) 'ext'. Obviamente, no se pretendía que fuese este el resultado.

Por esta razón, es importante hacer notar que PHP reemplazará automáticamente cualquier punto en los nombres de variables de entrada por guiones bajos (subrayados).

Determinando los tipos de variables

Dado que PHP determina los tipos de las variables y los convierte (generalmente) según necesita, no siempre resulta obvio de qué tipo es una variable dada en un momento concreto. PHP incluye varias funciones que descubren de qué tipo es una variable. Son **gettype()**, **is_long()**, **is_double()**, **is_string()**, **is_array()**, y **is_object()**.

Capítulo 8. Constantes

PHP define varias constantes y proporciona un mecanismo para definir más en tiempo de ejecución. Las constantes son como las variables, salvo por las dos circunstancias de que las constantes deben ser definidas usando la función **define()**, y que no pueden ser redefinidas más tarde con otro valor.

Las constantes predefinidas (siempre disponibles) son:

__FILE__

El nombre del archivo de comandos que está siendo interpretado actualmente. Si se usa dentro de un archivo que ha sido incluido o requerido, entonces se da el nombre del archivo incluido, y no el nombre del archivo padre.

__LINE__

El número de línea dentro del archivo que está siendo interpretado en la actualidad. Si se usa dentro de un archivo incluido o requerido, entonces se da la posición dentro del archivo incluido.

PHP_VERSION

La cadena que representa la versión del analizador de PHP en uso en la actualidad; e.g. '3.0.8-dev'.

PHP_OS

El nombre del sistema operativo en el cuál se ejecuta el analizador PHP; e.g. 'Linux'.

TRUE

Valor verdadero.

FALSE

Valor falso.

E_ERROR

Denota un error distinto de un error de interpretación del cual no es posible recuperarse.

E_WARNING

Denota una condición donde PHP reconoce que hay algo erróneo, pero continuará de todas formas; pueden ser capturados por el propio archivo de comandos. Un ejemplo sería una inválida regexp en **ereg()**.

E_PARSE

El interprete encontró sintaxis inválida en el archivo de comandos. La recuperación no es posible.

E_NOTICE

Ocurrió algo que pudo ser o no un error. La ejecución continúa. Los ejemplos incluyen usar una cadena sin comillas como un índice "hash", o acceder a una variable que no ha sido inicializada.

Las constantes E_* se usan típicamente con la función **error_reporting()** para configurar el nivel de informes de error.

Se pueden definir constantes adicionales usando la función **define()**.

Nótese que son constantes, no macros tipo C; con una constante sólo se pueden representar datos escalares válidos.

Ejemplo 8-1. Definiendo Constantes

```
<?php
define("CONSTANTE", "Hola mundo.");
echo CONSTANTE; // muestra "Hola mundo."
?>
```

Ejemplo 8-2. Usando __FILE__ y __LINE__

```
<?php
function report_error($file, $line, $message) {
```

```
echo "Un error ocurrió en $file en la línea $line: $message.";  
}  
  
report_error(__FILE__, __LINE__, "Algo fue mal!");  
?>
```

Capítulo 9. Expresiones

Las expresiones son la piedra angular de PHP. En PHP, casi cualquier cosa que escribes es una expresión. La forma más simple y ajustada de definir una expresión es "cualquier cosa que tiene un valor".

Las formas más básicas de expresiones son las constantes y las variables. Cuando escribes '\$a = 5', estás asignando '5' a \$a. '5', obviamente, tiene el valor 5 o, en otras palabras '5' es una expresión con el valor 5 (en este caso, '5' es una constante entera).

Después de esta asignación, esperarás que el valor de \$a sea 5 también, de manera que si escribes \$b = \$a, esperas que se comporte igual que si escribieras \$b = 5. En otras palabras, \$a es una expresión también con el valor 5. Si todo va bien, eso es exactamente lo que pasará.

Las funciones son un ejemplo algo más complejo de expresiones. Por ejemplo, considera la siguiente función:

```
function foo () {
    return 5;
}
```

Suponiendo que estés familiarizado con el concepto de funciones (si no lo estás échale un vistazo al capítulo sobre funciones), asumirás que teclear \$c = foo() es esencialmente lo mismo que escribir \$c = 5, y has acertado. Las funciones son expresiones que valen el valor que retornan. Como foo() devuelve 5, el valor de la expresión 'foo()' es 5. Normalmente las funciones no devuelven un valor fijo, sino que suele ser calculado.

Desde luego, los valores en PHP no se limitan a enteros, y lo más normal es que no lo sean. PHP soporta tres tipos escalares: enteros, punto flotante y cadenas (los tipos escalares son aquellos cuyos valores no pueden 'dividirse' en partes menores, no como los arrays, por ejemplo). PHP también soporta dos tipos compuestos (no escalares): arrays y objetos. Se puede asignar cada uno de estos tipos de valor a variables o bien retornarse de funciones, sin ningún tipo de limitación.

Hasta aquí, los usuarios de PHP/FI 2 no deberían haber notado ningún cambio. Sin embargo, PHP lleva las expresiones mucho más allá, al igual que otros lenguajes. PHP es un lenguaje orientado a expresiones, en el sentido de que casi todo es una expresión. Considera el ejemplo anterior '\$a = 5'. Es sencillo ver que hay dos valores involucrados, el valor de la constante entera '5', y el valor de \$a que está siendo actualizado también a 5. Pero la verdad es que hay un valor adicional implicado aquí, y es el valor de la propia asignación. La asignación misma se evalúa al valor asignado, en este caso 5. En la práctica, quiere decir que '\$a = 5', independientemente de lo que hace, es una expresión con el valor 5. De esta manera, escribir algo como '\$b = (\$a = 5)' es como escribir '\$a = 5; \$b = 5;' (un punto y coma marca el final de una instrucción). Como las asignaciones se evalúan de derecha a izquierda, puedes escribir también '\$b = \$a = 5'.

Otro buen ejemplo de orientación a expresiones es el pre y post incremento y decremento. Los usuarios de PHP/FI 2 y los de otros muchos lenguajes les sonará la notación variable++ y variable-. Esto son las operaciones de incremento y decremento. En PHP/FI 2, la instrucción '\$a++' no tiene valor (no es una expresión), y no puedes asignarla o usarla de ningún otro modo. PHP mejora las características del incremento/decremento haciéndolos también expresiones, como en C. En PHP, como en C, hay dos tipos de incremento - pre-incremento y post-incremento. Ambos, en esencia, incrementan la variable y el efecto en la variable es idéntico. La diferencia radica en el valor de la propia expresión incremento. El preincremento, escrito '++\$variable', se evalúa al valor incrementado (PHP incrementa la variable antes de leer su valor, de ahí el nombre 'preincremento'). El postincremento, escrito '\$variable++', se evalúa al valor original de \$variable antes de realizar el incremento (PHP incrementa la variable después de leer su valor, de ahí el nombre 'postincremento').

Un tipo muy corriente de expresiones son las expresiones de comparación. Estas expresiones se evalúan a 0 o 1, significando FALSO (FALSE) o CIERTO (TRUE), respectivamente. PHP soporta > (mayor que), >= (mayor o igual que), == (igual que), != (distinto), < (menor que) y <= (menor o igual que). Estas expresiones se usan frecuentemente dentro de la ejecución condicional como la instrucción if.

El último tipo de expresiones que trataremos, es la combinación operador-asignación. Ya sabes que si quieres incrementar \$a en 1, basta con escribir '\$a++' o '\$a += 1'. Pero qué pasa si quieres añadir más de 1, por ejemplo 3? Podrías escribir '\$a += 3' múltiples veces, pero no es una forma de hacerlo ni eficiente ni cómoda. Una práctica mucho más corriente es escribir '\$a += 3'. '\$a += 3' se evalúa al valor de \$a más 3, y se asigna de nuevo a \$a, lo que resulta en incrementar \$a en 3. En PHP, como en otros lenguajes como C, puedes escribir esto de una forma más concisa, que con el tiempo será más clara y también fácil de entender. Añadir 3 al valor actual de \$a se puede escribir como '\$a += 3'. Esto quiere decir exactamente "toma el valor de \$a, sumale 3, y asignalo otra vez a \$a". Además de ser más corto y claro, también resulta en una ejecución más rápida. El valor de '\$a += 3', como el valor de una asignación normal y corriente, es el valor asignado. Ten

en cuenta que NO es 3, sino el valor combinado de \$a más 3 (ése es el valor asignado a \$a). Cualquier operación binaria puede ser usada en forma de operador-asignación, por ejemplo '\$a -= 5' (restar 5 del valor de \$a), '\$b *= 7' (multiplicar el valor de \$b por 5), etc.

Hay otra expresión que puede parecer extraña si no la has visto en otros lenguajes, el operador condicional ternario:

```
$first ? $second : $third
```

Si el valor de la primera subexpresión es verdadero (distinto de cero), entonces se evalúa la segunda subexpresión, si no, se evalúa la tercera y ése es el valor.

El siguiente ejemplo te ayudará a comprender un poco mejor el pre y post incremento y las expresiones en general:

```
function double($i) {
    return $i*2;
}
$b = $a = 5;          /* asignar el valor cinco a las variables $a y $b */
$c = $a++;           /* postincremento, asignar el valor original de $a (5) a $c */
$d = $d = ++$b;      /* preincremento, asignar el valor incrementado de $b (6) a
                        $d y a $e */
                        /* en este punto, tanto $d como $e son iguales a 6 */

$f = double($d++);   /* asignar el doble del valor de $d antes
                        del incremento, 2*6 = 12 a $f */
$g = double(++$e);   /* asignar el doble del valor de $e después
                        del incremento, 2*7 = 14 a $g */
$h = $g += 10;        /* primero, $g es incrementado en 10 y termina valiendo 24.
                        después el valor de la asignación (24) se asigna a $h,
                        y $h también acaba valiendo 24. */
```

Al principio del capítulo hemos dicho que describiríamos los distintos tipos de instrucciones y, como prometimos, las expresiones pueden ser instrucciones. Sin embargo, no todas las expresiones son instrucciones. En este caso, una instrucción tiene la forma 'expr';, es decir, una expresión seguida de un punto y coma. En '\$b=\$a=5;', \$a=5 es una expresión válida, pero no es una instrucción en sí misma. Por otro lado '\$b=\$a=5;' sí es una instrucción válida.

Una última cosa que vale la pena mencionar, es el valor booleano de las expresiones. En muchas ocasiones, principalmente en condicionales y bucles, no estás interesado en el valor exacto de la expresión, sino únicamente si es CIERTA (TRUE) o FALSA (FALSE) (PHP no tiene un tipo booleano específico). El valor de verdad de las expresiones en PHP se calcula de forma similar a perl. Cualquier valor numérico distinto de cero es CIERTO (TRUE), cero es FALSO (FALSE). Fíjate en que los valores negativos son distinto de cero y considerados CIERTO (TRUE)! La cadena vacía y la cadena "0" son FALSO (FALSE); todas las demás cadenas son TRUE. Con los tipos no escalares (arrays y objetos) - si el valor no contiene elementos se considera FALSO (FALSE), en caso contrario se considera CIERTO (TRUE).

PHP te brinda una completa y potente implementación de expresiones, y documentarla enteramente está más allá del objetivo de este manual. Los ejemplos anteriores, deberían darte una buena idea de qué son las expresiones y cómo construir expresiones útiles. A lo largo del resto del manual, escribiremos *expr* para indicar una expresión PHP válida.

Capítulo 10. Operadores

Operadores Aritméticos

¿Recuerdas la aritmética básica del colegio? Pues estos operadores funcionan exactamente igual.

Tabla 10-1. Operadores Aritméticos

ejemplo	nombre	resultado
\$a + \$b	Adición	Suma de \$a y \$b.
\$a - \$b	Substracción	Diferencia entre \$a y \$b.
\$a * \$b	Multiplicación	Producto de \$a and \$b.
\$a / \$b	División	Cociente de \$a entre \$b.
\$a % \$b	Módulo	Resto de \$a dividido entre \$b.

Operadores de Asignación

El operador básico de asignación es "`=`". A primera vista podrías pensar que es el operador de comparación "igual que". Pero no. Realmente significa que el operando de la izquierda toma el valor de la expresión a la derecha, (esto es, "toma el valor de").

El valor de una expresión de asignación es el propio valor asignado. Esto es, el valor de "`$a = 3`" es 3. Esto permite hacer cosas curiosas como

```
$a = ($b = 4) + 5; // ahora $a es igual a 9, y $b vale 4.
```

Además del operador básico de asignación, existen los "operadores combinados" para todas las operaciones aritméticas y de cadenas que sean binarias. Este operador combinado te permite, de una sola vez, usar una variable en una expresión y luego establecer el valor de esa variable al resultado de la expresión. Por ejemplo:

```
$a = 3;
$a += 5; // establece $a a 8, como si hubiésemos escrito: $a = $a + 5;
$b = "Hola ";
$b .= "Ahí!"; // establece $b a "Hola Ahí!", igual que si hiciésemos $b = $b . "Ahí!";
```

Fíjate en que la asignación realiza una nueva copia de la variable original (asignación por valor), por lo que cambios a la variable original no afectan a la copia. Esto puede tener interés si necesitas copiar algo como un array con muchos elementos dentro de un bucle que se repita muchas veces (cada vez se realizará una nueva copia del array). PHP4 soporta asignación por referencia, usando la sintaxis `$var = &$othervar;`, pero esto no es posible en PHP3. 'Asignación por referencia' quiere decir que ambas variables acabarán apuntando al mismo dato y que nada es realmente copiado.

Operadores Bit a bit

Los operadores bit a bit te permiten activar o desactivar bits individuales de un entero.

Tabla 10-2. Operadores Bit a bit

ejemplo	nombre	resultado
<code>\$a & \$b</code>	Y	Se activan los bits que están activos tanto en \$a como \$b.
<code>\$a \$b</code>	O	Se activan los bits que están activos en \$a o que lo están en \$b.

ejemplo	nombre	resultado
<code>\$a ^ \$b</code>	Xor ("o exclusiva")	Se activan los bits que están activos en \$a o en \$b pero no en ambos a la vez.
<code>~ \$a</code>	No	Se activan los bits que no están activos en \$a.
<code>\$a << \$b</code>	Desplazamiento a la izquierda	Desplaza los bits de \$a, \$b posiciones hacia la izquierda (por aritmética binaria, cada posición desplazada equivale a multiplicar por dos el valor de \$a)
<code>\$a >> \$b</code>	Desplazamiento a la derecha	Desplaza los bits de \$a, \$b posiciones hacia la derecha (por aritmética binaria, cada posición desplazada equivale a dividir entre dos el valor de \$a)

Operadores de Comparación

Los operadores de comparación, como su nombre indica, permiten comparar dos valores.

Tabla 10-3. Operadores de Comparación

ejemplo	nombre	resultado
<code>\$a == \$b</code>	Igualdad	Cierto si \$a es igual a \$b.
<code>\$a === \$b</code>	Identidad	Cierto si \$a es igual a \$b y si son del mismo tipo (sólo PHP4)
<code>\$a != \$b</code>	Desigualdad	Cierto si \$a no es igual a \$b.
<code>\$a < \$b</code>	Menor que	Cierto si \$a es estrictamente menor que \$b.
<code>\$a > \$b</code>	Mayor que	Cierto si \$a es estrictamente mayor que \$b.
<code>\$a <= \$b</code>	Menor o igual que	Cierto si \$a es menor o igual que \$b.
<code>\$a >= \$b</code>	Mayor o igual que	Cierto si \$a es mayor o igual que \$b.

Otro operador condicional es el operador "?:"(o ternario), que funciona como en C y otros muchos lenguajes.

```
(expr1) ? (expr2) : (expr3);
```

La expresión toma el valor `expr2` si `expr1` se evalúa a cierto, y `expr3` si `expr1` se evalúa a falso.

Operador de ejecución

PHP soporta un operador de ejecución: el apóstrofe invertido (''). ¡Fíjate que no son apostrofes normales! PHP intentará ejecutar la instrucción contenida dentro de los apóstrofes invertidos como si fuera un comando del shell; y su salida devuelta como el valor de esta expresión (i.e., no tiene por qué ser simplemente volcada como salida; puede asignarse a una variable).

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

Ver también `system()`, `passthru()`, `exec()`, `popen()`, y `escapeshellcmd()`.

Operadores de Incremento/decremento

PHP soporta los operadores de predecremento y post incremento al estilo de C.

Tabla 10-4. Operadores de Incremento/decremeno

ejemplo	nombre	efecto
<code>++\$a</code>	Preincremento	Incrementa \$a en uno y después devuelve \$a.
<code>\$a++</code>	Postincremento	Devuelve \$a y después incrementa \$a en uno.
<code>-\$a</code>	Predecremento	Decrementa \$a en uno y después devuelve \$a.
<code>\$a-</code>	Postdecremento	Devuelve \$a y después decrementa \$a en uno.

He aquí un listado de ejemplo:

```
<?php
echo "<h3>Postincremento</h3>";
$a = 5;
echo "Debería ser 5: " . $a++ . "<br>\n";
echo "Debería ser 6: " . $a . "<br>\n";

echo "<h3>Preincremento</h3>";
$a = 5;
echo "Debería ser 6: " . ++$a . "<br>\n";
echo "Debería ser 6: " . $a . "<br>\n";

echo "<h3>Postdecremento</h3>";
$a = 5;
echo "Debería ser 5: " . $a- . "<br>\n";
echo "Debería ser 4: " . $a . "<br>\n";

echo "<h3>Predecremento</h3>";
$a = 5;
echo "Debería ser 4: " . -$a . "<br>\n";
echo "Debería ser 4: " . $a . "<br>\n";
?>
```

Operadores Lógicos

Tabla 10-5. Operadores Lógicos

ejemplo	nombre	resultado
<code>\$a and \$b</code>	Y	Cierto si tanto \$a como \$b son ciertos.
<code>\$a or \$b</code>	O	Cierto si \$a o \$b son ciertos.
<code>\$a xor \$b</code>	O exclusiva	Cierto si \$a es cierto o \$b es cierto, pero no ambos a la vez.
<code>! \$a</code>	Negación	Cierto si \$a no es cierto.
<code>\$a && \$b</code>	Y	Cierto si tanto \$a como \$b son ciertos.
<code>\$a \$b</code>	O	Cierto si \$a o \$b son ciertos.

La razón de las dos variaciones de "y" y "o" es que operan con distinta precedencia (ver [Precedencia de Operadores](#).)

Precedencia de Operadores

La precedencia de operadores especifica cómo se agrupan las expresiones. Por ejemplo, en la expresión `1 + 5 * 3`, la respuesta es 16 y no 18 porque el operador de multiplicación ("*") tiene una mayor precedencia que el de adición ("+").

La siguiente tabla lista la precedencia de operadores, indicándose primero los de menor precedencia.

Tabla 10-6. Precedencia de Operadores

Asociatividad	Operadores
izquierda	,
izquierda	or
izquierda	xor
izquierda	and
derecha	print
izquierda	= += -= *= /= .= %= &= = ^= ~= <<= >>=
izquierda	? :
izquierda	
izquierda	&&
izquierda	
izquierda	^
izquierda	&
no asociativo	== != ===
no asociativo	< <= > >=
izquierda	<< >>
izquierda	+ - .
izquierda	* / %
derecha	! ~ ++ - (int) (double) (string) (array) (object) @
derecha	[
no asociativo	new

Operadores de Cadenas

Hay dos operadores de cadenas. El primero es el operador de concatenación ('.'), que devuelve el resultado de concatenar sus operandos izquierdo y derecho. El segundo es el operador de concatenación y asignación ('.='). Consulta [Operadores de Asignación](#) para más información.

```
$a = "Hola ";
$b = $a . "Mundo!"; // ahora $b contiene "Hola Mundo!"

$a = "Hola ";
$a .= "Mundo!"; // ahora $a contiene "Hola Mundo!"
```

Capítulo 11. Estructuras de Control

Todo archivo de comandos PHP se compone de una serie de sentencias. Una sentencia puede ser una asignación, una llamada a función, un bucle, una sentencia condicional e incluso una sentencia que no haga nada (una sentencia vacía). Las sentencias normalmente acaban con punto y coma. Además, las sentencias se pueden agrupar en grupos de sentencias encapsulando un grupo de sentencias con llaves. Un grupo de sentencias es también una sentencia. En este capítulo se describen los diferentes tipos de sentencias.

if

La construcción `if` es una de las más importantes características de muchos lenguajes, incluido PHP. Permite la ejecución condicional de fragmentos de código. PHP caracteriza una estructura `if` que es similar a la de C:

```
if (expr)
    sentencia
```

Como se describe en la sección sobre expresiones, `expr` se evalúa a su valor condicional. Si `expr` se evalúa como TRUE, PHP ejecutará la sentencia, y si se evalúa como FALSE - la ignorará.

El siguiente ejemplo mostraría `a` es mayor que `b` si `$a` fuera mayor que `$b`:

```
if ($a > $b)
    print "a es mayor que b";
```

A menudo, se desea tener más de una sentencia ejecutada de forma condicional. Por supuesto, no hay necesidad de encerrar cada sentencia con una cláusula `if`. En vez de eso, se pueden agrupar varias sentencias en un grupo de sentencias. Por ejemplo, este código mostraría `a` es mayor que `b` si `$a` fuera mayor que `$b`, y entonces asignaría el valor de `$a` a `$b`:

```
if ($a > $b) {
    print "a es mayor que b";
    $b = $a;
}
```

Las sentencias `if` se pueden anidar indefinidamente dentro de otras sentencias `if`, lo cual proporciona una flexibilidad completa para ejecuciones condicionales en las diferentes partes de tu programa.

else

A menudo queremos ejecutar una sentencia si se cumple una cierta condición, y una sentencia distinta si la condición no se cumple. Esto es para lo que sirve `else`. `else` extiende una sentencia `if` para ejecutar una sentencia en caso de que la expresión en la sentencia `if` se evalúe como FALSE. Por ejemplo, el siguiente código mostraría `a` es mayor que `b` si `$a` fuera mayor que `$b`, y `a` NO es mayor que `b` en cualquier otro caso:

```
if ($a > $b) {
    print "a es mayor que b";
} else {
    print "a NO es mayor que b";
}
```

La sentencia `else` se ejecuta solamente si la expresión `if` se evalúa como FALSE, y si hubiera alguna expresión `elseif` - sólo si se evaluaron también a FALSE (Ver `elseif`).

elseif

`elseif`, como su nombre sugiere, es una combinación de `if` y `else`. Como `else`, extiende una sentencia `if` para ejecutar una sentencia diferente en caso de que la expresión `if` original se evalúe como `FALSE`. No obstante, a diferencia de `else`, ejecutará esa expresión alternativa solamente si la expresión condicional `elseif` se evalúa como `TRUE`. Por ejemplo, el siguiente código mostraría `a` es mayor que `b`, `a` es igual a `b` o `a` es menor que `b`:

```
if ($a > $b) {
    print "a es mayor que b";
} elseif ($a == $b) {
    print "a es igual que b";
} else {
    print "a es menor que b";
}
```

Puede haber varios `elseifs` dentro de la misma sentencia `if`. La primera expresión `elseif` (si hay alguna) que se evalúe como `true` se ejecutaría. En PHP, también se puede escribir 'else if' (con dos palabras) y el comportamiento sería idéntico al de un 'elseif' (una sola palabra). El significado sintáctico es ligeramente distinto (si estas familiarizado con C, es el mismo comportamiento) pero la línea básica es que ambos resultarían tener exactamente el mismo comportamiento.

La sentencia `elseif` se ejecuta sólo si la expresión `if` precedente y cualquier expresión `elseif` precedente se evalúan como `FALSE`, y la expresión `elseif` actual se evalúa como `TRUE`.

Sintaxis Alternativa de Estructuras de Control

PHP ofrece una sintaxis alternativa para alguna de sus estructuras de control; a saber, `if`, `while`, `for`, y `switch`. En cada caso, la forma básica de la sintaxis alternativa es cambiar abrir-llave por dos puntos (`:`) y cerrar-llave por `endif;`, `endwhile;`, `endfor;`, or `endswitch;`, respectivamente.

```
<?php if ($a==5): ?>
A es igual a 5
<?php endif; ?>
```

En el ejemplo de arriba, el bloque HTML "A = 5" se anida dentro de una sentencia `if` escrita en la sintaxis alternativa. El bloque HTML se mostraría solamente si `$a` fuera igual a 5.

La sintaxis alternativa se aplica a `else` y también a `elseif`. La siguiente es una estructura `if` con `elseif` y `else` en el formato alternativo:

```
if ($a == 5):
    print "a es igual a 5";
    print "...";
elseif ($a == 6):
    print "a es igual a 6";
    print "!!!";
else:
    print "a no es ni 5 ni 6";
endif;
```

Mirar también [while](#), [for](#), e [if](#) para más ejemplos.

while

Los bucles `while` son los tipos de bucle más simples en PHP. Se comportan como su contrapartida en C. La forma básica de una sentencia `while` es:

```
while (expr) sentencia
```

El significado de una sentencia `while` es simple. Le dice a PHP que ejecute la(s) sentencia(s) anidada(s) repetidamente, mientras la expresión `while` se evalúe como `TRUE`. El valor de la expresión es comprobado cada vez al principio del bucle, así que incluso si este valor cambia durante la ejecución de la(s) sentencia(s) anidada(s), la ejecución no parará hasta el fin de la iteración (cada vez que PHP ejecuta las sentencias en el bucle es una iteración). A veces, si la expresión `while` se evalúa como `FALSE` desde el principio de todo, la(s) sentencia(s) anidada(s) no se ejecutarán ni siquiera una vez.

Como con la sentencia `if`, se pueden agrupar multiples sentencias dentro del mismo bucle `while` encerrando un grupo de sentencias con llaves, o usando la sintaxis alternativa:

```
while (expr): sentencia ... endwhile;
```

Los siguientes ejemplos son idénticos, y ambos imprimen números del 1 al 10:

```
/* ejemplo 1 */

$i = 1;
while ($i <= 10) {
    print $i++; /* el valor impreso sería
                   $i antes del incremento
                   (post-incremento) */
}

/* ejemplo 2 */

$i = 1;
while ($i <= 10):
    print $i;
    $i++;
endwhile;
```

do..while

Los bucles `do..while` son muy similares a los bucles `while`, excepto que las condiciones se comprueban al final de cada iteración en vez de al principio. La principal diferencia frente a los bucles regulares `while` es que se garantiza la ejecución de la primera iteración de un bucle `do..while` (la condición se comprueba sólo al final de la iteración), mientras que puede no ser necesariamente ejecutada con un bucle `while` regular (la condición se comprueba al principio de cada iteración, si esta se evalúa como `FALSE` desde el principio la ejecución del bucle finalizará inmediatamente).

Hay una sola sintaxis para los bucles `do..while`:

```
$i = 0;
do {
    print $i;
} while ($i>0);
```

El bucle de arriba se ejecutaría exactamente una sola vez, después de la primera iteración, cuando la condición se comprueba, se evalúa como FALSE (\$i no es más grande que 0) y la ejecución del bucle finaliza.

Los usuarios avanzados de C pueden estar familiarizados con un uso distinto del bucle `do..while`, para permitir parar la ejecución en medio de los bloques de código, encapsulandolos con `do..while(0)`, y usando la sentencia `break`. El siguiente fragmento de código demuestra esto:

```
do {
    if ($i < 5) {
        print "i no es lo suficientemente grande";
        break;
    }
    $i *= $factor;
    if ($i < $minimum_limit) {
        break;
    }
    print "i es correcto";
    ...procesa i...
} while(0);
```

No se preocupe si no entiende esto completamente o en absoluto. Se pueden codificar archivos de comandos e incluso archivos de comandos potentes sin usar esta 'propiedad'.

for

Los bucles `for` son los bucles más complejos en PHP. Se comportan como su contrapartida en C. La sintaxis de un bucle `for` es:

```
for (expr1; expr2; expr3) sentencia
```

La primera expresión (`expr1`) se evalúa (ejecuta) incondicionalmente una vez al principio del bucle.

Al comienzo de cada iteración, se evalúa `expr2`. Si se evalúa como TRUE, el bucle continúa y las sentencias anidadas se ejecutan. Si se evalúa como FALSE, la ejecución del bucle finaliza.

Al final de cada iteración, se evalúa (ejecuta) `expr3`.

Cada una de las expresiones puede estar vacía. Que `expr2` esté vacía significa que el bucle debería correr indefinidamente (PHP implicitamente lo considera como TRUE, al igual que C). Esto puede que no sea tan inútil como se podría pensar, puesto que a menudo se quiere salir de un bucle usando una sentencia `break` condicional en vez de usar la condición de `for`.

Considera los siguientes ejemplos. Todos ellos muestran números del 1 al 10:

```
/* ejemplo 1 */

for ($i = 1; $i <= 10; $i++) {
    print $i;
}

/* ejemplo 2 */

for ($i = 1; ; $i++) {
    if ($i > 10) {
        break;
    }
    print $i;
```

```

}

/* ejemplo 3 */

$i = 1;
for (;;) {
    if ($i > 10) {
        break;
    }
    print $i;
    $i++;
}
/* ejemplo 4 */

for ($i = 1; $i <= 10; print $i, $i++) ;

```

Por supuesto, el primer ejemplo parece ser el mas elegante (o quizás el cuarto), pero uno puede descubrir que ser capaz de usar expresiones vacías en bucles `for` resulta útil en muchas ocasiones.

PHP también soporta la "sintaxis de dos puntos" alternativa para bucles `for`.

```
for (expr1; expr2; expr3): sentencia; ...; endfor;
```

Otros lenguajes poseen una sentencia `foreach` para traducir un array o una tabla hash. PHP3 no posee tal construcción; PHP4 sí (ver [foreach](#)). En PHP3, se puede combinar `while` con las funciones `list()` y `each()` para conseguir el mismo efecto. Mirar la documentación de estas funciones para ver un ejemplo.

foreach

PHP4 (PHP3 no) incluye una construcción `foreach`, tal como perl y algunos otros lenguajes. Esto simplemente da un modo fácil de iterar sobre arrays. Hay dos sintaxis; la segunda es una extensión menor, pero útil de la primera:

```
foreach(expresion_array as $value) sentencia
foreach(expresion_array as $key => $value) sentencia
```

La primera forma recorre el array dado por `expresion_array`. En cada iteración, el valor del elemento actual se asigna a `$value` y el puntero interno del array se avanza en una unidad (así en el siguiente paso, se estará mirando el elemento siguiente).

La segunda manera hace lo mismo, salvo que la clave del elemento actual será asignada a la variable `$key` en cada iteración.

Nota: Cuando `foreach` comienza su primera ejecución, el puntero interno a la lista (array) se reinicia automáticamente al primer elemento del array. Esto significa que no se necesita llamar a `reset()` antes de un bucle `foreach`.

Nota: Hay que tener en cuenta que `foreach` con una copia de la lista (array) especificada y no la lista en si, por ello el puntero de la lista no es modificado como en la construcción `each`.

Puede haber observado que las siguientes son funcionalidades idénticas:

```
reset( $arr );
while( list( , $value ) = each( $arr ) ) {
    echo "Valor: $value<br>\n";
}

foreach( $arr as $value ) {
    echo "Valor: $value<br>\n";
}
```

Las siguientes también son funcionalidades idénticas:

```
reset( $arr );
while( list( $key, $value ) = each( $arr ) ) {
    echo "Key: $key; Valor: $value<br>\n";
}

foreach( $arr as $key => $value ) {
    echo "Key: $key; Valor: $value<br>\n";
}
```

Algunos ejemplos más para demostrar su uso:

```
/* foreach ejemplo 1: sólo valor*/
$a = array(1, 2, 3, 17);

foreach($a as $v) {
    print "Valor actual de \$a: $v.\n";
}

/* foreach ejemplo 2: valor (con clave impresa para ilustrar) */
$a = array(1, 2, 3, 17);

$i = 0; /* sólo para propósitos demostrativos */

foreach($a as $v) {
    print "\$a[$i] => $k.\n";
}

/* foreach ejemplo 3: clave y valor */
$a = array(
    "uno" => 1,
    "dos" => 2,
    "tres" => 3,
    "diecisiete" => 17
);

foreach($a as $k => $v) {
    print "\$a[$k] => $v.\n";
}
```

break

`break` escapa de la estructuras de control iterante (bucle) actuales `for`, `while`, o `switch`.

`break` acepta un parámetro opcional, el cual determina cuantas estructuras de control hay que escapar.

```
$arr = array ('one', 'two', 'three', 'four', 'stop', 'five');
while (list ($, $val) = each ($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` se usa dentro de la estructura del bucle para saltar el resto de la iteración actual del bucle y continuar la ejecución al comienzo de la siguiente iteración.

`continue` acepta un parámetro opcional, el cual determina cuantos niveles (bluces) hay que saltar antes de continuar con la ejecución.

```
while (list($key,$value) = each($arr)) {
    if ($key % 2) { // salta los miembros impares
        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

La sentencia `switch` es similar a una serie de sentencias `IF` en la misma expresión. En muchas ocasiones, se quiere comparar la misma variable (o expresión) con muchos valores diferentes, y ejecutar una parte de código distinta dependiendo de a qué valor es igual. Para ello sirve la sentencia `switch`.

Los siguientes dos ejemplos son dos modos distintos de escribir la misma cosa, uno usa una serie de sentencias `if`, y el otro usa la sentencia `switch`:

```
if ($i == 0) {
    print "i es igual a 0";
}
if ($i == 1) {
    print "i es igual a 1";
}
if ($i == 2) {
    print "i es igual a 2";
}

switch ($i) {
    case 0:
        print "i es igual a 0";
        break;
    case 1:
        print "i es igual a 1";
        break;
    case 2:
        print "i es igual a 2";
        break;
}
```

Es importante entender cómo se ejecuta la sentencia `switch` para evitar errores. La sentencia `switch` ejecuta línea por línea (realmente, sentencia a sentencia). Al comienzo, no se ejecuta código. Sólo cuando se encuentra una sentencia `case` con un valor que coincide con el valor de la expresión `switch` PHP comienza a ejecutar las sentencias. PHP continúa ejecutando las sentencias hasta el final del bloque `switch`, o la primera vez que vea una sentencia `break`. Si no se escribe una sentencia `break` al final de una lista de sentencias `case`, PHP seguirá ejecutando las sentencias del siguiente `case`. Por ejemplo:

```
switch ($i) {
    case 0:
        print "i es igual a 0";
    case 1:
        print "i es igual a 1";
    case 2:
        print "i es igual a 2";
}
```

Aquí, si `$i` es igual a 0, ¡PHP ejecutaría todas las sentencias `print`! Si `$i` es igual a 1, PHP ejecutaría las últimas dos sentencias `print` y sólo si `$i` es igual a 2, se obtendría la conducta 'esperada' y solamente se mostraría 'i es igual a 2'. Así, es importante no olvidar las sentencias `break` (incluso aunque pueda querer evitar escribirlas intencionadamente en ciertas circunstancias).

En una sentencia `switch`, la condición se evalúa sólo una vez y el resultado se compara a cada sentencia `case`. En una sentencia `elseif`, la condición se evalúa otra vez. Si tu condición es más complicada que una comparación simple y/o está en un bucle estrecho, un `switch` puede ser más rápido.

La lista de sentencias de un case puede también estar vacía, lo cual simplemente pasa el control a la lista de sentencias del siguiente case.

```
switch ($i) {
    case 0:
    case 1:
    case 2:
        print "i es menor que 3, pero no negativo";
        break;
    case 3:
        print "i es 3";
}
```

Un case especial es el default case. Este case coincide con todo lo que no coincidan los otros case. Por ejemplo:

```
switch ($i) {
    case 0:
        print "i es igual a 0";
        break;
    case 1:
        print "i es igual a 1";
        break;
    case 2:
        print "i es igual a 2";
        break;
    default:
        print "i no es igual a 0, 1 o 2";
}
```

La expresión case puede ser cualquier expresión que se evalúe a un tipo simple, es decir, números enteros o de punto flotante y cadenas de texto. No se pueden usar aquí ni arrays ni objetos a menos que se conviertan a un tipo simple.

La sintaxis alternativa para las estructuras de control está también soportada con switch. Para más información, ver [Sintaxis alternativa para estructuras de control](#).

```
switch ($i):
    case 0:
        print "i es igual 0";
        break;
    case 1:
        print "i es igual a 1";
        break;
    case 2:
        print "i es igual a 2";
        break;
    default:
        print "i no es igual a 0, 1 o 2";
endswitch;
```

require()

La sentencia require() se sustituye a sí misma con el archivo especificado, tal y como funciona la directiva #include de C.

Un punto importante sobre su funcionamiento es que cuando un archivo se incluye con **include()** o se requiere con **require()**, el intérprete sale del modo PHP y entra en modo HTML al principio del archivo referenciado, y vuelve de nuevo al modo PHP al final. Por esta razón, cualquier código dentro del archivo referenciado que debiera ser ejecutado como código PHP debe ser encerrado dentro de [etiquetas válidas de comienzo y fin de PHP](#).

require() no es en realidad una función de PHP; es más una construcción del lenguaje. Está sujeta a algunas reglas distintas de las de funciones. Por ejemplo, **require()** no está sujeto a ninguna estructura de control contenida. Por otro lado, no devuelve ningún valor; intentar leer un valor de retorno de una llamada a un **require()** resulta en un error del intérprete.

A diferencia de **include()**, **require()** siempre leerá el archivo referenciado, *incluso si la línea en que está no se ejecuta nunca*. Si se quiere incluir condicionalmente un archivo, se usa **include()**. La sentencia conditional no afecta a **require()**. No obstante, si la línea en la cual aparece el **require()** no se ejecuta, tampoco se ejecutará el código del archivo referenciado.

De forma similar, las estructuras de bucle no afectan la conducta de **require()**. Aunque el código contenido en el archivo referenciado está todavía sujeto al bucle, el propio **require()** sólo ocurre una vez.

Esto significa que no se puede poner una sentencia **require()** dentro de una estructura de bucle y esperar que incluya el contenido de un archivo distinto en cada iteración. Para hacer esto, usa una sentencia **include()**.

```
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. */
```

En PHP3, es posible ejecutar una sentencia **return** dentro de un archivo referenciado con **require()**, en tanto en cuanto esa sentencia aparezca en el ámbito global del archivo requerido (**require()**). No puede aparecer dentro de ningún bloque (lo que significa dentro de llaves({})). En PHP4, no obstante, esta capacidad ha sido desestimada. Si se necesita esta funcionalidad, véase **include()**.

Ver tambien **include()**, **require_once()**, **include_once()**, **readfile()**, y **virtual()**.

include()

La sentencia **include()** incluye y evalúa el archivo especificado.

Si "URL fopen wrappers"esta activada en PHP (como está en la configuración inicial), se puede especificar el fichero que se va a incluir usando una URL en vez de un fichero local (con su Path) Ver [Ficheros remotos](#) y **fopen()** para más información.

Un punto importante sobre su funcionamiento es que cuando un archivo se incluye con **include()** o se requiere con **require()**, el intérprete sale del modo PHP y entra en modo HTML al principio del archivo referenciado, y vuelve de nuevo al modo PHP al final. Por esta razón, cualquier código dentro del archivo referenciado que debiera ser ejecutado como código PHP debe ser encerrado dentro de [etiquetas válidas de comienzo y fin de PHP](#).

Esto sucede cada vez que se encuentra la sentencia **include()**, así que se puede usar una sentencia **include()** dentro de una estructura de bucle para incluir un número de archivos diferentes.

```
$archivos = array ('primero.inc', 'segundo.inc', 'tercero.inc');
for ($i = 0; $i < count($archivos); $i++) {
    include $archivos[$i];
}
```

include() difiere de **require()** en que la sentencia **include** se re-evalúa cada vez que se encuentra (y sólo cuando está siendo ejecutada), mientras que la sentencia **require()** se reemplaza por el archivo referenciado cuando se encuentra por primera vez, se vaya a evaluar el contenido del archivo o no (por ejemplo, si está dentro de una sentencia **if** cuya condición evaluada es falsa).

Debido a que **include()** es una construcción especial del lenguaje, se debe encerrar dentro de un bloque de sentencias si está dentro de un bloque condicional.

```
/* Esto es ERRÓNEO y no funcionará como se desea. */

if ($condicion)
    include($archivo);
else
    include($otro);

/* Esto es CORRECTO. */

if ($condicion) {
    include($archivo);
} else {
    include($otro);
}
```

En ambos, PHP3 y PHP4, es posible ejecutar una sentencia **return** dentro de un archivo incluido con **include()**, para terminar el procesado de ese archivo y volver al archivo de comandos que lo llamó. Existen algunas diferencias en el modo en que esto funciona, no obstante. La primera es que en PHP3, **return** no puede aparecer dentro de un bloque a menos que sea un bloque de función, en el cual **return** se aplica a esa función y no al archivo completo. En PHP4, no obstante, esta restricción no existe. También, PHP4 permite devolver valores desde archivos incluidos con **include()**. Se puede capturar el valor de la llamada a **include()** como se haría con una función normal. Esto genera un error de intérprete en PHP3.

Ejemplo 11-1. include() en PHP3 y PHP4

Asumamos la existencia del siguiente archivo (llamado `test.inc`) en el mismo directorio que el archivo principal:

```
<?php
echo "Antes del return <br>\n";
if ( 1 ) {
```

```

        return 27;
}
echo "Después del return <br>\n";
?>
```

Asumamos que el archivo principal (`main.html`) contiene lo siguiente:

```
<?php
$retval = include( 'test.inc' );
echo "El archivo devolvió: '$retval'<br>\n";
?>
```

Cuando se llama a `main.html` en PHP3, generará un error del intérprete en la linea 2; no se puede capturar el valor de un `include()` en PHP3. En PHP4, no obstante, el resultado será:

```
Antes del return
El archivo devolvió: '27'
```

Ahora, asumamos que se ha modificado `main.html` para que contenga lo siguiente:

```
<?php
include( 'test.inc' );
echo "De vuelta en main.html<br>\n";
?>
```

En PHP4, la salida será:

```
Antes del return
De vuelta en main.html
```

No obstante, PHP3 dará la siguiente salida:

```
Antes del return
27De vuelta en main.html
```

```
Parse error: parse error in /home/torben/public_html/phptest/main.html on line 5
```

El error del intérprete es resultado del hecho de que la sentencia `return` está encerrada en un bloque de no-función dentro de `test.inc`. Cuando el `return` se mueve fuera del bloque, la salida es:

```
Antes del return
27De vuelta en main.html
```

El '27' espúreo se debe al hecho de que PHP3 no soporta devolver valores con `return` desde archivos como ese.

When a file is `include()`d, 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()`d 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()`d 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`

Ejemplo 11-2. utils.inc

```

<?php
define(PHPVERSION, floor(phpversion()));
echo "GLOBALS ARE NICE\n";
function goodTea() {
    return "Oolong tea tastes good!";
}
?>

```

Ejemplo 11-3. foolib.inc

```

<?php
require ("utils.inc");
function showVar($var) {
    if (PHPVERSION == 4) {
        print_r($var);
    } else {
        dump_var($var);
    }
}

// bunch of other functions ...
?>

```

And then you write a script `cause_error_require.php`

Ejemplo 11-4. 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 output will be (using PHP 4.0.1pl2):

```
GLOBALS ARE NICE
GLOBALS ARE NICE

Fatal error: Cannot redeclare causeerror() in utils.inc on line 5
```

By modifying `foolib.inc` and `cause_error_require.php` to use `require_once()` instead of `require()` and renaming the last one to `avoid_error_require_once.php`, we have:

Ejemplo 11-5. foolib.inc (fixed)

```
...
require_once("utils.inc");
function showVar($var) {
...
}
```

Ejemplo 11-6. 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 reassessments, etc.

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()**, **require_once()**, **get_required_files()**, **get_included_files()**, **readfile()**, and **virtual()**.

Capítulo 12. Funciones

Funciones definidas por el usuario

Una función se define con la siguiente sintaxis:

```
function foo ($arg_1, $arg_2, ..., $arg_n) {
    echo "Función de ejemplo.\n";
    return $retval;
}
```

Cualquier instrucción válida de PHP puede aparecer en el cuerpo de la función, incluso otras funciones y definiciones de [clases](#).

En PHP3, las funciones deben definirse antes de que se referencien. En PHP4 no existe tal requerimiento.

PHP no soporta la sobrecarga de funciones, y tampoco es posible redefinir u ocultar funciones previamente declaradas.

PHP3 no soporta un número variable de parámetros, aunque sí soporta parámetros por defecto (ver [Valores por defecto de los parámetros](#) para más información). PHP4 soporta ambos: ver [Listas de longitud variable de parámetros](#) y las referencias de las funciones **func_num_args()**, **func_get_arg()**, y **func_get_args()** para más información.

Parámetros de las funciones

La información puede suministrarse a las funciones mediante la lista de parámetros, una lista de variables y/o constantes separadas por comas.

PHP soporta pasar parámetros por valor (el comportamiento por defecto), [por referencia](#), y [parámetros por defecto](#). Listas de longitud variable de parámetros sólo están soportadas en PHP4 y posteriores; ver [Listas de longitud variable de parámetros](#) y la referencia de las funciones **func_num_args()**, **func_get_arg()**, y **func_get_args()** para más información. Un efecto similar puede conseguirse en PHP3 pasando un array de parámetros a la función:

```
function takes_array($input) {
    echo "$input[0] + $input[1] = ", $input[0]+$input[1];
}
```

Pasar parámetros por referencia

Por defecto, los parámetros de una función se pasan por valor (de manera que si cambias el valor del argumento dentro de la función, no se ve modificado fuera de ella). Si deseas permitir a una función modificar sus parámetros, debes pasarlo por referencia.

Si quieres que un parámetro de una función siempre se pase por referencia, puedes anteponer un ampersand (&) al nombre del parámetro en la definición de la función:

```
function add_some_extra(&$string) {
    $string .= ' y algo más.';
}
$str = 'Esto es una cadena, ';
add_some_extra($str);
echo $str; // Saca 'Esto es una cadena, y algo más.'
```

Si deseas pasar una variable por referencia a una función que no toma el parámetro por referencia por defecto, puedes anteponer un ampersand al nombre del parámetro en la llamada a la función:

```
function foo ($bar) {
```

```

    $bar .= ' y algo más.';
}
$str = 'Esto es una cadena, ';
foo ($str);
echo $str; // Saca 'Esto es una cadena,
foo (&$str);
echo $str; // Saca 'Esto es una cadena, y algo más.'

```

Parámetros por defecto

Una función puede definir valores por defecto para los parámetros escalares estilo C++:

```

function makecoffee ($type = "cappuccino") {
    return "Hacer una taza de $type.\n";
}
echo makecoffee ();
echo makecoffee ("espresso");

```

La salida del fragmento anterior es:

```

Hacer una taza de cappuccino.
Hacer una taza de espresso.

```

El valor por defecto tiene que ser una expresión constante, y no una variable o miembro de una clase.

En PHP 4.0 también es posible especificar `unset` como parámetro por defecto. Esto significa que el argumento no tomará ningún valor en absoluto si el valor no es suministrado.

Destacar que cuando se usan parámetros por defecto, estos tienen que estar a la derecha de cualquier parámetro sin valor por defecto; de otra manera las cosas no funcionarán de la forma esperada. Considera el siguiente fragmento de código:

```

function makeyogurt ($type = "acidophilus", $flavour) {
    return "Haciendo un bol de $type $flavour.\n";
}

echo makeyogurt ("mora"); // No funcionará de la manera esperada

```

La salida del ejemplo anterior es:

```

Warning: Missing argument 2 in call to makeyogurt() in
/usr/local/etc/httpd/htdocs/php3test/functest.html on line 41
Haciendo un bol de mora.

```

Y ahora, compáralo con:

```

function makeyogurt ($flavour, $type = "acidophilus") {
    return "Haciendo un bol de $type $flavour.\n";
}

echo makeyogurt ("mora"); // funciona como se esperaba

```

La salida de este ejemplo es:

```
Haciendo un bol de acidophilus mora.
```

Listas de longitud variable de parámetros

PHP4 soporta las listas de longitud variable de parámetros en las funciones definidas por el usuario. Es realmente fácil, usando las funciones **func_num_args()**, **func_get_arg()**, y **func_get_args()**.

No necesita de ninguna sintaxis especial, y las listas de parámetros pueden ser escritas en la llamada a la función y se comportarán de la manera esperada.

Devolver valores

Los valores se retornan usando la instrucción opcional `return`. Puede devolverse cualquier tipo de valor, incluyendo listas y objetos.

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

No puedes devolver múltiples valores desde una función, pero un efecto similar se puede conseguir devolviendo una lista.

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

old_function

La instrucción `old_function` permite declarar una función usando una sintaxis idéntica a la de PHP/FI2 (excepto que debes reemplazar 'function' por 'old_function').

Es una característica obsoleta, y debería ser usada únicamente por el conversor PHP/FI2->PHP3.

Aviso

Las funciones declaradas como `old_function` no pueden llamarse desde el código interno de PHP. Entre otras cosas, esto significa que no puedes usarlas en funciones como **usort()**, **array_walk()**, y **register_shutdown_function()**. Puedes solventar esta limitación escribiendo un "wrapper"(en PHP3 normal) que a su vez llame a la función declarada como `old_function`.

Funciones variable

PHP soporta el concepto de funciones variable, esto significa que si una variable tiene unos paréntesis añadidos al final, PHP buscará una función con el mismo nombre que la evaluación de la variable, e intentará ejecutarla. Entre otras cosas, esto te permite implementar retrollamadas (callbacks), tablas de funciones y demás.

Ejemplo 12-1. Ejemplo de función variable

```
<?php
function foo() {
    echo "Dentro de foo()<br>\n";
}

function bar( $arg = " " ) {
    echo "Dentro de bar(); el parámetro fue '$arg'.<br>\n";
}

$func = 'foo';
$func();
$func = 'bar';
$func( 'test' );
?>
```

Capítulo 13. Clases y Objetos

class

Una clase es una colección de variables y de funciones que acceden a esas variables. Una clase se define con la siguiente sintaxis:

```
<?php
class Cart {
    var $items; // Items en nuestro carro de la compra

    // Añadir $num artículos de tipo $artnr al carro

    function add_item ($artnr, $num) {
        $this->items[$artnr] += $num;
    }

    // Sacar $num artículos del tipo $artnr del carro

    function remove_item ($artnr, $num) {
        if ($this->items[$artnr] > $num) {
            $this->items[$artnr] -= $num;
            return true;
        } else {
            return false;
        }
    }
}
?>
```

El ejemplo define una clase llamada Cart que consiste en un array asociativo de artículos en el carro y dos funciones para meter y sacar ítems del carro

Las clases son tipos, es decir, son plantillas para variables. Tienes que crear una variable del tipo deseado con el operador new.

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

Este ejemplo crea un objeto \$cart de clase Cart. La función add_item() de ese objeto se llama para añadir un ítem del artículo número 10 al carro.

Las Clases pueden ser extensiones de otras clases. Las clases extendidas o derivadas tienen todas las variables y funciones de la clase base y lo que les añadas al extender la definición. La herencia múltiple no está soportada.

```
class Named_Cart extends Cart {
    var $owner;

    function set_owner ($name) {
        $this->owner = $name;
    }
}
```

Ese ejemplo define una clase Named_Cart (carro con nombre o dueño) que tiene todas las variables y funciones de Cart, y además añade la variable \$owner y una función adicional set_owner(). Un carro con nombre se crea de la forma habitual y, una vez hecho, puedes acceder al propietario del carro. En los carros con nombre también puedes acceder a las funciones normales del carro:

```
$ncart = new Named_Cart; // Creamos un carro con nombre
$ncart->set_owner ("kris"); // Nombramos el carro
print $ncart->owner; // Imprimimos el nombre del propietario
```

```
$ncart->add_item ("10", 1); // Funcionalidad heredada de Cart
```

Entre funciones de una clase, la variable `$this` hace referencia al propio objeto. Tienes que usar `$this->loquesea` para acceder a una variable o función llamada loquesea del objeto actual.

Los constructores son funciones de una clase que se llaman automáticamente al crear una nueva instancia (objeto) de una clase. Una función se convierte en constructor cuando tiene el mismo nombre que la clase.

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

Este ejemplo define una clase `Auto_Cart` que es un `Cart` junto con un constructor que inicializa el carro con un ítem del tipo de artículo "10" cada vez que se crea un nuevo `Auto_Cart` con `"new"`. Los constructores también pueden recibir parámetros y estos parámetros pueden ser opcionales, lo que los hace más útiles.

```
class Constructor_Cart extends Cart {
    function Constructor_Cart ($item = "10", $num = 1) {
        $this->add_item ($item, $num);
    }
}

// Compramos las mismas cosas aburridas de siempre

$default_cart = new Constructor_Cart;

// Compramos las cosas interesantes

$different_cart = new Constructor_Cart ("20", 17);
```

Atención

Para las clases derivadas, el constructor de la clase padre no es llamado automáticamente cuando se llama al constructor de la clase derivada.

Capítulo 14. References Explained

What are References

References in PHP are means to call same variable content with different names. They are not like C pointers, they are symbol table aliases. Note that in PHP, variable names and variable content are different, so same content can have different names. The most close analogy is 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 do References

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 second thing references do is to pass variables by-reference. This is done by making local function variable and caller variable to be 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`.

The third thing reference can do is [return by-reference](#).

What aren't References

As said above, references aren't pointers. That means, the following construct won't do what you expect:

```
function foo (&$var) {
    $var = & $GLOBALS[ "baz" ];
}
foo($bar);
```

What will happen 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 caller to something else using 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).

Returning References

Returning by-refernce it is useful when you want to use function to find variable which 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, property of the object returned by the `find_var` function would be set, not of 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 than reference binding and not 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 the Reference

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.

Parte III. Características

Capítulo 15. Manejando errores

Hay 4 tipos de errores y avisos en PHP. Esto son:

- 1 - Errores Normales de Funciones (Normal Function Errors)
- 2 - Avisos Normales (Normal Warnings)
- 4 - Errores del Analizador de código (Parser Errors)
- 8 - Avisos (Notices, advertencia que puedes ignorar, pero que puede implicar un error en tu código).

Los 4 números de arriba son sumados para definir un nivel de aviso de error. El nivel de aviso de error por defecto es el nivel 7, el cual es la suma de 1+2+4, es decir todo excepto los avisos. Este nivel puede ser cambiado en el fichero php3.ini con la directiva `error_reporting`. También puede ser configurado en el fichero de configuración del servidor de páginas Apache httpd.conf, con la directiva `php3_error_reporting` o también se puede cambiar en tiempo de ejecución usando la función `error_reporting()`.

Todas las [expresiones PHP](#) pueden también ser llamadas con el prefijo "@", el cual desactiva el aviso de errores para esa expresión en particular. Si ocurre un error en una expresión en tal situación y la característica `track_errors` está habilitada, podrás encontrar el mensaje de error en la variable global `$php_errormsg`.

Capítulo 16. Creando imágenes GIF

PHP no está limitado a crear solo salidas de HTML. Puede ser usado también para crear ficheros de imágenes GIF, o incluso mejor secuencias de imágenes GIF. Necesitará compilar PHP con la librería de funciones de imágenes GD para esta tarea.

Ejemplo 16-1. Creación de GIFs con PHP

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

Este ejemplo será llamado desde una página con una línea como esta: <imgsrc="button.php3?text"> Este script de arriba button.php3 toma esta cadena "text" la situa sobre la imagen base, en este caso es "images/button1.gif" y muestra la imagen resultante. Esta es una forma muy conveniente para evitar tener que dibujar un nuevo botón cada vez que quiera cambiar el texto del mismo. Con este método los botones son generados dinámicamente.

Capítulo 17. Autentificación HTTP con PHP

Las características de autentificación HTTP en PHP solo están disponibles cuando se está ejecutando como un módulo en Apache y hasta ahora no lo están en la versión CGI. En un script PHP como módulo de Apache, se puede usar la función **Header()** para enviar un mensaje de "Autentificación requerida" al navegador cliente haciendo que muestre una ventana de entrada emergente con nombre de usuario y contraseña. Una vez que el usuario ha llenado el nombre y la contraseña, la URL que contiene el script PHP vuelve a ser llamada con las variables `$PHP_AUTH_USER`, `$PHP_AUTH_PW` y `$PHP_AUTH_TYPE` llenadas con el nombre de usuario, la contraseña y el tipo de autentificación respectivamente. Sólo autentificación "Básica" está soportada en este momento. Consulte la función **Header()** para más información.

Un fragmento de script de ejemplo que fuerce la autentificación del cliente en una página sería como el siguiente:

Ejemplo 17-1. Ejemplo de autentificación HTTP

```
<?php
if(!isset($PHP_AUTH_USER)) {
    Header("WWW-Autentificación: Basic realm=\"Mi Reino\" ");
    Header("HTTP/1.0 401 No autorizado");
    echo "Texto a enviar si pulsa el botón Cancelar\n";
    exit;
} else {
    echo "Hola $PHP_AUTH_USER.<P>";
    echo "Ha introducido $PHP_AUTH_PW como su contraseña.<P>";
}
?>
```

En vez de, sencillamente, mostrar `$PHP_AUTH_USER` y `$PHP_AUTH_PW`, seguramente quiera comprobar la validez del nombre de usuario y la contraseña. Tal vez enviando una consulta a una base de datos o buscando el usuario en un fichero dbm.

Vigile aquí los navegadores Internet Explorer con bugs. Parecen muy quisquillosos con el orden de las cabeceras. Enviar la cabecera *WWW-Autentificación* antes que la cabecera *HTTP/1.0 401* parece ser el truco por ahora.

Para prevenir que alguien escriba un script que revele la contraseña de una página que ha sido autenticada a través de algún mecanismo externo tradicional, las variables `PHP_AUTH` no serán llenadas si algún tipo de autentificación externo ha sido activado para una página en particular. En este caso, la variable `REMOTE_USER` puede ser usada para identificar al usuario autenticado externamente.

Nota, a pesar de todo, lo ya dicho no proteje de que alguien que controle una URL no autenticada robe contraseñas de URLs autenticadas en el mismo servidor.

Tanto Netscape como Internet Explorer borrarán la caché de la ventana de autentificación en el navegador local después de recibir una respuesta 401 del servidor. Esto puede usarse, de forma efectiva, para "desconectar" a un usuario, forzandole a reintroducir su nombre y contraseña. Algunas personas usan esto para "hacer caducar" entradas, o para proveer un botón de "desconectar".

Ejemplo 17-2. Ejemplo de autentificación HTTP forzando una reentrada

```
<?php
function authenticate() {
    Header("WWW-Autentificación: reino básico='Test Autentificación Sistema '");
    Header("HTTP/1.0 401 No autorizado");
    echo "Debe introducir un nombre de usuario y contraseña válidos para acceder a este recurso\n";
    exit;
}

if(!isset($PHP_AUTH_USER) || ($SeenBefore == 1 && !strcmp($OldAuth, $PHP_AUTH_USER))) {
    authenticate();
}
```

```
else {
echo "Bienvenido: $PHP_AUTH_USER<BR>" ;
echo "Old: $OldAuth" ;
echo "<FORM ACTION=\"$PHP_SELF\" METHOD=POST>\n";
echo "<INPUT TYPE=HIDDEN NAME=\"SeenBefore\" VALUE=\"1\">\n";
echo "<INPUT TYPE=HIDDEN NAME=\"OldAuth\" VALUE=\"$PHP_AUTH_USER\">\n";
echo "<INPUT TYPE=Submit VALUE=\"Re Authenticate\">\n";
echo "</FORM>\n";
}

?>
```

Este comportamiento no es requerido por el estándar de autentificación básica de HTTP, por lo que nunca debe depender de esto. Pruebas con Lynx han demostrado que Lynx no borra las credenciales de autentificación con una respuesta 401 del servidor, por lo que pulsando atrás y después adelante abriría el recurso de nuevo (siempre que los requerimientos de contraseña no hayan cambiado).

Además note que esto no funciona usando el servidor IIS de Microsoft y la versión CGI de PHP debido a una limitación del IIS

Capítulo 18. Cookies

PHP soporta transparentemente cookies HTTP. Las Cookies son un mecanismo que sirve para almacenar datos en el navegador del usuario remoto, para así poder identificar al usuario cuando vuelva. Se pueden poner cookies usando la función **setcookies()**. Las Cookies son parte de la cabecera HTTP, por tanto la función **setcookie()** debe ser llamada antes de que se produzca cualquier salida al navegador. Esta limitación es la misma a la de la función **header()**.

Cualquier cookie enviada a ti desde el cliente, automáticamente se convertirá en una variable PHP igual como ocurre con los métodos de datos GET y POST. Si deseas asignar multiples valores a una cookie simple, añade simplemente `[]` a el nombre de la cookie. Para más detalles ver la función **setcookie()**.

Capítulo 19. El envío de archivos

Envío de archivos con el método POST

PHP es capaz de recibir envíos de archivo de cualquier navegador que cumpla la norma RFC-1867 (entre los que se incluyen Netscape Navigator 3 o posterior, Microsoft Internet Explorer 3 con un parche o posterior sin éste). Esta característica permite que los usuarios envíen archivos de texto y binarios. Mediante la autenticación y funciones de manejo de archivos de PHP, es posible un control total de quién puede enviar archivos y que se hace con éstos una vez recibidos.

Es importante destacar que PHP también soporta el método PUT para envío de archivos tal y como lo utiliza Netscape Composer y el cliente Amaya de W3C. Consulte [Soporte del método PUT](#) para más detalles.

Una página de envío de archivos se puede crear mediante un formulario parecido a éste:

Ejemplo 19-1. Formulario de envío de archivo

```
&lt;FORM ENCTYPE="multipart/form-data" ACTION="_URL_" METHOD=POST&gt;
&lt;INPUT TYPE="hidden" name="MAX_FILE_SIZE" value="1000"&gt;
Enviar este archivo: &lt;INPUT NAME="userfile" TYPE="file"&gt;
&lt;INPUT TYPE="submit" VALUE="Enviar"&gt;
&lt;/FORM&gt;
```

La _URL_ debe tener como destino un script PHP. El campo MAX_FILE_SIZE debe encontrarse antes del campo INPUT y su valor determina el tamaño máximo de archivo que se puede enviar en bytes. Tras la recepción del archivo se definirán en el script PHP destino las siguientes variables:

- \$userfile - El archivo temporal que se ha guardado en el servidor.
- \$userfile_name - El nombre original del archivo enviado.
- \$userfile_size - El tamaño del archivo recibido.
- \$userfile_type - El tipo mime del archivo si el navegador envio esta información. Por ejemplo: "image/gif".

Es importante recordar que la primera palabra "\$userfile" de éstas variables corresponde al nombre ("NAME=") del campo "INPUT TYPE=file" del formulario. En el ejemplo anterior usamos "userfile".

Los archivos enviados serán guardados en el directorio temporal por defecto del servidor. Podemos variar este directorio mediante la variable de entorno TMPDIR en el entorno donde corre PHP. No se puede establecer este valor usando **putenv()** desde un script PHP.

El script PHP que recibe el archivo enviado debe implementar las acciones que se deben llevar a cabo con el archivo acabado de recibir. Por ejemplo se podría utilizar \$file_size para decidir descartar los archivos que sean demasiado pequeños o demasiado grandes. Sean cual sean las acciones a tomar se debe borrar el archivo temporal o moverlo a algún otro directorio.

El archivo recibido será eliminado inmediatamente del directorio temporal al finalizar el script PHP que lo recibió si no ha sido movido o renombrado.

Errores comunes

El valor de MAX_FILE_SIZE no puede ser mayor que el tamaño del archivo que se especifica en la variable upload_max_filesize del archivo PHP3.ini o la correspondiente directiva php3_upload_max_filesize de Apache. Por defecto es 2 Megabytes.

El servidor CERN parece que elimina cualquier cosa antes del primer espacio en blanco en la cabecera mime content-type que recibe de los clientes. Mientras esto no varie, CERN httpd no podrá soportar el envío de archivos.

Envío de más de un archivo

Es posible el envío de varios archivos simultáneamente y poder clasificar la información automáticamente por arrays. Esto de hace de la misma manera en que se organizan por arrays los SELECT o CHECKBOX:

Nota: El soporte para múltiple envíos de archivos se añadió en la versión 3.0.10

Ejemplo 19-2. Formulario de envío multiple de archivos

```
&lt;form action="file-
upload.html" method="post" enctype="multipart/form-data">
    Enviar estos archivos:&lt;br&gt;
    &lt;input name="userfile[]" type="file"&gt;&lt;br&gt;
    &lt;input name="userfile[]" type="file"&gt;&lt;br&gt;
    &lt;input type="submit" value="Enviar"&gt;
&lt;/form&gt;
```

Cuando el formulario es procesado, los arrays \$userfile, \$userfile_name, y \$userfile_size se crearán de alcance global (igual que \$HTTP_POST_VARS). Cada uno será un array con índice numérico con los valores apropiados para los archivos enviados.

Por ejemplo, supongamos que los siguientes archivos /home/test/review.html y /home/test/xwp.out son enviados. En este caso, \$userfile_name[0] almacenaría el valor review.html, y \$userfile_name[1] almacenaría el valor xwp.out. Así, \$userfile_size[0] almacenaría el tamaño de review.html y así con los valores siguientes.

Soporte del método PUT

PHP soporta el metodo HTTP PUT que usan aplicaciones como Netscape Composer y Amaya de W3C. Las peticiones PUT son más sencillas que el método POST. Un ejemplo:

```
PUT /path/filename.html HTTP/1.1
```

Esto normalmente significaría que el cliente remoto quiere salvar el contenido como: /path/filename.html en tu árbol web. Lógicamente no una buena idea que la gente pueda escribir en tu árbol web. Para manipular esta petición debes decirle al servidor que esta petición sea atendida por un script PHP. En Apache, por ejemplo, se utiliza para esto la directiva *Script* en los alguno de los archivos de configuración del servidor. Un sitio típico de uso es dentro del bloque <Directory> o quizás en el bloque <Virtualhost>. Una linea así debería hacer ésta función:

```
Script PUT /put.php3
```

Ésto le dice a Apache que envie todas peticiones PUT para URIs que contengan esta linea al script put.php3. Se asume que PHP se encuentra activo y con la extensión php3 enlazada a él.

Dentro del script put.php3 se podría implementar algo así:

```
&lt;? copy($PHP_UPLOADED_FILE_NAME, $DOCUMENT_ROOT.$REQUEST_URI); ?&gt;
```

Esto copiaría el archivo a la localización requerida por el cliente remoto. Aquí se pueden ejecutar funciones de autenticación de usuario o cualquier otro tipo de chequeo. El archivo se guarda en el archivo temporal del sistema servidor de la misma manera que el [Método POST](#). Cuando la petición finaliza, el archivo temporal es eliminado. En

consecuencia el script debe proceder al trato de éste inmediatamente, ya sea para copiarlo, renombrarlo, etc. El archivo se encuentra en la variable `$PHP_PUT_FILENAME`, y el destino sugerido por el cliente en la variable `$REQUEST_URI` (puede variar en servidores diferentes de Apache). No es necesario hacer caso al destino sugerido por el cliente. Por ejemplo se podrían copiar los archivos enviados a directorios especialmente designados para esta tarea.

Capítulo 20. Usando archivos remotos

Siempre que el soporte para la "envoltura URL fopen" esté habilitado cuando se configura PHP (lo cual ocurre a menos que se pasa explícitamente la opción `-disable-url-fopen-wrapper` a configure), se pueden usar URLs HTTP y FTP con la mayoría de las funciones que toman un archivo como parámetro, incluyendo las sentencias `require()` e `include()`.

Nota: No se pueden usar archivos remotos en las sentencias `include()` y `require()` en Windows.

Por ejemplo, se puede usar este para abrir un archivo en un servidor web remoto, analizar en la salida la información que se quiera, y entonces, usar la información en una consulta a base de datos, o simplemente para sacarla en un estilo que coincida con el resto de su sitio web.

Ejemplo 20-1. Obtener el título de una página remota

```
<?php
$archivo = fopen("http://www.php.net/", "r");
if (!$archivo) {
    echo "<p>No se pudo abrir el archivo remoto.\n";
    exit;
}
while (!feof($archivo)) {
    $linea = fgets($archivo, 1024);
    /* Esto sólo funciona si el título y sus etiquetas
     * están en una línea. */
    if (eregi("<title>(.*)</title>", $linea, $salida)) {
        $title = $salida[1];
        break;
    }
}
fclose($file);
?>
```

También se puede escribir a archivos en un FTP siempre que se conecte como un usuario con los correctos derechos de acceso, y el archivo no exista ya. Para conectar como un usuario distinto de 'anonymous', se necesita especificar el nombre de usuario (y posiblemente contraseña) dentro de la URL, tales como '`ftp://usuario:clave@ftp.ejemplo.com/camino/a/archivo`'. (Se puede usar la misma clase de sintaxis para acceder a archivos via HTTP cuando se requería una autenticación de same sort of syntax to access files via HTTP when they require Basic authentication.)

Ejemplo 20-2. Storing data on a remote server

```
<?php
$file = fopen("ftp://ftp.php.net/incoming/outputfile", "w");
if (!$file) {
    echo "<p>Unable to open remote file for writing.\n";
    exit;
}
/* Write the data here. */
fputs($file, "$HTTP_USER_AGENT\n");
fclose($file);
?>
```

Nota: You might get the idea from the example above to use this technique to write to a remote log, but as mentioned above, you can only write to a new file using the URL fopen() wrappers. To do distributed logging like that, you should take a look at `syslog()`.

Capítulo 21. Manejando conexiones

Nota: Todo lo siguiente se aplica a partir de la 3.0.7 y posterior.

Internamente en PHP se mantiene el estado de la conexión. Hay 3 posibles estados:

- 0 - NORMAL
- 1 - ABORTED (Abortado)
- 2 - TIMEOUT (Fuera de tiempo)

Cuando un script PHP se está ejecutando se activa el estado NORMAL. Si el cliente remoto se desconecta, se pasa al estado ABORTADO. Esto suele ocurrir cuando el usuario pulsa en el botón STOP del navegador. Si se alcanza el límite de tiempo impuesto por PHP (ver `set_time_limit()`), se pasa al estado TIMEOUT.

Puedes decidir si quieres que la desconexión de un cliente cause que tu script sea abortado. Algunas veces es cómodo que tus scripts se ejecuten por completo, incluso si no existe ya un navegador remoto que reciba la salida. El comportamiento por defecto es sin embargo, que tu script se aborte cuando el cliente remoto se desconecta. Este comportamiento puede ser configurado vía la directiva `ignore_user_abort` en el fichero `php3.ini`, o también con la función `ignore_user_abort()`. Si no le especificas al PHP que cuando un usuario aborte lo ignore, tu script terminará su ejecución. La única excepción es si tienes registrada una función de desconexión usando la función `register_shutdown_function()`. Con una función de desconexión, cuando un usuario remoto pulsa en el botón STOP, la próxima vez que tu script intenta mostrar algo, PHP detecta que la conexión ha sido abortada y se llama a la función de desconexión. Esta función de desconexión también se llama al final de la ejecución de tu script cuando se ha ejecutado normalmente, de manera que si quieres hacer algo diferente en caso de que un cliente se haya desconectado, puedes usar la función `connection_aborted()`. Esta función devuelve verdadero si la conexión fue abortada.

Tu script también se puede terminar por un temporizador interno. El timeout por defecto es de 30 segundos. Se puede cambiar usando la directiva `max_execution_time` en el fichero `php3.ini` o la correspondiente directiva `php3_max_execution_time` en la configuración del servidor de páginas Apache, como también con la función `set_time_limit()`. Cuando el temporizador expira, el script se aborta como en el caso de la desconexión del cliente, de manera que si se ha definido una función de desconexión, esta se llamará. Dentro de esta función de desconexión, puedes comprobar si fue el timeout el que causó que se llamara a la función de desconexión, llamando a la función `connection_timeout()`. Esta función devolverá verdadero si el timeout causa que se llame a la función de desconexión.

Hay que destacar que ambos, el estado ABORTED y el TIMEOUT, se pueden activar al mismo tiempo. Esto es posible si le dices a PHP que ignore las desconexiones intencionadas de los usuarios. PHP aún notará el hecho de que el usuario puede haberse desconectado, pero el script continuará ejecutándose. Si se alcanza el tiempo límite de ejecución será abortado y, si se ha definido una función de desconexión, esta será llamada. En este punto, encontrarás que las funciones `connection_timeout()` y `connection_aborted()` devuelven verdadero. Puedes comprobar ambos estados de una manera simple usando la función `connection_status()`. Esta función devuelve un campo de bit de los estados activos. De este modo, si ambos estados están activos devolvería por ejemplo un valor 3.

Capítulo 22. Conexiones persistentes a bases de datos

Las conexiones persistentes son enlaces SQL que no se cierran cuando termina la ejecución del archivo de comandos. Cuando se pide una conexión persistente, PHP comprueba si hay ya una conexión persistente idéntica (que permanecía abierta desde antes) - y si existe, la usa. Si no existe, crea un enlace. Una conexión 'idéntica' es una conexión que se abrió hacia el mismo "host", con el mismo nombre de usuario y la misma contraseña (donde sea aplicable).

La gente que no está familiarizada con el modo como trabajan y distribuyen la carga los servidores "web" puede confundir que persistente significa lo que no es. En particular, ellas *no* te dan la habilidad de abrir 'sesiones de usuario' en el mismo enlace SQL, *no* dan la habilidad de construir una transacción de forma eficiente, y no hacen un montón de otras cosas. De hecho, para ser extremadamente claros sobre el tema las conexiones persistentes *no* te dan *ninguna* funcionalidad que no fuera posible con sus hermanas no-persistentes.

¿Por qué?

Esto tiene que ver con el modo como funcionan los servidores "web". Hay tres modos en que un servidor "web" puede utilizar PHP para generar páginas web.

El primer método es usar PHP como una capa CGI. Cuando corre de este modo, se crea y destruye una instancia del intérprete PHP por cada página solicitada (para una página PHP) a tu servidor. Debido a que se destruye después de cada petición, cualquier recurso que adquiera (como un enlace a un servidor de base de datos SQL) se cierra cuando es destruido. En este caso, no se gana nada si se intentan usar conexiones persistentes.

El segundo, y más popular, método es correr PHP como un módulo en un servidor web multiproceso, lo cual actualmente sólo incluye Apache. Un servidor multiproceso tiene típicamente un proceso (el padre) que coordina un conjunto de procesos (sus hijos) que realmente hacen el trabajo de servir las páginas web. Cuando entra cada petición de un cliente, es entregada a uno de los hijos que no esté ya sirviendo a otro cliente. Esto significa que cuando el mismo cliente hace una segunda petición al servidor, puede ser atendida por un proceso hijo distinto del de la primera vez. Lo que una conexión persistente hace por ti en este caso es hacerlo de tal modo que cada proceso hijo sólo necesita conectar a tu SQL server la primera vez que sirve una página que hace uso de una conexión así. Cuando otra página solicita una conexión a SQL server, puede reutilizar la conexión que el hijo estableció previamente.

El último método es usar PHP como un "plug-in" para un servidor web multihilo. En la actualidad es solamente teórico – PHP no funciona aún como "plug-in" para ningún servidor web multihilo. Hay trabajo en progreso para soportar ISAPI, WSAPI y NSAPI (en Windows), lo cual permitirá a PHP ser utilizado como "plug-in" para servidores web multihilo como Netscape FastTrack, Internet Information Server (IIS) de Microsoft, y O'Reilly's WebSite Pro. Cuando esto ocurra, el comportamiento será exactamente el mismo que para el modelo de multiprocesador descrito anteriormente.

Si las conexiones persistentes no aportan ninguna funcionalidad añadida, ¿para qué son buenas?

La respuesta aquí es extremadamente simple – eficiencia. Las conexiones persistentes son buenas si las cabeceras de control para crear un enlace a tu servidor SQL es alta. Que estas cabeceras sean o no realmente altas depende de muchos factores. Como, qué clase de base de datos es, si esta o no situada en el mismo ordenador que el servidor web, cómo está de cargada la máquina donde se encuentre el servidor SQL, y otras así. El hecho fundamental es que si la cabecera de conexión es alta, las conexiones persistentes te ayudan considerablemente. Ellas hacen que el proceso hijo simplemente conecte solamente una vez durante todo su intervalo de vida, en vez de cada vez que procesa una página que requiere conectar al servidor SQL. Esto significa que por cada hijo que abrió una conexión persistente tendrá su propia conexión persistente al servidor. Por ejemplo, si tienes 20 procesos hijos distintos que corran un archivo de comandos que cree una conexión persistente a tu servidor SQL, tendrás 20 conexiones diferentes a tu servidor SQL, una por cada hijo.

Un resumen importante. Las conexiones persistentes fueron diseñadas para tener una equivalencia uno-a-uno con las conexiones normales. Eso significa que deberás *siempre* ser capaz de reemplazar las conexiones persistentes por conexiones no persistentes y no cambiará, el modo como se comporta el archivo de comandos. Puede cambiar la eficiencia del archivo de comandos (y probablemente lo hará), ¡pero no su comportamiento!

Parte IV. Referencia de las Funciones

I. Funciones específicas de Apache

apache_lookup_uri (PHP 3>= 3.0.4, PHP 4)

Efectua una petición parcial a la URI especificada y devuelve toda la información sobre ella.

```
class apache_lookup_uri (string filename)
```

Esta función efectúa una llamada parcial a URI. Esta llamada no hace sino obtener toda la información importante sobre el recurso pedido y la devuelve en un tipo clase .Las propiedades de esa clase son:

```
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: Nota: apache_lookup_uri solo funciona cuando el PHP está instalado como módulo del Apache.

apache_note (PHP 3>= 3.0.2, PHP 4)

Recibe y establece los valores de una petición en una tabla de notas del Apache

```
string apache_note (string note_name [, string note_value])
```

apache_note() es una función específica del Apache que recibe y establece valores de la petición en una tabla de notas. Si se llama con un solo parámetro,devuelve el valor de note_name. Si se llama con dos parámetros, establece el valor de note_value en note_value y devuelve el valor que había en note_name.

getallheaders (PHP 3, PHP 4)

Recibe todas las cabeceras de una petición HTTP

```
array getallheaders(void);
```

Esta función devuelve asociados en un vector todas las cabeceras de la actual petición HTTP.

Nota: También puedes obtener los valores de las variables de los CGIs mediante variables de entorno, que funcionan, esté o no el PHP funcionando como módulo del Apache. Utiliza **phpinfo()** para ver una lista de todas las variables de entorno definidas de esta forma.

Ejemplo 1. ObtenerTodaslasCabeceras() Ejemplo

```
$cabeceras = getallheaders();
while (list($cabecera, $valor) = each($cabeceras)) {
    echo "$cabecera: $valor<br>\n";
}
```

Este ejemplo visualiza todas las cabeceras de la petición actual.

Nota: ObtenerTodaslasCabeceras() actualmente solo funcionará si el PHP es cargado como módulo del Apache .

virtual (PHP 3, PHP 4)

Ejecuta una sub-petición al Apache

```
int virtual (string filename)
```

virtual() es una función específica del Apache que es equivalente a `<!--#include virtual...-->` en mod_include. Esto ejecuta una sup-petición al Apache .Esto, es util para incluir CGI-scripts o páginas .shtml o cualquier tipo de fichero que puedas procesar mediante el Apache. Los CGI-scripts deberán generar cabeceras válidas. Esto, implica como mínimo un **include()** ó un **require();** La función **virtual()** no puede ser usada para incluir un documento que sea por si mismo un documento PHP.

II. Funciones de matrices

array (unknown)

Crear una matriz

```
array array(...);
```

Devuelve una matriz con los parámetros que se le pasan. A dichos parámetros se les puede dar un índice usando el operador =>.

Nota: **array()** es una construcción del lenguaje que se utiliza para representar matrices literales, no una función regular.

El siguiente ejemplo demuestra cómo crear una matriz bidimensional, cómo especificar claves para matrices asociativas, y cómo especificar índices no consecutivos en matrices normales.

Ejemplo 1. Ejemplo de array()

```
$frutas = array (
    "frutas"  => array("a"=>"naranja", "b"=>"plátano", "c"=>"manzana"),
    "números" => array(1, 2, 3, 4, 5, 6),
    "hoyos"   => array("primero", 5 => "segundo", "tercero")
);
```

Vea también: **list()**.

array_count_values (PHP 4 >= 4.0b4)

Cuenta todos los valores de una matriz

```
array array_count_values (array entrada)
```

array_count_values() devuelve una matriz usando los valores de la matriz *entrada* como claves y su frecuencia de aparición en la *entrada* como valores.

Ejemplo 1. Ejemplo de array_count_values()

```
$matriz = array(1, "hola", 1, "mundo", "hola");
array_count_values($matriz); // devuelve array(1=>2, "hola"=>2, "mundo"=>1)
```

Nota: Esta función fue añadida en el PHP 4.0.

array_flip (PHP 4 >= 4.0b4)

Intercambia los valores de una matriz

```
array array_flip (array trans)
```

array_flip() devuelve una matriz con los valores intercambiados.

Ejemplo 1. Ejemplo de array_flip()

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

Nota: Esta función fue añadida en el PHP 4.0.

array_keys (PHP 4)

Devuelve todas las claves de una matriz

```
array array_keys (array entrada [, mixed val_a_buscar])
```

array_keys() devuelve las claves, numéricas y de cadena, de la matriz *entrada*.

Si se especifica el parámetro opcional *val_a_buscar*, sólo se devuelven las claves para dicho valor. De otro modo, se devuelven todas las claves de la *entrada*.

Ejemplo 1. Ejemplo de array_keys()

```
$matriz = array(0 => 100, "color" => "rojo");
array_keys ($matriz);           // devuelve array (0, "color")

$matriz = array(1, 100, 2, 100);
array_keys ($matriz, 100);    // devuelve array (0, 2)
```

Vea también: **array_values()**.

Nota: Esta función fue añadida en el PHP 4.0.

array_merge (PHP 4)

Combina dos o más matrices

```
array array_merge (array matriz1, array matriz2 [, ...])
```

array_merge() combina los elementos de dos o más matrices conjuntamente de modo que los valores de una son agregados al final de los valores de la anterior. Devuelve la matriz resultante.

Si las matrices de entrada tienen las mismas claves de cadena, el último valor para cada clave reemplazará el valor previo de la misma. Si, por el contrario, las matrices tienen la misma clave numérica, esto no pasa y los valores son simplemente agregados.

Ejemplo 1. Ejemplo de array_merge()

```
$matriz1 = array ("color" => "rojo", 2, 4);
$matriz2 = array ("a", "b", "color" => "verde", "forma" => "trapezoide");
array_merge ($matriz1, $matriz2);
```

La matriz resultante sería array("color"=> "verde", 2, 4, "a", "b", "forma"=> "trapezoide").

Nota: Esta función fue añadida en el PHP 4.0.

array_pad (PHP 4 >= 4.0b4)

Rellena una matriz con un valor hasta el tamaño especificado

```
array array_pad (array entrada, int tama_relleno, mixed valor_relleno)
```

array_pad() Devuelve una copia de la *entrada* rellenada hasta el tamaño *tama_relleno* con el valor *valor_relleno*. Si *tama_relleno* es positivo, entonces la matriz es rellenada por la derecha, y si es negativo, por la izquierda. Si el valor absoluto de *tama_relleno* es menor o igual que el tamaño de la *entrada* no se produce relleno alguno.

Ejemplo 1. Ejemplo de array_pad()

```
$entrada = array (12, 10, 9);

$resultado = array_pad ($entrada, 5, 0);
// el resultado es array (12, 10, 9, 0, 0)

$resultado = array_pad ($entrada, -7, -1);
// el resultado es array (-1, -1, -1, -1, 12, 10, 9)

$resultado = array_pad ($entrada, 2, "no");
// no rellenado
```

array_pop (PHP 4)

Extrae el último elemento de la matriz

```
mixed array_pop (array matriz)
```

array_pop() extrae y devuelve el último valor de la *matriz*, acortando la *matriz* en un elemento.

Ejemplo 1. Ejemplo de array_pop()

```
$pila = array ("naranja", "manzana", "framboesa");
$fruta = array_pop ($pila);
```

Tras esto, \$pila contiene sólo 2 elementos: "naranja" y "manzana", y \$fruta contiene "framboesa".

Vea también: **array_push()**, **array_shift()**, y **array_unshift()**.

Nota: Esta función fue añadida en el PHP 4.0.

array_push (PHP 4)

Inserta uno o más elementos al final de la matriz

```
int array_push (array matriz, mixed var [, ...])
```

array_push() considera a la *matriz* como una pila, e inserta las variables que se le pasan al final de la *matriz*. La longitud de la *matriz* se incrementa en el número de variables insertadas. Tiene el mismo efecto que ejecutar:

```
$matriz[] = $var;
```

para cada *var*.

Devuelve el nuevo número de elementos de la matriz.

Ejemplo 1. Ejemplo de array_push()

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

Este ejemplo dejará \$pila conteniendo 4 elementos: 1, 2, "+", y 3.

Vea también: **array_pop()**, **array_shift()**, y **array_unshift()**.

Nota: Esta función fue añadida en el PHP 4.0.

array_reverse (PHP 4 >= 4.0b4)

Devuelve una matriz con los elementos en orden inverso

```
array array_reverse (array matriz)
```

array_reverse() toma la *matriz* de entrada y devuelve una nueva matriz con los elementos en orden inverso.

Ejemplo 1. Ejemplo de array_reverse()

```
$entrada = array ("php", 4.0, array ("verde", "rojo"));
$resultado = array_reverse ($entrada);
```

Esto hace que \$resultado contenga array (array ("verde", "rojo"), 4.0, "php").

Nota: Esta función fue añadida en PHP 4.0 Beta 3.

array_shift (PHP 4)

Extrae un elemento del comienzo de la matriz

```
mixed array_shift (array matriz)
```

array_shift() extrae el primer valor de la *matriz* y lo devuele, acortando la *matriz* en un elemnto y moviendo todo hacia arriba.

Ejemplo 1. Ejemplo de array_shift()

```
$args = array ("-v", "-f");
$opcion = array_shift ($args);
```

Esto da como resultado que \$args tenga como elemento restante "-f"y que \$opcion valga "-v".

Vea también: **array_unshift()**, **array_push()**, y **array_pop()**.

Nota: Esta función fue añadida en el PHP 4.0.

array_slice (PHP 4)

Extrae una porción de la matriz

```
array array_slice (array matriz, int desplazamiento [, int tamano])
```

array_slice() devuelve una secuencia de elementos de la *matriz* especificada por los parámetros *desplazamiento* y *tamano*.

Si el *desplazamiento* es positivo, la secuencia comenzará en dicha posición de la *matriz*. Si el *desplazamiento* es negativo, la secuencia comenzará en esa posición desde el final de la *matriz*.

Si se especifica el *tamano* y éste es positivo, la secuencia contendrá tantos elementos como se diga en él. Si fuese negativo, la secuencia se detendrá a tantos elementos del final de la matriz. Si se omite, la secuencia contendrá todos los elementos desde el *desplazamiento* hasta el final de la *matriz*.

Ejemplo 1. Ejemplo de array_slice() examples

```
$entrada = array ("a", "b", "c", "d", "e");

$salida = array_slice ($entrada, 2);           // devuelve "c", "d", y "e"
$salida = array_slice ($entrada, 2, -1);        // devuelve "c", "d"
$salida = array_slice ($entrada, -2, 1);         // devuelve "d"
$salida = array_slice ($entrada, 0, 3);          // devuelve "a", "b", y "c"
```

Vea también: **array_splice()**.

Nota: Esta función fue añadida en el PHP 4.0.

array_splice (PHP 4)

Suprime una porción de la matriz y la sustituye por otra cosa

```
array array_splice (array entrada, int desplazamiento [, int tamano [, array sustitucion]])
```

array_splice() suprime los elementos designados por el *desplazamiento* y el *tamano* de la matriz *entrada*, y los sustituye con los elementos de la matriz *sustitucion* si se especifica.

Si el *desplazamiento* es positivo, el comienzo de la parte suprimida sería en esa posición desde el comienzo de la matriz de *entrada*. Si el *desplazamiento* es negativo, se cuenta la posición desde el final de la matriz de *entrada*.

Si se omite *tamano*, se suprime todo desde el *desplazamiento* hasta el final de la matriz. Si se especifica el *tamano* y es positivo, se suprimirán tantos elementos como se especifica. Si fuera negativo, el final de la porción eliminada estará a tantos elementos del final de la matriz. Truco: para eliminar todo desde el *desplazamiento* hasta el final de la matriz cuando también se especifica *sustitucion*, utilice `count($entrada)` como *tamano*.

Si se especifica la matriz de *sustitucion*, entonces los elementos suprimidos son reemplazados con los elementos de dicha matriz. Si los valores de *desplazamiento* y *tamano* son tales que nada es borrado, los elementos de la matriz *sustitucion* se insertarán en la posición indicada por el *desplazamiento*. Truco: si sólo se va a sustituir algo por un elemento nada más, no hace falta poner `array()` alrededor del mismo, salvo que dicho elemento sea una matriz en sí mismo.

Las siguientes funciones son equivalentes:

<code>array_push(\$entrada, \$x, \$y)</code>	<code>array_splice(\$entrada, count(\$entrada), 0, array(\$x, \$y))</code>
<code>array_pop(\$entrada)</code>	<code>array_splice(\$entrada, -1)</code>
<code>array_shift(\$entrada)</code>	<code>array_splice(\$entrada, 0, 1)</code>
<code>array_unshift(\$entrada, \$x, \$y)</code>	<code>array_splice(\$entrada, 0, 0, array(\$x, \$y))</code>
<code>\$a[\$x] = \$y</code>	<code>array_splice(\$entrada, \$x, 1, \$y)</code>

Devuelve una matriz que tiene los elementos eliminados

Ejemplo 1. Ejemplos de array_splice()

```
$entrada = array("rojo", "verde", "azul", "amarillo");

array_splice($entrada, 2);           // $entrada vale ahora array("rojo", "verde")
array_splice($entrada, 1, -1);       // $entrada vale ahora array("rojo", "amarillo")
array_splice($entrada, 1, count($entrada), "naranja");
```

```
// $entrada vale ahora array("rojo", "naranja")
array_splice($entrada, -1, 1, array("negro", "marrón"));
// $entrada vale ahora array("rojo", "verde",
//                           "azul", "negro", "marrón")
```

Vea también: **array_slice()**.

Nota: Esta función fue añadida en el PHP 4.0.

array_unshift (PHP 4)

Introduce uno o más elementos al principio de la matriz

```
int array_unshift (array matriz, mixed var [, ...])
```

array_unshift() añade los elementos que se le pasan al principio de la *matriz*. Nótese que la lista de elementos es añadida como un todo, de modo que los elementos añadidos mantienen su orden.

Devuelve el número de elementos en la *matriz*.

Ejemplo 1. Ejemplo de array_unshift()

```
$cola = array("p1", "p3");
array_unshift($cola, "p4", "p5", "p6");
```

Esto hará que \$cola contenga 5 elementos: "p4", "p5", "p6", "p1", y "p3".

Vea también: **array_shift()**, **array_push()**, y **array_pop()**.

Nota: Esta función fue añadida en el PHP 4.0.

array_values (PHP 4)

Devuelve todos los valores de una matriz

```
array array_values (array entrada)
```

array_values() devuelve todos los valores de la matriz *entrada*.

Ejemplo 1. Ejemplo de array_values()

```
$matriz = array("talla" => "XL", "color" => "dorado");
array_values($matriz); // devuelve array("XL", "dorado")
```

Nota: Esta función fue añadida en el PHP 4.0.

array_walk (PHP 3>= 3.0.3, PHP 4)

Aplica una función del usuario a cada elemento de una matriz.

```
int array_walk (array matriz, string func, mixed datosvarios)
```

Aplica la función llamada *func* a cada elemento de la *matriz*. La función *func* recibirá el valor de la matriz como primer parámetro y la clave como segundo. Si se proporciona el parámetro *datosvarios* será pasado como tercer parámetro a la función de usuario.

Si *func* necesita más de dos o 3 argumentos, dependiendo de *datosvarios*, se generará un aviso cada vez que **array_walk()** llama a *func*. Estos avisos pueden suprimirse si se pone '@' antes de la llamada a **array_walk()**, o usando la función **error_reporting()**.

Nota: Si *func* precisa trabajar con los valores reales de la matriz, especifique que el valor del primer parámetro de *func* debe pasarse por referencia. Desde ese instante, los cambios realizados sobre dichos elementos también serán realizados en la propia matriz.

Nota: El pasar la clave y los datos de usuario a *func* fue una característica añadida en PHP 4.0.

En PHP 4 se debe llamar **reset()** las veces necesarias, pues **array_walk()** no reajusta la matriz por defecto.

Ejemplo 1. Ejemplo de array_walk()

```
$frutas = array ("d"=>"limón", "a"=>"naranja", "b"=>"plátano", "c"=>"manzana" );

function test_alterar (&$item1, $clave, $prefix) {
    $item1 = "$prefix: $item1";
}

function test_ver ($item2, $clave) {
    echo "$clave. $item2<br>\n";
}

array_walk ($frutas, 'test_ver');
reset ($frutas);
array_walk ($frutas, 'test_alterar', 'fruta');
reset ($frutas);
array_walk ($frutas, 'test_ver');
```

Vea también: **each()** y **list()**.

arsort (PHP 3, PHP 4)

Ordena una matriz en orden inverso y mantiene la asociación de índices

```
void arsort (array matriz)
```

Esta función ordena una matriz de modo que los índices mantengan su correlación con los elementos de la misma a los que están asociados. Esto se utiliza principalmente para ordenar matrices asociativas en las que el orden de los elementos es importante.

Ejemplo 1. Ejemplo de arsort()

```
$frutas = array ("d"=>"limón", "a"=>"naranja", "b"=>"plátano", "c"=>"manzana" );
arsort ($frutas);
for (reset ($frutas); $clave = key ($frutas); next ($frutas)) {
    echo "frutas[$clave] = ".$frutas[$clave]."\n";
}
```

Este ejemplo mostraría: frutas[b] = plátano frutas[a] = naranja frutas[c] = manzana frutas[d] = limón Las frutas han sido ordenadas en orden alfabético inverso y los índices asociados con cada elemento se han mantenido.

Vea también: **asort()**, **rsort()**, **ksort()**, y **sort()**.

asort (PHP 3, PHP 4)

Ordena una matriz y mantiene la asociación de índices

```
void asort (array matriz)
```

Esta función ordena una matriz de modo que los índices mantengan su correlación con los elementos de la misma a los que están asociados. Esto se utiliza principalmente para ordenar matrices asociativas en las que el orden de los elementos es importante.

Ejemplo 1. Ejemplo de asort()

```
$frutas = array ("d"=>"limón", "a"=>"naranja", "b"=>"plátano", "c"=>"manzana" );
asort ($frutas);
for (reset ($frutas); $clave = key ($frutas); next ($frutas)) {
    echo "frutas[$clave] = ".$frutas[$clave]."\n";
}
```

Este ejemplo mostrará: frutas[d] = limón frutas[a] = naranja frutas[c] = manzana frutas[b] = plátano Las frutas han sido ordenadas en orden alfabético y los índices asociados con cada elemento se han mantenido.

Vea también: **arsort()**, **rsort()**, **ksort()**, y **sort()**.

compact (PHP 4)

Crea una matriz que contiene variables y sus valores

```
array compact (string nombrevar / array nombrevars [, ...])
```

compact() toma un número variable de parámetros. Cada uno puede ser tanto una cadena que contiene el nombre de la variable, como una matriz de nombres de variable. La matriz puede contener otras matrices de nombres de variable en su interior; **compact()** los procesa recursivamente.

Para cada uno de estos, **compact()** busca una variable con dicho nombre en la tabla de símbolos y la añade a la matriz de salida de modo que el nombre de la variable es la clave y el contenido de ésta es el valor para dicha clave. Para resumir, hace lo contrario de **extract()**. Devuelve la matriz de salida con las variables añadidas a la misma.

Ejemplo 1. Ejemplo de compact()

```
$ciudad = "San Francisco";
$estado = "CA";
$evento = "SIGGRAPH";

	location_vars = array ("ciudad", "estado");

$resultado = compact ("evento", $location_vars);
```

Tras esto, \$resultado valdrá array ("evento"=> "SIGGRAPH", "ciudad"=> "San Francisco", "estado"=> "CA").

Vea también: **extract()**.

Nota: Esta función fue añadida en el PHP 4.0.

count (PHP 3, PHP 4)

Cuenta los elementos de una variable

```
int count (mixed var)
```

Devuelve el número de elementos en *var*, que típicamente es una matriz (porque cualquier otra cosa tendría sólo un elemento).

Devuelve 1 si la variable no es una matriz.

Devuelve 0 si la variable no tiene valor.

Aviso

count() puede devolver 0 para una variable sin valor, pero también puede devolver 0 para una variable ya inicializada pero con una matriz vacía. Utilice **isset()** para comprobar si una variable está inicializada.

Vea también: **sizeof()**, **isset()**, y **is_array()**.

current (PHP 3, PHP 4)

Devuelve el elemento actual de una matriz

```
mixed current (array matriz)
```

Cada matriz tiene un puntero interno al elemento "actual", que se inicializa al primer elemento insertado en la misma.

La función **current()** simplemente devuelve el elemento de la tabla al que apunta el puntero interno. No mueve el puntero de ninguna manera. Si el puntero interno apunta fuera del final de la lista de elementos, **current()** devuelve false.

Aviso

Si la matriz contiene elementos vacíos (0 ó "", la cadena vacía) esta función devolverá false también para dichos elementos. Esto hace imposible determinar si se está realmente al final de la lista en tales matrices usando **current()**. Para recorrer adecuadamente una matriz que pueda contener elementos vacíos, utilice la función **each()**.

Vea también: **end()**, **next()**, **prev()** y **reset()**.

each (PHP 3, PHP 4)

Devuelve el siguiente par clave/valor de una matriz

```
array each (array matriz)
```

Devuelve el par clave/valor actual para la *matriz* y avanza el cursor de la misma. Esta pareja se devuele en una matriz de 4 elementos, con las claves *0*, *1*, *key*, y *value*. Los elementos *0* y *key* contienen el nombre de clave del elemento de la matriz, y *1* y *value* contienen los datos.

Si el puntero interno para la matriz apunta pasado el final del contenido de la matriz, **each()** devuelve false.

Ejemplo 1. Ejemplos de each()

```
$chorrada = array ("bob", "fred", "jussi", "jouni", "egon", "marliese");
$tonteria = each ($chorrada);
```

\$tonteria contiene ahora los siguientes pares clave/valor:

- 0 => 0
- 1 => 'bob'
- key => 0
- value => 'bob'

```
$chorrada = array ("Robert" => "Bob", "Seppo" => "Sepi");
$tonteria = each ($chorrada);
```

\$tonteria contiene ahora los siguientes pares clave/valor:

- 0 => 'Robert'
- 1 => 'Bob'
- key => 'Robert'
- value => 'Bob'

each() se usa normalmente de forma conjunta a **list()** para recorrer una matriz; por ejemplo, **\$HTTP_POST_VARS**:

Ejemplo 2. Recorriendo **\$HTTP_POST_VARS** con each()

```
echo "Valores enviados con el método POST:<br>" ;
```

```
reset ($HTTP_POST_VARS);
while (list ($clave, $val) = each ($HTTP_POST_VARS)) {
    echo "$clave => $val<br>";
}
```

Cuando se ha ejecutado **each()**, el cursor de la matriz quedará en el siguiente elemento de la misma, o en el último si llega al final de ésta.

Vea también: **key()**, **list()**, **current()**, **reset()**, **next()**, y **prev()**.

end (PHP 3, PHP 4)

Mueve el puntero interno de una tabla al último elemento

```
end (array matriz)
```

end() avanza el puntero interno de la *matriz* al último elemento.

Vea también: **current()**, **each()**, **end()**, **next()**, y **reset()**.

extract (PHP 3>= 3.0.7, PHP 4)

Importa variables a la tabla de símbolos desde una matriz

```
void extract (array matriz_vars [, int tipo_extraccion [, string prefijo]])
```

Esta función se utiliza para importar variables desde una matriz a la tabla de símbolos actual. Toma la matriz asocialiva *matriz_vars* y trata las claves como nombres de variable y los valores como los valores de éstas. Para cada par clave/valor creará una variable en la tabla de símbolos actual, sujeto a los parámetros *tipo_extraccion* y *prefijo*.

extract() controla las colisiones con las variables que ya existen. La forma de tratar éstas se determina por el *tipo_extraccion*. Puede tener únicamente uno de los siguientes valores:

EXTR_OVERWRITE

Si hay colisión, sobreescribe la variable existente.

EXTR_SKIP

Si hay colisión, no sobreescibas la variable existente.

EXTR_PREFIX_SAME

Si hay una colisión, añade el *prefijo* a la nueva variable.

EXTR_PREFIX_ALL

Añade el *prefijo* a todas las variables.

Si no se especifica *tipo_extraccion*, se asume que vale EXTR_OVERWRITE.

Nótese que el *prefijo* sólo se necesita si *tipo_extraccion* vale EXTR_PREFIX_SAME o EXTR_PREFIX_ALL.

extract() comprueba si cada clave es un nombre válido de variable, y sólo lo importa si lo es.

Nota: N.T.: En el caso español, no valdría "año" como nombre variable (pero sí como clave en una matriz cualquiera).

Un uso posible para extract sería importar en la tabla de símbolos las variables contenidas en la matriz asociativa que devuelve **wddx_deserialize()**.

Ejemplo 1. Ejemplo de Extract()

```
<php?
/* Suponemos que $matriz_var es una matriz devuelta por
wddx_deserialize */

$tamano = "grande";
$matriz_var = array ("color" => "azul",
                     "tamano" => "media",
                     "forma" => "esfera");
extract ($matriz_var, EXTR_PREFIX_SAME, "wddx");

print "$color, $tamano, $forma, $wddx_tamano\n";

?>
```

El programa anterior producirá:

azul, grande, esfera, media

La variable \$tamano no fue sobreescrita porque especificamos EXTR_PREFIX_SAME, que provocó la creación de \$wddx_tamano. Si se hubiera especificado EXTR_SKIP, \$wddx_tamano ni siquiera habría sido creada. EXTR_OVERWRITE habría provocado que \$tamano tuviera el valor "media", y EXTR_PREFIX_ALL habría provocado que aparecieran nuevas variables llamadas \$wddx_color, \$wddx_tamano, y \$wddx_forma.

in_array (PHP 4)

Devuelve true si un valor está en una matriz

```
bool in_array (mixed aguja, array pajar)
```

Busca la *aguja* en el *pajar*, y devuelve true si se encuentra y false en caso contrario.

Ejemplo 1. Ejemplo de in_array()

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

Nota: Esta función fue añadida en el PHP 4.0.

key (PHP 3, PHP 4)

Obtiene una clave de una matriz asociativa

```
mixed key (array matriz)
```

key() devuelve el elemento índice de la posición actual en la matriz.

Vea también: **current()**, **next()**

krsort (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Ordena una matriz por clave en orden inverso

```
int krsort (array matriz)
```

Ordena una matriz por clave en orden inverso, manteniendo las correlaciones clave a dato. Esto es útil principalmente en matrices asociativas.

Ejemplo 1. Ejemplo de krsort()

```
$frutas = array ("d"=>"limón", "a"=>"naranja", "b"=>"plátano", "c"=>"manzana");
krsort ($frutas);
for (reset ($frutas); $clave = key ($frutas); next ($frutas)) {
    echo "frutas[$clave] = ".$frutas[$clave]."\n";
}
```

Este ejemplo mostrará: frutas[d] = limón frutas[c] = manzana frutas[b] = plátano frutas[a] = naranja

Vea también: **asort()**, **arsort()**, **ksort()** **sort()**, y **rsort()**.

ksort (PHP 3, PHP 4)

Ordena una matriz por clave

```
int ksort (array matriz)
```

Ordena una matriz por clave, manteniendo las correlaciones clave a dato. Esto es útil principalmente en matrices asociativas.

Ejemplo 1. Ejemplo de ksort()

```
$frutas = array ("d"=>"limón", "a"=>"naranja", "b"=>"plátano", "c"=>"manzana");
ksort ($frutas);
for (reset ($frutas); $clave = key ($frutas); next ($frutas)) {
    echo "frutas[$clave] = ".$frutas[$clave]."\n";
}
```

Este ejemplo mostrará: frutas[a] = naranja frutas[b] = plátano frutas[c] = manzana frutas[d] = limón

Vea también: **asort()**, **arsort()**, **sort()**, y **rsort()**.

list (unknown)

Asigna variables como si fueran una matriz

```
void list(...);
```

Como **array()**, esta no es realmente una función, sino una construcción del lenguaje. **list()** se usa para asignar una lista de variables en una sola operación.

Ejemplo 1. Ejemplo de list()

```
<table>
<tr>
  <th>Nombre empleado</th>
  <th>Sueldo</th>
</tr>

<?php

$resultado = mysql($conn, "SELECT id, nombre, salario FROM empleados");
while (list($id, $nombre, $salario) = mysql_fetch_row($resultado)) {
    print(" <tr>\n".
          "   <td><a href=\"info.php?id=$id\">$nombre</a></td>\n".
          "   <td>$salario</td>\n".
          "   </tr>\n");
}

?>

</table>
```

Vea también: **each()**, **array()**.

next (PHP 3, PHP 4)

Avanza el puntero interno de una matriz

```
mixed next (array matriz)
```

Devuelve el elemento de la matriz que ocupa el lugar siguiente al apuntado por el puntero interno, o false si no hay más elementos.

next() se comporta como **current()**, con una diferencia. Avanza el puntero interno de la matriz en una posición antes de devolver el elemento. Eso significa que devuelve el siguiente elemento de la matriz y que avanza el puntero interno en uno. Si al avanzar se pasa del final de la lista de elementos, **next()** devuelve false.

Aviso

Si la matriz contiene elementos vacíos, esta función también devolverá false para dichos elementos. Para recorrer adecuadamente una matriz que pueda contener elementos vacíos, vea la función **each()**.

Vea también: **current()**, **end()** **prev()** y **reset()**

pos (PHP 3, PHP 4)

Obtiene el elemento actual de una matriz

```
mixed pos (array matriz)
```

Este es un alias para **current()**.

Vea también: **end()**, **next()**, **prev()** y **reset()**.

prev (PHP 3, PHP 4)

Rebobina el puntero interno de una matriz

```
mixed prev (array matriz)
```

Devuelve el elemento de la matriz que está en la posición anterior a la que apuntaba previamente el puntero interno, o false si no hay más elementos.

Aviso

Si la matriz contiene elementos vacíos, esta función también devolverá false para dichos elementos. Para recorrer adecuadamente una matriz que puede contener elementos vacíos, vea la función **each()**.

prev() se comporta igual que **next()**, excepto que rebobina el puntero interno una posición en lugar de avanzarlo.

Vea también: **current()**, **end()** **next()** y **reset()**

rango (unknown)

Crea una matriz que contiene un rango de enteros

```
array rango (int bajo, int alto)
```

rango() devuelve una matriz de enteros desde *bajo* hasta *alto*, ambos inclusive.

Vea un ejemplo de su uso en la función **shuffle()**.

reset (PHP 3, PHP 4)

Fija el puntero interno de una matriz a su primer elemento

```
mixed reset (array matriz)
```

reset() rebobina el puntero interno de la *matriz* a su primer elemento.

reset() devuelve el valor del primer elemento de la matriz.

Vea también: **current()**, **each()**, **next()**, **prev()**, y **reset()**.

rsort (PHP 3, PHP 4)

Ordena una matriz en orden inverso

```
void rsort (array matriz)
```

Esta función ordena una matriz en orden inverso (mayor a menor).

Ejemplo 1. Ejemplo de rsort()

```
$frutas = array ("limón", "naranja", "plátano", "manzana");
rsort ($frutas);
for (reset ($frutas); list ($clave, $valor) = each ($frutas); ) {
    echo "frutas[$clave] = ", $valor, "\n";
}
```

Este ejemplo mostrará: frutas[0] = plátano frutas[1] = naranja frutas[2] = manzana frutas[3] = limón Las frutas han sido ordenadas en orden alfabético inverso.

Vea también: **arsort()**, **asort()**, **ksort()**, **sort()**, y **usort()**.

shuffle (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Mezcla una matriz

```
void shuffle (array matriz)
```

Esta función mezcla (cambia aleatoriamente el orden de los elementos de) una matriz.

Ejemplo 1. Ejemplo de shuffle()

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

Vea también: **arsort()**, **asort()**, **ksort()**, **rsort()**, **sort()** y **usort()**.

sizeof (PHP 3, PHP 4)

Obtiene el número de elementos de una matriz

```
int sizeof (array matriz)
```

Devuelve el número de elementos de la matriz.

Vea también: **count()**

sort (PHP 3, PHP 4)

Ordena una matriz

```
void sort (array matriz)
```

Esta función ordena una matriz. Los elementos estarán ordenados de menor a mayor cuando la función termine.

Ejemplo 1. Ejemplo de sort()

```
$frutas = array ("limón", "naranja", "plátano", "manzana");
sort ($frutas);
for (reset ($frutas); $clave = key ($frutas); next ($frutas)) {
    echo "frutas[$clave] = ".$frutas[$clave]."\n";
}
```

Este ejemplo mostrará: frutas[0] = limón frutas[1] = manzana frutas[2] = naranja frutas[3] = plátano Las frutas han sido ordenadas en orden alfabético.

Vea también: **arsort()**, **asort()**, **ksort()**, **rsort()**, y **usort()**.

uasort (PHP 3>= 3.0.4, PHP 4)

Ordena una matriz mediante una función de comparación definida por el usuario y mantiene la asociación de índices

```
void uasort (array matriz, function func_comparar)
```

Esta función ordena una matriz de modo que los índices de la misma mantengan su correlación con los elementos a los que están asociados. Esto se utiliza principalmente para ordenar matrices asociativas en las que el orden de los elementos es importante. La función de comparación viene definida por el usuario.

uksort (PHP 3>= 3.0.4, PHP 4)

Ordena una matriz por claves mediante una función definida por el usuario

```
void uksort (array matriz, function func_comparar)
```

Esta función ordenará las claves de una matriz utilizando una función de comparación suministrada por el usuario. Si la matriz a ordenar necesita utilizar un criterio poco trivial, esta es la función que deberá usar.

Ejemplo 1. Ejemplo de uksort()

```
function micomparar ($a, $b) {
    if ($a == $b) return 0;
    return ($a > $b) ? -1 : 1;
}
$a = array (4 => "cuatro", 3 => "tres", 20 => "veinte", 10 => "diez");
uksort ($a, micomparar);
while (list ($clave, $valor) = each ($a)) {
    echo "$clave: $valor\n";
}
```

Este ejemplo mostrará: 20: veinte 10: diez 4: cuatro 3: tres

Vea también: **arsort()**, **asort()**, **uasort()**, **ksort()**, **rsort()**, y **sort()**.

usort (PHP 3>= 3.0.3, PHP 4)

Ordena una matriz por valores mediante una función definida por el usuario

```
void usort (array matriz, function func_comparar)
```

Esta función ordenará una matriz por sus valores utilizando una función suministrada por el usuario. Si la matriz que desea ordenar necesita utilizar un criterio poco trivial, esta es la función que deberá usar.

La función de comparación deberá devolver un entero menor, igual, o mayor que cero, si el primer argumento se considera respectivamente menor que, igual que, o mayor que el segundo. Si dos miembros resultan ser iguales, su orden en la matriz ordenada será cualquiera.

Ejemplo 1. Ejemplo de 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 ($clave, $valor) = each ($a)) {
    echo "$clave: $valor\n";
}
```

Este ejemplo mostrará:

```
0: 6 1: 5 2: 3 3: 2 4: 1
```

Nota: Obviamente en este caso trivial la función **rsort()** habría sido más apropiada.

Aviso

La función quicksort subyacente en ciertas librerías de C (tales como las de Solaris) pueden hacer que el PHP falle si la función de comparación no devuelve valores consistentes.

Vea también: **arsort()**, **asort()**, **uasort()**, **ksort()**, **rsort()** y **sort()**.

III. Funciones Ortográficas

Las funciones **aspell()** permiten comprobar la ortografía de una palabra ofreciéndote sugerencias..

Para estas funciones, son necesarias las librerías aspell (ortográficas) disponibles en <http://metalab.unc.edu/kevina/aspell/>

aspell_new (PHP 3>= 3.0.7, PHP 4)

Lee un nuevo diccionario

```
int aspell_new (string master, string personal)
```

aspell_new() Abre un nuevo diccionario devolviendo el identificador de este para ser utilizado en otras funciones ortográficas.

Ejemplo 1. Nuevo_diccionario

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

aspell_check (PHP 3>= 3.0.7, PHP 4)

Comprueba una palabra

```
boolean aspell_check (int dictionary_link, string word)
```

aspell_check() comprueba la ortografía de una palabra, y devuelve cierto(True) si la ortografía es correcta ,falso (False) si no lo es .

Ejemplo 1. aspell_check

```
$aspell_link=aspell_new("english");
if (aspell_check($aspell_link,"testt")) {
    echo "Está bien escrita";
} else {
    echo "Lo siento, está mal escrita";
}
```

aspell_check-raw (PHP 3>= 3.0.7, PHP 4)

Comprueba una palabra sin cambiarla o intentar arreglarla

```
boolean aspell_check_raw (int dictionary_link, string word)
```

aspell_check_raw() chequea la ortografía de una palabra,sin cambiarla ni intentar arreglarla esté bien o mal.Si está bien, devuelve cierto (True), si no lo está, devuelve falso(False).

Ejemplo 1. aspell_check_raw

```
$aspell_link=aspell_new("english");
if (aspell_check_raw($aspell_link,"testt")) {
    echo "Está bien escrito";
} else {
```

```
    echo "Lo siento, mal escrito";
}
```

aspell_suggest (PHP 3>= 3.0.7, PHP 4)

sugiere la ortografía para una palabra

```
array aspell_suggest (int dictionary_link, string word)
```

aspell_suggest() devuelve un array con posibles correcciones ortográficas para la palabra dada.

Ejemplo 1. aspell_suggest

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

if (!aspell_check($aspell_link,"testt")) {
    $sugerencias=aspell_suggest($aspell_link,"testt");

    for($i=0; $i < count($sugerencias); $i++) {
        echo "Posibles palabras: " . $sugerencias[$i] . "<br>";
    }
}
```

IV. Funciones matemáticas de precisión arbitraria

Estas funciones sólo están disponibles si el PHP se configuró con `-enable-bcmath`.

bcadd (PHP 3, PHP 4)

Suma dos números de precisión arbitraria.

```
string bcadd (string operando izq, string operando der [, int escala])
```

Suma el *operando izq* con el *operando der* y devuelve la suma en una string. El parámetro opcional *escala* se usa para fijar el número de dígitos tras el punto decimal que aparecerán en el resultado.

Vea también **bbsub()**.

bccomp (PHP 3, PHP 4)

Compara dos números de precisión arbitraria.

```
int bccomp (string operando izq, string operando der [, int escala])
```

Compara el *operando izq* con el *operando der* y devuelve el resultado como un entero. El parámetro opcional *escala* se usa para fijar el número de dígitos tras el punto decimal que se utilizarán en la comparación. El valor devuelto es 0 si los dos operandos son iguales. Si el *operando izq* es mayor que el *operando der* el valor devuelto es +1 y si el *operando izq* es menor que el *operando der* el valor devuelto es -1.

bcdif (PHP 3, PHP 4)

Divide dos números de precisión arbitraria.

```
string bcdif (string operando izq, string operando der [, int escala])
```

Divide el *operando izq* por el *operando der* y devuelve el resultado. El parámetro opcional *escala* fija el número de dígitos tras el punto decimal a usar en el resultado.

Vea también **bcmul()**.

bcmod (PHP 3, PHP 4)

Obtiene el módulo de un número de precisión arbitraria.

```
string bcmod (string operando izq, string modulo)
```

Obtiene el módulo del *operando izq* usando *modulo*.

Vea también **bcdif()**.

bcmul (PHP 3, PHP 4)

Multiplica dos números de precisión arbitraria.

```
string bcmul (string operando izq, string operando der [, int escala])
```

Multiplica el *operando izq* por el *operando der* y devuelve el resultado. El parámetro opcional *escala* fija el número de dígitos tras el punto decimal del resultado.

Vea también **bcddiv()**.

bcpow (PHP 3, PHP 4)

Eleva un número de precisión arbitraria a otro.

```
string bcpow (string x, string y [, int escala])
```

Eleva *x* a la potencia de *y*. El parámetro opcional *escala* se puede usar para fijar el número de dígitos tras el punto decimal del resultado.

Vea también **bcsqrt()**.

bcscale (PHP 3, PHP 4)

Fija el parámetro de escala por defecto para todas las funciones matemáticas bc.

```
string bcsetScale (int escala)
```

Esta función fija el parámetro de escala por defecto para las subsiguientes funciones matemáticas bc que no especifican dicho parámetro explícitamente.

bcsqrt (PHP 3, PHP 4)

Obtiene la raíz cuadrada de un número de precisión arbitraria.

```
string bcsqrt (string operando, int escala)
```

Devuelve la raíz cuadrada del *operando*. El parámetro opcional *escala* fija el número de dígitos tras el punto decimal del resultado.

Vea también **bcpow()**.

bcsub (PHP 3, PHP 4)

Resta un número de precisión arbitraria de otro.

```
string bcsub (string operando izq, string operando der [, int escala])
```

Resta el *operando der* desde el *operando izq* y devuelve el resultado en una cadena. El parámetro opcional *escala* se utiliza para fijar el número de dígitos tras el punto decimal del resultado.

Vea también **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[=DIR]` 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.

Ejemplo 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.

Ejemplo 1. bzcompress() Example

```
$str = "sample data";
$bzstr = bzcompress($str, 9);
```

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.

Ejemplo 1. bzdecompress()

```
$str = $bzdecompress($bzstr);
```

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*.

Ejemplo 1. **bzerror()** Example

```
$error = bzerror($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.

Ejemplo 1. bzopen() Example

```
$bz = bzopen("/tmp/foo.bz2", "r");
```

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.

Ejemplo 1. bzread() Example

```
$bz = bzopen("/tmp/foo.bz2", "r");
$str = bzread($bz, 2048);
echo $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.

Ejemplo 1. bzwrite() Example

```
$str = "uncompressed data";
$bz = bzopen("/tmp/foo.bz2", "w");
bzwrite($bz, $str, strlen($str));
```

See also **bzread()** and **bzopen()**.

VI. Funciones de calendario

Las funciones de calendario sólo están disponibles si ha compilado la extensión de calendario que hay en `dl/calendar`. Lea el documento `dl/README` como referencia de uso.

La extensión `calendar` presenta una serie de funciones para simplificar la conversión entre los distintos formatos de calendario. El intermediario estándar en que se basa es en la Cuenta de Días Juliana. La Cuenta de Días Juliana es una cuenta que comienza mucho antes que lo que mucha gente podría necesitar contar (como alrededor del 4000 AC). Para convertir entre sistemas de calendario, primero deberá convertir a la Cuenta de Días Juliana y luego al sistema de su elección. ¡La Cuenta de Días es muy diferente del Calendario Juliano! Para más información sobre sistemas de calendario, visite <http://genealogy.org/~scottlee/cal-overview.html>. En estas instrucciones se han incluído extractos entrecorbillados de dicha página.

JDToGregorian (PHP 3, PHP 4)

Convierte de Cuenta de Días a fecha Gregoriana

```
string jdtogregorian (int diajuliano)
```

Convierte de Cuenta de Días Juliana a una cadena que contiene la fecha Gregoriana en formato "mes/día/año"

GregorianToJD (PHP 3, PHP 4)

Convierte de fecha Gregoriana a Cuenta de Días

```
int gregoriantojd (int mes, int dia, int anno)
```

El rango válido para el Calendario Gregoriano es desde el 4714 A.C. hasta el 9999 D.C.

Aunque este programa puede manejar fechas tan lejanas como el 4714 A.C., usarlo no tendría sentido. El calendario Gregoriano fue instituído el 15 de octubre de 1582 (o el 5 de octubre de 1582 en el calendario Julian). Algunos países no lo aceptaron hasta mucho después. Por ejemplo, Gran Bretaña se convirtió en 1752, la URSS en 1918 y Grecia en 1923. Muchos países europeos usaron el calendario Julian antes que el Gregoriano.

Ejemplo 1. Funciones de calendario

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

JDToJulian (PHP 3, PHP 4)

Convierte de Cuenta de Días a Calendario Juliano

```
string jdtojulian (int diajuliano)
```

Convierte una Cuenta de Días Julian a una cadena que contiene la fecha del Calendario Juliano en formato "mes/día/año".

JulianToJD (PHP 3, PHP 4)

Convierte de Calendario Juliano a Cuenta de Días

```
int juliantojd (int mes, int dia, int anno)
```

Rango válido para el Calendario Juliano: del 4713 A.C al 9999 D.C.

Aunque este programa puede manejar fechas tan lejanas como el 4713 A.C., usarlo no tendría sentido. El calendario se creó en el 46 A.C., pero sus detalles no se estabilizaron hasta al menos el 8 D.C., y quizás no lo hiciera hasta el siglo IV. Además, el comienzo de un año variaba de una a otra cultura: no todas aceptaban enero como el primer mes.

JDToJewish (PHP 3, PHP 4)

Convierte de Cuenta de Días a Calendario Judío

```
string jdtojewish (int diajuliano)
```

Convierte una Cuenta de Días Juliana al Calendario Judío.

JewishToJD (PHP 3, PHP 4)

Convierte del Calendario Judío a la Cuenta de Días

```
int jewishtojd (int mes, int dia, int anno)
```

El rango válido para el Calendario Judío va del año 1 hasta el 9999

Aunque este programa puede manejar fechas tan lejanas como el año 1 (3761 A.C.), usarlo no tendría sentido. El Calendario Judío ha estado en uso miles de años, pero en los días primeros no había una fórmula que calculara el comienzo de un mes. Un mes comenzaba cuando se veía por primera vez la luna nueva.

JDToFrench (PHP 3, PHP 4)

Convierte de Cuenta de Días al Calendario Republicano Francés

```
string jdtotfrench (int diajuliano)
```

Convierte una Cuenta de Días Juliana al Calendario Republicano Francés.

FrenchToJD (PHP 3, PHP 4)

Convierte del Calendario Republicano Francés a la Cuenta de Días

```
int frenchtojd (int mes, int dia, int anno)
```

Convierte una fecha del Calendario Republicano Francés a la Cuenta de Días Juliana.

Estas rutinas sólo convierten fechas entre los años 1 y 14 (fechas Gregorianas del 22 de septiembre de 1792 al 22 de septiembre de 1806). Esto cubre ampliamente el periodo en el que estuvo en uso este calendario.

JDMonthName (PHP 3, PHP 4)

Devuelve el nombre de un mes

```
string jdmonthname (int diajuliano, int modo)
```

Devuelve una cadena que contiene el nombre del mes. *modo* le dice a esta función a qué calendario debe convertir la Cuenta de Días Juliana, y qué tipo de nombres de mes debe devolver.

Tabla 1. Modos de calendario

Modo	Significado
0	Gregoriano - abreviado
1	Gregoriano
2	Juliano - abreviado
3	Juliano
4	Judío
5	Republicano Francés

JDDayOfWeek (PHP 3, PHP 4)

Devuelve el día de la semana

```
mixed jddayofweek (int diajuliano, int modo)
```

Devuelve el día de la semana. Dependiendo del modo, devuelve un entero o una cadena.

Tabla 1. Modos para el día de la semana

Modo	Significado
0	devuelve el día de la semana como entero (0=domingo, 1=lunes, etc)
1	devuelve una cadena con el día de la semana (inglés, gregoriano)
2	devuelve una cadena con el día de la semana abreviado (inglés, gregoriano)

easter_date (PHP 3>= 3.0.9, PHP 4 >= 4.0RC2)

devuelve la marca de tiempo UNIX para la medianoche de Pascua de un año dado

```
int easter_date (int anno)
```

Devuelve la marca de tiempo UNIX que corresponde a la medianoche de Pascua del año dado. Si no se especifica un año, se asume el actual.

Aviso: Esta función generará un aviso si el año está fuera del rango para las marcas de tiempo del UNIX (es decir, antes de 1970 o después del 2037).

Ejemplo 1. ejemplo de easter_date()

```

echo date( "d-M-Y", easter_date(1999) );           /* "04-Apr-1999" */
echo date( "d-M-Y", easter_date(2000) );           /* "23-Apr-2000" */
echo date( "d-M-Y", easter_date(2001) );           /* "15-Apr-2001" */

```

La fecha del Día de Pascua fue definida por el Concilio de Nicea en el 325 D.C. como el domingo tras la primera luna llena que cayera en o después del equinoccio de Primavera. El equinoccio se supone que siempre cae en el 21 de marzo, de modo que el cálculo se reduce a determinar la fecha de la luna llena y la del domingo siguiente. El algoritmo usado aquí fue introducido en el año 532 por Dionisio Exiguo. Bajo el Calendario Juliano (para años anteriores al 1753), se usa un ciclo simple de 19 años para calcular las fases de la luna. Bajo el Calendario Gregoriano (años posteriores al 1753, diseñado por Clavio y Lilio, e introducido por el Papa Gregorio XIII en Octubre de 1582, y en Gran Bretaña y sus colonias en septiembre de 1752) se añaden dos factores de corrección para hacer el ciclo más preciso.

(El código se basa en un programa en C de Simon Kershaw, <webmaster@ely.anglican.org>)

Vea **easter_days()** para calcular la Pascua antes del 1970 o después del 2037.

easter_days (PHP 3>= 3.0.9, PHP 4 >= 4.0RC2)

obtiene el número de días tras el 21 de marzo en que cae la Pascua en un año dado

```
int easter_days (int anno)
```

Devuelve el número de días tras el 21 de marzo en que cae la Pascua en un año dado. Si no se especifica año, se asume el actual.

Esta función se puede usar en lugar de **easter_date()** para calcular la Pascua para años que se salen del rango de las marcas de fecha del UNIX (o sea, antes del 1970 o después del 2037).

Ejemplo 1. ejemplo de easter_date()

```

echo easter_days(1999);      /* 14, es decir, 4 de abril */
echo easter_days(1492);      /* 32, es decir, 22 de abril */
echo easter_days(1913);      /* 2, es decir, 23 de marzo */

```

La fecha del Día de Pascua fue definida por el Concilio de Nicea en el 325 D.C. como el domingo tras la primera luna llena que cayera en o después del equinoccio de Primavera. El equinoccio se supone que siempre cae en el 21 de marzo, de modo que el cálculo se reduce a determinar la fecha de la luna llena y la del domingo siguiente. El algoritmo usado aquí fue introducido en el año 532 por Dionisio Exiguo. Bajo el Calendario Juliano (para años anteriores al 1753), se usa un ciclo simple de 19 años para calcular las fases de la luna. Bajo el Calendario Gregoriano (años posteriores al 1753, diseñado por Clavio y Lilio, e introducido por el Papa Gregorio XIII en Octubre de 1582, y en Gran Bretaña y sus colonias en septiembre de 1752) se añaden dos factores de corrección para hacer el ciclo más preciso.

(El código se basa en un programa en C de Simon Kershaw, <webmaster@ely.anglican.org>)

Vea también **easter_date()**.

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. soporte de las funciones COM para Windows

Estas funciones solo están disponibles en la versión para Windows de PHP. Estas funciones han sido añadidas en PHP4.

com_load (PHP 3>= 3.0.3, PHP 4 <= 4.0.4)

???

```
string com_load (string module name [, string server name])
```

com_invoke (PHP 3>= 3.0.3, PHP 4 <= 4.0.4)

???

```
mixed com_invoke (resource object, string function_name [, mixed function parameters, ...])
```

com_propget (PHP 3>= 3.0.3, PHP 4 <= 4.0.4)

???

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

com_get (PHP 3>= 3.0.3, PHP 4 <= 4.0.4)

???

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

com_propput (PHP 3>= 3.0.3, PHP 4 <= 4.0.4)

???

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

com_propset (PHP 3>= 3.0.3, PHP 4 <= 4.0.4)

???

```
void com_propset (resource object, string property, mixed value)
```

Esta función es un alias para **com_propput()**.

com_set (PHP 3>= 3.0.3, PHP 4 <= 4.0.4)

???

```
void com_set (resource object, string property, mixed value)
```

Esta función es un alias para **com_set()**.

IX. Funciones de Clases/Objectos

get_class_methods (PHP 4 >= 4.0RC1)

Devuelve un vector (matriz unidimensional) con los nombres de los métodos de la clase en question.

```
vector get_class_methods (string class_name)
```

Esta función devuelve un vector con los nombres de los métodos definidos en la clase especificada como *class_name*.

get_class_vars (PHP 4 >= 4.0RC1)

Devuelve un vector con las propiedades (inicializadas por defecto) de la clase

```
array get_class_vars (string class_name)
```

Esta función devuelve un vector con las propiedades que han sido inicializadas por defecto en la clase.

get_object_vars (PHP 4 >= 4.0RC1)

Devuelve un vector de propiedades del objeto

```
array get_class_vars (object obj)
```

Esta función devuelve un vector con las propiedades de objeto definidas en el objeto especificado como *obj*.

method_exists (PHP 4 >= 4.0b2)

Comprueba que el método de clase existe

```
bool method_exists (object object, string method_name)
```

Esta función devuelve verdadero si el método referido por *method_name* ha sido definido en el objeto *object*, en cualquier otro caso devuelve falso

X. Funciones de ClibPDF

ClibPDF Le permite crear documentos PDF con PHP. Está disponible en FastIO (<http://www.fastio.com>) pero no es software libre. Debería leer la licencia antes de comenzar a utilizar ClibPDF. Si usted no puede cumplir el acuerdo de la licencia considere el utilizar la pdflib de Thomas Merz, que tambien es muy potente. La funcionalidad y la API de ClibPDF son similares a la pdflib de Thomas Merz pero, de acuerdo con FastIO, ClibPDF es mas rápida y crea documentos mas pequeños. Esto puede haber cambiado con la nueva versión 2.0 de pdflib. Un simple banco de pruebas (el ejemplo pdfclock.c de pdflib 2.0 trasformado en un script php) en realidad no muestra ninguna diferencia en velocidad. Por tanto, pruebe las dos y vea cual hace el mejor trabajo para usted.

Esta documentación debería ser leída junto con el manual de ClibPDF ya que este explica la librería con mucho mas detalle.

Muchas funciones en le ClibPDF nativa y el módulo PHP, así como en pdflib, tienen el mismo nombre. Todas las funciones excepto **cpdf_open()** toman el manejador del documento com el primer parámetro. Actualmente este manejador no se usa internamente desde que ClibPDF no soporta la creación de varios documentos PDF al mismo tiempo. Realmente, ni debería intentarlo, los resultados son impredecibles. No puedo supervisar cuales son las consecuencias en un sistema multihilo. De acuerdo con el autor de ClibPDF, esto cambiará en alguno de las próximas veriones (la versión actual, cuando eto fue escrito es 1.10). Si usted necesita esta capacidad, use el módulo pdflib.

Nota: La función **cpdf_set_font()** ha cambiado desde que PHP3 soporta fuentes asiáticas. El parámetro que codifica ya no es un entero sino una cadena.

Una gran ventaja de ClibPDF sobre pdflib es la posibilidad de crear el documento PDF completamente en memoria sin usar ficheros temporales. Esto también proporciona la capaciad de pasar coordenadas en una unidad de longitud predefinida. Esta es una cualidad útil pero puede ser simulada con **pdf_translate()**.

La mayoría de las funciones son fáciles de usar. La parte mas difícil es, probablemente, crear un documento PDF muy simple. El siguiente ejemplo debería ayudarle a comenzar. En él se crea un documento con una página. La página contiene el texto "Times-Roman"con una fuente de 30pt. El texto está subrayado.

Ejemplo 1. Ejemplo simple de ClibPDF

```
<?php
$cpdf = cpdf_open(0);
cpdf_page_init($cpdf, 1, 0, 595, 842);
cpdf_add_outline($cpdf, 0, 0, 0, 1, "Page 1");
cpdf_set_font($cpdf, "Times-Roman", 30, "WinAnsiEncoding");
cpdf_set_text_rendering($cpdf, 1);
cpdf_text($cpdf, "Times Roman outlined", 50, 750);
cpdf_moveto($cpdf, 50, 740);
cpdf_lineto($cpdf, 330, 740);
cpdf_stroke($cpdf);
cpdf_finalize($cpdf);
Header("Content-type: application/pdf");
cpdf_output_buffer($cpdf);
cpdf_close($cpdf);
?>
```

La distribución de pdflib contiene un ejemplo mas comlejo que crea una serie de páginas con un reloj analógico. Aquí está ese ejemplo convertido en PHP usando la extensión ClibPDF:

Ejemplo 2. Ejemplo con pdfclock de la distribución pdflib 2.0

```
<?php
$radius = 200;
$margin = 20;
$pagecount = 40;
```

```

$pdf = cpdf_open(0);
cpdf_set_creator($pdf, "pdf_clock.php3");
cpdf_set_title($pdf, "Reloj Analógico");

while($pagecount- > 0) {
    cpdf_page_init($pdf, $pagecount+1, 0, 2 * ($radius + $margin), 2 * ($radius + $margin), 1.0);

    cpdf_set_page_animation($pdf, 4, 0.5, 0, 0, 0); /* limpiar */

    cpdf_translate($pdf, $radius + $margin, $radius + $margin);
    cpdf_save($pdf);
    cpdf_setrgbcolor($pdf, 0.0, 0.0, 1.0);

    /* cambio de minuto */
    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);

    /* cambios de 5 minutos */
    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();

    /* dibujar la aguja de las horas */
    cpdf_save($pdf);
    cpdf_rotate($pdf, -($ltime['minutos']/60.0) + $ltime['horas'] - 3.0) * 30.0);
    cpdf_moveto($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);

    /* dibujar el minutero */
    cpdf_save($pdf);
    cpdf_rotate($pdf, -($ltime['segundos']/60.0) + $ltime['minutos'] - 15.0) * 6.0);
    cpdf_moveto($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);

    /* dibujar la segunda mano */
    cpdf_setrgbcolor($pdf, 1.0, 0.0, 0.0);
    cpdf_setlinewidth($pdf, 2);
}

```

```
cpdf_save($pdf);
cpdf_rotate($pdf, -($ltime['segundos'] - 15.0) * 6.0));
cpdf_moveto($pdf, -$radius/5, 0.0);
cpdf_lineto($pdf, $radius, 0.0);
cpdf_stroke($pdf);
cpdf_restore($pdf);

/* dibujar un pequeño círculo en el centro */
cpdf_circle($pdf, 0, 0, $radius/30);
cpdf_fill($pdf);

cpdf_restore($pdf);

cpdf_finalize_page($pdf, $pagecount+1);
}

cpdf_finalize($pdf);
Header("Content-type: application/pdf");
cpdf_output_buffer($pdf);
cpdf_close($pdf);
?>
```


cpdf_global_set_document_limits (PHP 4 >= 4.0b4)

Sets document limits for any pdf document

```
void cpdf_global_set_document_limits (int maxpages, int maxfonts, int maximages, int maxannotations, int maxobjects)
```

La función **cpdf_global_set_document_limits()** define varios límites del documento. Esta función debe ser llamada antes de **cpdf_open()** para que haga efecto. Ello define los límites de cualquier documento abierto con anterioridad.

Vea también **cpdf_open()**.

cpdf_set_creator (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Define el campo creator en el documento PDF

```
void cpdf_set_creator (string creator)
```

La función **cpdf_set_creator()** define el creador de un documento PDF.

Vea también **cpdf_set_subject()**, **cpdf_set_title()**, **cpdf_set_keywords()**.

cpdf_set_title (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Define el campo title de un documento PDF

```
void cpdf_set_title (string title)
```

La función **cpdf_set_title()** define el título de un documento PDF

Vea también **cpdf_set_subject()**, **cpdf_set_creator()**, **cpdf_set_keywords()**.

cpdf_set_subject (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Define el valor del campo subject de un documento PDF

```
void cpdf_set_subject (string subject)
```

La función **cpdf_set_subject()** define el asunto de un documento PDF

Vea también **cpdf_set_title()**, **cpdf_set_creator()**, **cpdf_set_keywords()**.

cpdf_set_keywords (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Pone el valor del campo 'keywords'(palabras clave) de un documento PDF

```
void cpdf_set_keywords (string keywords)
```

La función **cpdf_set_keywords()** define las palabras clave de un documento PDF.

Vea también **cpdf_set_title()**, **cpdf_set_creator()**, **cpdf_set_subject()**.

cpdf_open (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Abre un nuevo documento PDF

```
int cpdf_open (int compression, string filename)
```

LA función **cpdf_open()** abre un documento PDF nuevo. El primer parámetro activa la compresión del documento si no es igual a 0. El segundo parámetro, opcional, es el fichero en el que el documento es escrito. Si es omitido, el documento es creado en memoria y puede ser escrito en un fichero mediante la función **cpdf_save_to_file()** o escrito por la salida estándar con **cpdf_output_buffer()**.

Nota: El valor de retorno será necesario en nuevas versiones de ClibPDF como el primer parámetro en todas las demás funciones que escriben en el documento PDF.

La librería ClibPDF toma el nombre de fichero "-" como sinónimo de stdout (salida estándar). Si se compila PHP como módulo de apache esto no funcionará porque la manera en que ClibPDF direcciona a la salida estándar no funciona con apache. Usted puede solucionar este problema evitando el enobre de fichero y usando **cpdf_output_buffer()** para la salida de documentos PDF.

Vea también **cpdf_close()**, **cpdf_output_buffer()**.

cpdf_close (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Cierra un documento PDF

```
void cpdf_close (int pdf document)
```

La función **cpdf_close()** cierra un documento PDF. Esta debería ser la última operación incluso después de **cpdf_finalize()**, **cpdf_output_buffer()** y **cpdf_save_to_file()**.

Vea también **cpdf_open()**.

cpdf_page_init (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Comienza una nueva página

```
void cpdf_page_init (int pdf document, int page number, int orientation, double height,  
double width, double unit)
```

La función **cpdf_page_init()** crea una nueva página de altura *height* y profundidad *width*. La página tiene el número *page number* y orientación *orientation*. *orientation* puede ser 0 para retrato y 1 para paisaje. El último parámetro opcional *unit* define la unidad del sistema de coordenadas. El valor debería ser el número de puntos postscript por unidad. Como el valor de una pulgada es igual a 72 puntos, un valor de 72 sería la unidad para una pulgada. Por defecto es 72.

Vea también **cpdf_set_current_page()**.

cpdf_finalize_page (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Finaliza una página

```
void cpdf_finalize_page (int pdf document, int page number)
```

La función **cpdf_finalize_page()** finaliza una página con número de página *page number*. Esta función es sólo para ahorrar memoria. Una página terminada ocupa menos memoria pero no puede volver a ser modificada.

Vea también **cpdf_page_init()**.

cpdf_finalize (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Finaliza un documento

```
void cpdf_finalize (int pdf document)
```

La función **cpdf_finalize()** finaliza un documento. Aún se tiene que llamar a **cpdf_close()**.

Vea también **cpdf_close()**.

cpdf_output_buffer (PHP 3>= 3.0.9, PHP 4 >= 4.0b4)

Pone el documento PDF en el buffer de memoria

```
void cpdf_output_buffer (int pdf document)
```

La función **cpdf_output_buffer()** muestra el documento PDF por la salida estándar. El documento debe ser creado en memoria, que es el caso de la función **cpdf_open()** cuando ha sido llamada sin parámetros.

Vea también **cpdf_open()**.

cpdf_save_to_file (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Escribe el documento PDF en un fichero

```
void cpdf_save_to_file (int pdf document, string filename)
```

La función **cpdf_save_to_file()** guarda el documento PDF en un fichero si este documento ha sido creado en memoria. Esta función no es necesaria si el documento PDF ha sido abierto mediante la especificación de un nombre de fichero en la función **cpdf_open()**.

Vea también **cpdf_output_buffer()**, **cpdf_open()**.

cpdf_set_current_page (PHP 3>= 3.0.9, PHP 4 >= 4.0b4)

Define la página actual

```
void cpdf_set_current_page (int pdf document, int page number)
```

La función **cpdf_set_current_page()** define la página en la que se van a realizar todas las operaciones. Uno puede cambiar entre páginas a menos que una página ha sido finalizada con **cpdf_finalize_page()**.

Vea también **cpdf_finalize_page()**.

cpdf_begin_text (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Inicializa una sección de texto

```
void cpdf_begin_text (int pdf document)
```

La función **cpdf_begin_text()** comienza una sección de texto. Debe ser terminada con **cpdf_end_text()**.

Ejemplo 1. Salida de texto

```
<?php cpdf_begin_text($pdf);
cpdf_set_font($pdf, 16, "Helvetica", "WinAnsiEncoding");
cpdf_text($pdf, 100, 100, "Algún texto");
cpdf_end_text($pdf) ?>
```

Vea también **cpdf_end_text()**.

cpdf_end_text (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Finaliza una sección de texto

```
void cpdf_end_text (int pdf document)
```

La función **cpdf_end_text()** finaliza una sección de texto que fue inicializada con **cpdf_begin_text()**.

Ejemplo 1. Salida de texto

```
<?php cpdf_begin_text($pdf);
cpdf_set_font($pdf, 16, "Helvetica", "WinAnsiEncoding");
cpdf_text($pdf, 100, 100, "Algún texto");
cpdf_end_text($pdf) ?>
```

Vea también **cpdf_begin_text()**.

cpdf_show (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Muestra el texto en la posición actual

```
void cpdf_show (int pdf document, string text)
```

La función **cpdf_show()** muestra la cadena *text* en la posición actual.

Vea también **cpdf_text()**, **cpdf_begin_text()**, **cpdf_end_text()**.

cpdf_show_xy (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Muestra texto en la posición

```
void cpdf_show_xy (int pdf document, string text, double x-coor, double y-coor, int mode)
```

La función **cpdf_show_xy()** muestra la cadena *text* en la posición con coordenadas (*x-coor*, *y-coor*). El último parámetro opcional determina la longitud de la unidad. Si es 0 o se omite, se usa la unidad por defecto especificada para la página. De otro modo las coordenadas son medidas en puntos postscript, despreciando la unidad actual.

Nota: La función **cpdf_show_xy()** es idéntica a **cpdf_text()** sin el parámetro opcional.

Vea también **cpdf_text()**.

cpdf_text (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Muestra texto con parámetros

```
void cpdf_text (int pdf document, string text, double x-coor, double y-coor, int mode,  
double orientation, int alignmode)
```

La función **cpdf_text()** muestra la cadena *text* en la posición de coordenadas (*x-coor*, *y-coor*). El parámetro opcional determina la longitud de la unidad. Si es 0 o se omite, se usa la unidad por defecto especificada para la página. De otro modo las coordenadas son medidas en puntos postscript despreciando la unidad actual. El parámetro opcional *orientation* es la rotación del texto en grados. El parámetro opcional *alignmode* determina cómo está alineado el texto. Vea la documentación de ClibPDF para los posibles valores.

Vea también **cpdf_show_xy()**.

cpdf_set_font (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Selecciona la fuente y el tamaño actual

```
void cpdf_set_font (int pdf document, string font name, double size, string encoding)
```

La función **cpdf_set_font()** define la fuente actual, el tamaño y la codificación. Actualmente solo son soportadas las fuentes estándar de postscript. El último parámetro *encoding* puede tomar los siguientes valores: "MacRomanEncoding", "MacExpertEncoding", "WinAnsiEncoding", y "NULL". "NULL" es para el cifrado incluido en la fuente. Para más información vea el manual de ClibPDF, especialmente para cómo soportar las fuentes asiáticas.

cpdf_set_leading (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Define la distancias entre las líneas de texto

```
void cpdf_set_leading (int pdf document, double distance)
```

La función **cpdf_set_leading()** define la distancia entre las líneas de texto. Esto se usará si el texto es la salida de **cpdf_continue_text()**.

Vea también **cpdf_continue_text()**.

cpdf_set_text_rendering (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Determina cómo es presentado el texto

```
void cpdf_set_text_rendering (int pdf document, int mode)
```

La función **cpdf_set_text_rendering()** determina cómo es presentado el texto. Los posibles valores para *mode* son 0=llenar texto, 1=poner texto, 2=llenar y poner texto, 3=invisible, 4=llenar texto y añadirlo al camino de corte, 5=poner texto y añadirlo al camino de corte, 6=llenar y poner texto y añadirlo al camino de corte, 7=añadirlo al camino de corte

cpdf_set_horiz_scaling (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Define la escala horizontal del texto

```
void cpdf_set_horiz_scaling (int pdf document, double scale)
```

La función **cpdf_set_horiz_scaling()** define la escala horizontal al *scale* por ciento.

cpdf_set_text_rise (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Define la elevación del texto

```
void cpdf_set_text_rise (int pdf document, double value)
```

La función **cpdf_set_text_rise()** define la elevación del texto a *value* unidades.

cpdf_set_text_matrix (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Define la matriz de texto

```
void cpdf_set_text_matrix (int pdf document, array matrix)
```

La función **cpdf_set_text_matrix()** define una matriz que describe una transformación aplicada a la fuente actual de texto.

cpdf_set_text_pos (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Define la posición del texto

```
void cpdf_set_text_pos (int pdf document, double x-koor, double y-koor, int mode)
```

La función **cpdf_set_text_pos()** define la posición del texto para la siguiente llamada a **cpdf_show()**.

El último parámetro opcional *mode* determina la longitud de la unidad. Si es 0 o se omite, se usa el valor por defecto para la página. De otro modo, las coordenadas son medidas en puntos postscript, despreciando la unidad actual.

Vea también **cpdf_show()**, **cpdf_text()**.

cpdf_set_char_spacing (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Determina el espacio entre caracteres

```
void cpdf_set_char_spacing (int pdf document, double space)
```

La función **cpdf_set_char_spacing()** define el espacio entre caracteres.

Vea también **cpdf_set_word_spacing()**, **cpdf_set_leading()**.

cpdf_set_word_spacing (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Define el espacio entre palabras

```
void cpdf_set_word_spacing (int pdf document, double space)
```

La función **cpdf_set_word_spacing()** especifica el espacio entre palabras.

Vea también **cpdf_set_char_spacing()**, **cpdf_set_leading()**.

cpdf_continue_text (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Pone texto en la línea siguiente

```
void cpdf_continue_text (int pdf document, string text)
```

La función **cpdf_continue_text()** pone la cadena *text* en la línea siguiente.

Vea también **cpdf_show_xy()**, **cpdf_text()**, **cpdf_set_leading()**, **cpdf_set_text_pos()**.

cpdf_stringwidth (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Devuelve la anchura del texto en la fuente actual

```
double cpdf_stringwidth (int pdf document, string text)
```

La función **cpdf_stringwidth()** devuelve la anchura de la cadena *text*. Requiere haber definido antes una fuente.

Vea también **cpdf_set_font()**.

cpdf_save (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Salva el entorno actual

```
void cpdf_save (int pdf document)
```

La función **cpdf_save()** salva el entorno actual. Funciona como el comando gsave de postscript. Muy útil si se quiere trasladar o rotar un objeto sin afectar a los demás.

Vea también **cpdf_restore()**.

cpdf_restore (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Restaura un entorno formalmente salvado

```
void cpdf_restore (int pdf document)
```

La función **cpdf_restore()** restaura el entorno salvado con **cpdf_save()**. Funciona como el comando grestore de postscript. Muy útil si se quiere trasladar o rotar un objeto sin afectar otros objetos.

Ejemplo 1. Salvar/Restaurar

```
<?php cpdf_save($pdf);
// hacer todo tipo de rotaciones, transformaciones, ...
cpdf_restore($pdf) ?>
```

Vea también **cpdf_save()**.

cpdf_translate (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Define el sistema de origen de coordenadas

```
void cpdf_translate (int pdf document, double x-koor, double y-koor, int mode)
```

La función **cpdf_translate()** define el sistema origen de coordenadas en el punto (*x-coor*, *y-coor*).

El último parámetro opcional determina la longitud de la unidad. Si es 0 o se omite, se usa la unidad por defecto especificada en la página. De otro modo las coordenadas son medidas en puntos postscript, depreciando la unidad actual.

cpdf_scale (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Define la escala

```
void cpdf_scale (int pdf document, double x-scale, double y-scale)
```

La función **cpdf_scale()** define el factor de escala en los dos sentidos.

cpdf_rotate (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Define la rotación

```
void cpdf_rotate (int pdf_document, double angle)
```

La función **cpdf_rotate()** define la rotación en *angle* grados.

cpdf_setflat (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Define la monotonía

```
void cpdf_setflat (int pdf_document, double value)
```

La función **cpdf_setflat()** pone la monotonía a un valor de entre 0 y 100.

cpdf_setlinejoin (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Define el parámetro linejoin

```
void cpdf_setlinejoin (int pdf_document, long value)
```

La función **cpdf_setlinejoin()** define el parámetro entre un valor de 0 y 2. 0 = ingletes, 1 = redondear, 2 = ángulo oblícuo

cpdf_setlinecap (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Define el parámetro linecap

```
void cpdf_setlinecap (int pdf_document, int value)
```

La función **cpdf_setlinecap()** define el parámetro linecap entre los valores 0 y 2. 0 = empalmar al final, 1 = redondear, 2 = esquina proyectada

cpdf_setmiterlimit (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Define el límite del inglete

```
void cpdf_setmiterlimit (int pdf_document, double value)
```

La función **cpdf_setmiterlimit()** define el límite del inglete a un valor mayor o igual a 1.

cpdf_setlinewidth (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Define la profundidad de la línea

```
void cpdf_setlinewidth (int pdf_document, double width)
```

La función **cpdf_setlinewidth()** define la preofundidad de la línea a *width*.

cpdf_setdash (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Define el patrón de la raya

```
void cpdf_setdash (int pdf_document, double white, double black)
```

La función **cpdf_setdash()** define el patrón de la raya *white* unidades blancas y *black* unidades negras. Si los dos son 0 se pone una línea sólida.

cpdf_moveto (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Define el punto actual

```
void cpdf_moveto (int pdf_document, double x-koor, double y-koor, int mode)
```

La función **cpdf_moveto()** pone el punto actual en las coordenadas *x-koor* y *y-koor*.

El último parámetro opcional determina la longitud de la unidad. Si es 0 o se omite, la unidad por defecto será la especificada para la página. De otro modo las coordenadas se medirán en puntos postscript despreciando la unidad en curso.

cpdf_rmoveto (PHP 3>= 3.0.9, PHP 4 >= 4.0b4)

Define el punto actual

```
void cpdf_rmoveto (int pdf_document, double x-koor, double y-koor, int mode)
```

La función **cpdf_rmoveto()** pone el punto actual relativo a las coordenadas *x-koor* y *y-koor*.

El último parámetro opcional determina la loingitud de la unidad. Si es 0 o se omite, la unidad por defecto será la especificada para la página. De otro modo las coordenadas se medirán en puntos postscript, despreciando la unidad en curso.

Vea también **cpdf_moveto()**.

cpdf_curveto (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Dibuja una curva

```
void cpdf_curveto (int pdf_document, double x1, double y1, double x2, double y2, double x3, double y3, int mode)
```

La función **cpdf_curveto()** dibuja una curva Bezier desde el punto actual al punto (*x3*, *y3*) usando (*x1*, *y1*) y (*x2*, *y2*) como puntos de control.

El último parámetro opcional especifica el tamaño de la unidad. Si es 0 o se omite, se usa la unidad especificada para la página. De otro modo las coordenadas se miden en puntos postscript, despreciando la unidad en curso.

Vea también **cpdf_moveto()**, **cpdf_rmoveto()**, **cpdf_rlineto()**, **cpdf_lineto()**.

cpdf_lineto (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Dibuja una línea

```
void cpdf_lineto (int pdf document, double x-koor, double y-koor, int mode)
```

La función **cpdf_lineto()** dibuja una línea desde el punto actual al punto con coordenadas (*x-koor*, *y-koor*).

El último parámetro opcional determina el tamaño de la unidad. Si es 0 o se omite, se usa el valor especificado para la página por defecto. De otro modo las coordenadas se miden en puntos postscript, despreciando la unidad actual.

Vea también **cpdf_moveto()**, **cpdf_rmoveto()**, **cpdf_curveto()**.

cpdf_rlineto (PHP 3>= 3.0.9, PHP 4 >= 4.0b4)

Dibuja una línea

```
void cpdf_rlineto (int pdf document, double x-koor, double y-koor, int mode)
```

La función **cpdf_rlineto()** dibuja una línea desde el punto actual al punto relativo con coordenadas (*x-koor*, *y-koor*).

El último parámetro opcional determina la longitud de la unidad. Si es 0 o se omite, se usa el valor por defecto para la página. De otro modo las coordenadas se miden en puntos postscript, despreciando la unidad actual.

Vea también **cpdf_moveto()**, **cpdf_rmoveto()**, **cpdf_curveto()**.

cpdf_circle (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Dibuja un círculo

```
void cpdf_circle (int pdf document, double x-koor, double y-koor, double radius, int mode)
```

La función **cpdf_circle()** dibuja un círculo con centro en el punto (*x-koor*, *y-koor*) y radio *radius*.

El último parámetro opcional define el tamaño de la unidad. Si es 0 o se omite, se usa el valor por defecto para la página. De otro modo las coordenadas se miden en puntos postscript, despreciando la unidad actual.

Vea también **cpdf_arc()**.

cpdf_arc (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Dibuja un arco

```
void cpdf_arc (int pdf document, double x-koor, double y-koor, double radius, double start, double end, int mode)
```

La función **cpdf_arc()** dibuja un arco con el centro en el punto (*x-koor*, *y-koor*) y radio *radius*, empezando en el ángulo *start* y terminando en el ángulo *end*.

El último parámetro opcional especifica el tamaño de la unidad. Si es 0 o se omite, se usa la unidad especificada por defecto. De otro modo las coordenadas son medidas en puntos postscript,despreciando la unidad actual.

Vea también **cpdf_circle()**.

cpdf_rect (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Dibuja un rectángulo

```
void cpdf_rect (int pdf document, double x-koor, double y-koor, double width, double height, int mode)
```

La función **cpdf_rect()** dibuja un rectángulo con su esquina inferior izquierda en el punto (*x-koor*, *y-koor*). La anchura es *width*. La altura es *height*.

El último parámetro opcional define el tamaño de la unidad. Si es 0 o se omite, se usa la unidad por defecto especificada para la página. De otro modo las coordenadas se miden en puntos postscript, despreciando la unidad actual.

cpdf_closepath (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Cierra el camino

```
void cpdf_closepath (int pdf document)
```

La función **cpdf_closepath()** cierra el camino actual.

cpdf_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Dibuja una línea a lo largo del camino

```
void cpdf_stroke (int pdf document)
```

La función **cpdf_stroke()** dibuja una línea a lo largo del camino actual.

Vea también **cpdf_closepath()**, **cpdf_closepath_stroke()**.

cpdf_closepath_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Cierra el camino y dibuja una línea a lo largo del camino

```
void cpdf_closepath_stroke (int pdf document)
```

La función **cpdf_closepath_stroke()** es una combinación de **cpdf_closepath()** y **cpdf_stroke()**. Después limpia el camino.

Vea también **cpdf_closepath()**, **cpdf_stroke()**.

cpdf_fill (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

LLena el camino actual

```
void cpdf_fill (int pdf document)
```

La función **cpdf_fill()** llena el interior del camino actual con el color actual de relleno.

Vea también **cpdf_closepath()**, **cpdf_stroke()**, **cpdf_setgray_fill()**, **cpdf_setgray()**, **cpdf_setrgbcolor_fill()**, **cpdf_setrgbcolor()**.

cpdf_fill_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

LLena y traza el camino actual

```
void cpdf_fill_stroke (int pdf document)
```

La función **cpdf_fill_stroke()** llena el interior del camino actual con el color de relleno actual y dibuja el camino actual.

Vea también **cpdf_closepath()**, **cpdf_stroke()**, **cpdf_fill()**, **cpdf_setgray_fill()**, **cpdf_setgray()**, **cpdf_setrgbcolor_fill()**, **cpdf_setrgbcolor()**.

cpdf_closepath_fill_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Cierra, llena y traza el camino actual

```
void cpdf_closepath_fill_stroke (int pdf document)
```

La función **cpdf_closepath_fill_stroke()** cierra, llena el interior del camino actual con el color actual de relleno y dibuja el camino actual.

Vea también **cpdf_closepath()**, **cpdf_stroke()**, **cpdf_fill()**, **cpdf_setgray_fill()**, **cpdf_setgray()**, **cpdf_setrgbcolor_fill()**, **cpdf_setrgbcolor()**.

cpdf_clip (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Ajusta al camino actual

```
void cpdf_clip (int pdf document)
```

La función **cpdf_clip()** ajusta todos los dibujos al camino actual.

cpdf_setgray_fill (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Pone el color de relleno al valor gris

```
void cpdf_setgray_fill (int pdf document, double value)
```

La función **cpdf_setgray_fill()** define el valor de gris actual para rellenar un camino.

Vea también **cpdf_setrgbcolor_fill()**.

cpdf_setgray_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Define el color para dibujar al valor gris

```
void cpdf_setgray_stroke (int pdf document, double gray value)
```

La función **cpdf_setgray_stroke()** pone el color de dibujo actual al valor de gris dado.

Vea también **cpdf_setrgbcolor_stroke()**.

cpdf_setgray (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Pone el color de relleno y dibujo a gris

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

La función **cpdf_setgray_stroke()** pone el color de relleno y dibujo al color gris dado.

Vea también **cpdf_setrgbcolor_stroke()**, **cpdf_setrgbcolor_fill()**.

cpdf_setrgbcolor_fill (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Pone el color de relleno a l valor de clor rgb

```
void cpdf_setrgbcolor_fill (int pdf document, double red value, double green value, double blue value)
```

La función **cpdf_setrgbcolor_fill()** pone el color rgb actual para llenar un camino.

Vea también **cpdf_setrgbcolor_stroke()**, **cpdf_setrgbcolor()**.

cpdf_setrgbcolor_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Pone el color de dibujo al valor de color rgb

```
void cpdf_setrgbcolor_stroke (int pdf document, double red value, double green value, double blue value)
```

La función **cpdf_setrgbcolor_stroke()** pone el color de dibujo actual al valor de color rgb dado.

Vea también **cpdf_setrgbcolor_fill()**, **cpdf_setrgbcolor()**.

cpdf_setrgbcolor (PHP 3>= 3.0.8, PHP 4 >= 4.0b4)

Pone el color de relleno y dibujo al valor de color rgb

```
void cpdf_setrgbcolor (int pdf document, double red value, double green value, double blue value)
```

La función **cpdf_setrgbcolor_stroke()** pone el color de relleno y dibujo actual al color rgb dado.

Vea también **cpdf_setrgbcolor_stroke()**, **cpdf_setrgbcolor_fill()**.

cpdf_add_outline (PHP 3>= 3.0.9, PHP 4 >= 4.0b4)

Añade una marca en la página actual

```
void cpdf_add_outline (int pdf document, string text)
```

La función **cpdf_add_outline()** añade una marca con el texto *text* que apunta a la página actual.

Ejemplo 1. Añadiendo un contorno de página

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

cpdf_set_page_animation (PHP 3>= 3.0.9, PHP 4 >= 4.0b4)

Define la separación entre páginas

```
void cpdf_set_page_animation (int pdf document, int transition, double duration)
```

La función **cpdf_set_page_animation()** define la transición entre páginas que se siguen.

El valor de *transition* puede ser

- 0 para ninguno,
- 1 para dos líneas que se barren a través de la pantalla, revelen la página,
- 2 para múltiples líneas,
- 3 para que una caja revele la página,

- 4 para una única línea,
- 5 para que la página anterior se disipe para revelar la pagina,
- 6 para que el efecto de disolución se mueva de un extremop de la página al otro,
- 7 para que la página antigua simplemente sea reemplazada por la nueva página (default)

El valor de *duration* es el número de segundos entre las páginas que se pasan.

cpdf_import_jpeg (PHP 3>= 3.0.9, PHP 4 >= 4.0b4)

Abre una imagen JPEG

```
int cpdf_open_jpeg (int pdf_document, string file_name, double x-koor, double y-koor,
double angle, double width, double height, double x-scale, double y-scale, int mode)
```

La función **cpdf_import_jpeg()** abre una imagen almacenada en el fichero de nombre *file name*. El formato de la imagen debe ser JPEG. La imagen es situada en la página actual en la posición (*x-koor*, *y-koor*). La imagen es rotada *angle* grados.

El último parámetro opcional determina el tamaño de la unidad. Si es 0 o se omite, se usa la unidad por defecto especificada para la página. De otro modo las coordenadas se miden en puntos postscript, despreciando la unidad actual.

Vea también **cpdf_place_inline_image()**.

cpdf_place_inline_image (PHP 3>= 3.0.9, PHP 4 >= 4.0b4)

Situa una imagen en la página

```
void cpdf_place_inline_image (int pdf_document, int image, double x-koor, double y-koor,
double angle, double width, double height, int mode)
```

La función **cpdf_place_inline_image()** situa una imagen creada con las funciones de imágenes de PHP en la posición de la página (*x-koor*, *y-koor*). La imagen puede ser escalada al mismo tiempo.

El último parámetro opcional determina el tamaño de la unidad. Si es 0 o se omite, se usa la unidad por defecto especificada para la página. De otro modo las coordenadas son medidas en puntos postscript, descartando la unidad actual.

Vea también **cpdf_import_jpeg()**.

cpdf_add_annotation (PHP 3>= 3.0.12, PHP 4 >= 4.0b4)

Añade una anotación

```
void cpdf_add_annotation (int pdf_document, double llx, double lly, double urx, double
ury, string title, string content, int mode)
```

La función **cpdf_add_annotation()** añade una nota con la esquina inferior izquierda en (*llx*, *lly*) y la esquina superior derecha en (*urx*, *ury*).

El último parámetro opcional determina el tamaño de la unidad. Si es 0 o se omite, se usa la unidad por defecto especificada para la página. De otro modo las coordenadas se miden en puntos postscript, despreciando la unidad actual.

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:

Ejemplo 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_INFILE, $fp);
curl_setopt ($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.

Ejemplo 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-form-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.
- *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 "PORT"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 (void);
```

The **curl_version()** function returns a string containing the current CURL version.

XII. Funciones de pago electrónico

Estas funciones solo están disponibles si el intérprete ha sido compilado con `-with-cybercash=[DIR]`. Estas funciones han sido añadidas en PHP4.

cybercash_encr (PHP 4 >= 4.0b4)

???

```
array cybercash_encr (string wmk, string sk, string inbuff)
```

La función devuelve un array asociativo con los elementos "errcode"y, si "errcode"es false, "outbuff"(string), "outLth"(long) y "macbuff"(string).

cybercash_decr (PHP 4 >= 4.0b4)

???

```
array cybercash_decr (string wmk, string sk, string inbuff)
```

La función devuelve un array asociativo con los elementos "errcode"y, si "errcode"es false, "outbuff"(string), "outLth"(long) y "macbuff"(string).

cybercash_base64_encode (PHP 4 >= 4.0b4)

???

```
string cybercash_base64_encode (string inbuff)
```

cybercash_base64_decode (PHP 4 >= 4.0b4)

```
string cybercash_base64_decode (string inbuff)
```


XIII. Character type functions

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.

Aviso

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_somthing()** 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)
```

See also **setlocale()**.

ctype_alpha (PHP 4 >= 4.0.4)

Check for alphabetic character(s)

```
bool ctype_alpha (string c)
```

ctype_cntrl (PHP 4 >= 4.0.4)

Check for control character(s)

```
bool ctype_cntrl (string c)
```

ctype_digit (PHP 4 >= 4.0.4)

Check for numeric character(s)

```
bool ctype_digit (string c)
```

ctype_lower (PHP 4 >= 4.0.4)

Check for lowercase character(s)

```
bool ctype_lower (string c)
```

ctype_graph (PHP 4 >= 4.0.4)

Check for any printable character(s) except space

```
bool ctype_graph (string c)
```

ctype_print (PHP 4 >= 4.0.4)

Check for printable character(s)

```
bool ctype_print (string c)
```

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)
```

ctype_space (PHP 4 >= 4.0.4)

Check for whitespace character(s)

```
bool ctype_space (string c)
```

ctype_upper (PHP 4 >= 4.0.4)

Check for uppercase character(s)

```
bool ctype_upper (string c)
```

ctype_xdigit (PHP 4 >= 4.0.4)

Check for character(s) representing a hexadecimal digit

```
bool ctype_xdigit (string c)
```

XIV. Funciones de la capa de abstraccion de bases de datos (dbm-style)

Estas funciones son la base para el acceso a bases de datos del estilo Berkeley DB.

Este es un nivel de abstraccion general para varias bases de datos. Como tal su funcionalidad esta limitada a un grupo de modernas bases de datos como Sleepycat Software's DB2 (). (Esta no debe confundirse con IBM DB2 software, la cual es soportada mediante las funciones [ODBC](#).)

El comportamiento de varios aspectos depende de la implementacion de la base de datos. Funciones como **dba_optimize()** y **dba_sync()** cumpliran su funcionalidad con unas bases de datos pero no con otras.

Los siguientes manejadores (handlers) estan soportados:

- dbm es el mas antiguo (original) tipo de base de datos de la familia de Berkeley DB. Se debe evitar su uso, si es posible. Nosotros no soportamos las funciones de compatibilidad de DB2 y gdbm, porque ellas solo son compatibles a nivel de codigo fuente, pero no pueden manejar el formato original dbm.
- ndbm es un tipo mas nuevo y mas flexible que dbm. Todavia tiene la mayoria de las limitaciones de dbm (Por lo tanto es descartado).
- gdbm es el gestor de bases de datos de GNU (database manager) () .
- db2 es Sleepycat Software's DB2 (). Es descrito como "un conjunto de herramientas de programacion que proveen acceso de alto nivel a bases de datos en aplicaciones standalone o en el modelo cliente/servidor. "
- cdb es "una rapida, de confianza, sencilla herramienta para la creacion y lectura de bases de datos constantes."Fue creada por el autor de qmail y puede encontrarse en [here](#) (). Como la base es constante solo se soportan las operaciones de lectura.

Ejemplo 1. Ejemplo de DBA

```
<?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 es "binary safe"y no tiene ningun limite arbitrario. Hereda todas sus limitaciones de la implementacion de base de datos que tenga.

Todos las bases de datos basadas en ficheros deben proveer un mecanismo para establecer el modo a la hora de crear nuevas bases de datos, si ello es posible. Habitualmente este modo es pasado como el cuarto argumento en **dba_open()** o en **dba_popen()**.

Se puede acceder a todas las entradas de una base de datos de modo secuencial (lineal) usando las funciones **dba_firstkey()** y **dba_nextkey()**. No se puede cambiar la base de datos mientras se recorre (traversing) por ella.

Ejemplo 2. Recorriendo una base de datos

```
<?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 >= 4.0b2)

Cerrar una base de datos

```
void dba_close (int handle)
```

dba_close() cierra la conexión con una base de datos previamente abierta y libera todos los recursos especificados por *handle*.

handle es un manejador (handle) de la base de datos devuelto por **dba_open()**.

dba_close() no devuelve ningún valor.

Ver también: **dba_open()** **dba_popen()**

dba_delete (PHP 3>= 3.0.8, PHP 4 >= 4.0b2)

Borra una entrada especificada por la clave key

```
string dba_delete (string key, int handle)
```

dba_delete() borra la entrada especificada por *key* de la base de datos especificada por *handle*.

key es la clave de la entrada que es borrada.

handle es un manejador (handle) de la base de datos devuelto por **dba_open()**.

dba_delete() devuelve true o false, si la entrada es borrada o no, respectivamente.

Ver también: **dba_exists()** **dba_fetch()** **dba_insert()** **dba_replace()**

dba_exists (PHP 3>= 3.0.8, PHP 4 >= 4.0b2)

Comprueba si la clave key existe

```
bool dba_exists (string key, int handle)
```

dba_exists() comprueba si la clave *key* existe en la base de datos especificada por *handle*.

key es la clave para la que se realiza la comprobación.

handle es un manejador (handle) de la base de datos devuelto por **dba_open()**.

dba_exists() devuelve true o false, si la clave es hallada o no, respectivamente.

Ver también: **dba_fetch()** **dba_delete()** **dba_insert()** **dba_replace()**

dba_fetch (PHP 3>= 3.0.8, PHP 4 >= 4.0b2)

Extrae los datos especificados por la clave key

```
string dba_fetch (string key, int handle)
```

dba_fetch() extrae los datos especificados por la clave *key* de la base de datos determinada por *handle*.

key es la clave de la entrada de los datos que queremos extraer.

handle es un manejador (handle) de la base de datos devuelto por **dba_open()**.

dba_fetch() devuelve la cadena asociada o false, si el par key/data es hallado o no, respectivamente.

Ver tambien: **dba_exists()** **dba_delete()** **dba_insert()** **dba_replace()**

dba_firstkey (PHP 3>= 3.0.8, PHP 4 >= 4.0b2)

Conseguir la primera clave

```
string dba_firstkey (int handle)
```

dba_firstkey() devuelve la primera clave de la base de datos especificada por *handle* y resetea el puntero interno de claves. Esto permite una busqueda lineal por toda la base de datos.

handle es un manejador (handle) de la base de datos devuelto por **dba_open()**.

dba_firstkey() devuelve la clave o false en funcion de si tiene exito o falla, respectivamente.

Ver tambien: **dba_nextkey()**

dba_insert (PHP 3>= 3.0.8, PHP 4 >= 4.0b2)

Insertar una entrada

```
bool dba_insert (string key, string value, int handle)
```

dba_insert() inserta la entrada descrita con *key* y *value* dentro de la base de datos especificada por *handle*. Fallara si ya existe una entrada con el mismo parametro *key*.

key es la clave de la entrada a ser insertada.

value es el valor a ser insertado.

handle es un manejador (handle) de la base de datos devuelto por **dba_open()**.

dba_insert() devuelve true o false, en funcion de si tiene exito o falla, respectivamente.

Ver tambien: **dba_exists()** **dba_delete()** **dba_fetch()** **dba_replace()**

dba_nextkey (PHP 3>= 3.0.8, PHP 4 >= 4.0b2)

Extraer la siguiente clave

```
string dba_nextkey (int handle)
```

dba_nextkey() devuelve la siguiente clave de la base de datos especificada por *handle* e incrementa el puntero de claves interno.

handle es un manejador (handle) de la base de datos devuelto por **dba_open()**.

dba_nextkey() devuelve la clave o false dependiendo de si tiene exito o falla, respectivamente.

Ver tambien: **dba_firstkey()**

dba_popen (PHP 3>= 3.0.8, PHP 4 >= 4.0b2)

Aertura persistente de una base de datos

```
int dba_popen (string path, string mode, string handler [, ...])
```

dba_popen() establece una instancia persistente para *path* con *mode* usando *handler*.

path normalmente es el "path"en el sistema de ficheros.

mode es "r"para acceso de lectura, "w"para lectura/escritura de una base de datos ya existente, "c"para lectura/escritura y creacion de una base datos si esta no existe, y "n"para crear, truncar y lectura/escritura.

handler es el nombre del manejador (handler) que sera usado para el acceso al *path*. Es pasado como un parametro opcional a **dba_popen()** y puede usarse en lugar de ella.

dba_popen() devuelve un valor positivo de handler o false, en el caso de que la apertura de la base de datos se realice o si falla, respectivamente.

Ver tambien: **dba_open()** **dba_close()**

dba_open (PHP 3>= 3.0.8, PHP 4 >= 4.0b2)

Abrir una base de datos

```
int dba_open (string path, string mode, string handler [, ...])
```

dba_open() establece una instancia para *path* con *mode* usando *handler*.

path normalmente es el "path"en el sistema de ficheros.

mode es "r"para acceso de lectura, "w"para lectura/escritura de una base de datos ya existente, "c"para lectura/escritura y creacion de una base datos si esta no existe, y "n"para crear, truncar y lectura/escritura.

handler es el nombre de el manejador (handler) que sera usado para el acceso al *path*. Es pasado como un parametro opcional a **dba_open()** y puede usarse en lugar de ella.

dba_open() devuelve un valor positivo de handler o false, en el caso de que la apertura de la base de datos se realice o si falla, respectivamente.

Ver tambien: **dba_popen()** **dba_close()**

dba_optimize (PHP 3>= 3.0.8, PHP 4 >= 4.0b2)

Optimiza la base de datos

```
bool dba_optimize (int handle)
```

dba_optimize() optimiza la base de datos especificada por *handle*.

handle es un manejador (handle) de la base de datos devuelto por **dba_open()**.

dba_optimize() devuelve true o false, si la optimizacion tiene exito o falla, respectivamente.

Ver tambien: **dba_sync()**

dba_replace (PHP 3>= 3.0.8, PHP 4 >= 4.0b2)

Reemplaza o inserta una entrada

```
bool dba_replace (string key, string value, int handle)
```

dba_replace() reemplaza o inserta la entrada descrita con *key* y *value* dentro de la base de datos especificada por *handle*.

key es la clave de la entrada a insertar.

value es el valor a ser insertado.

handle es un manejador (handle) de la base de datos devuelto por **dba_open()**.

dba_replace() devuelve true o false, dependiendo de si tiene exito o falla respectivamente.

Ver tambien: **dba_exists()** **dba_delete()** **dba_fetch()** **dba_insert()**

dba_sync (PHP 3>= 3.0.8, PHP 4 >= 4.0b2)

Sincroniza la base de datos

```
bool dba_sync (int handle)
```

dba_sync() sincroniza la base de datos especificada por *handle*. Esto probablemente realice una escritura fisica en el disco, si es soportado.

handle es un manejador (handle) de la base de datos devuelto por **dba_open()**.

dba_sync() devuelve true o false, si la sincronizacion tiene exito o falla, respectivamente.

Ver tambien: **dba_optimize()**

XV. Funciones de fecha y hora

checkdate (PHP 3, PHP 4)

valida una fecha u hora

```
int checkdate (int month, int day, int year)
```

Devuelve un valor verdadero si la fecha dada es válida; en caso contrario, devuelve un valor falso. Comprueba la validez de la fecha formada por los argumentos. Se considera válida una fecha si:

- el año está entre 0 y 32767, ambos incluidos
- el mes está entre 1 y 12, ambos incluidos
- el día está en el rango permitido para el mes dado. Se tienen en consideración los años bisiestos.

date (PHP 3, PHP 4)

da formato a la fecha/hora local

```
string date (string format [, int timestamp])
```

Devuelve una cadena formateada de acuerdo con la cadena de formato dada, utilizando el valor de *timestamp* dado o la hora local actual si no hay parámetro.

Se reconocen los siguientes caracteres en la cadena de formato:

- a - "am"o "pm"
- A - "AM"o "PM"
- d - día del mes, dos dígitos con cero a la izquierda; es decir, de "01" a "31"
- D - día de la semana, en texto, con tres letras; por ejemplo, "Fri"
- F - mes, en texto, completo; por ejemplo, "January"
- h - hora, de "01" a "12"
- H - hora, de "00" a "23"
- g - hour, sin ceros, de "1" a "12"
- G - hour, sin ceros; de "0" a "23"
- i - minutos; de "00" a "59"
- j - día del mes sin cero inicial; de "1" a "31"
- l ('L' minúscula) - día de la semana, en texto, completo; por ejemplo, "Friday"
- L - "1"or "0", según si el año es bisiesto o no
- m - mes; de "01" a "12"
- n - mes sin cero inicial; de "1" a "12"
- M - mes, en texto, 3 letras; por ejemplo, "Jan"
- s - segundos; de "00" a "59"
- S - sufijo ordinal en inglés, en texto, 2 caracteres; por ejemplo, "th", "nd"
- t - número de días del mes dado; de "28" a "31"

- U - segundos desde el valor de 'epoch'
- w - día de la semana, en número, de "0"(domingo) a "6"(sábado)
- Y - año, cuatro cifras; por ejemplo, "1999"
- y - año, dos cifras; por ejemplo, "99"
- z - día del año; de "0" a "365"
- Z - diferencia horaria en segundos (de "-43200" a "43200")

Los caracteres no reconocidos se imprimen tal cual. El formato "Z" siempre devuelve "0" en la función **gmdate()**

Ejemplo 1. Ejemplo de date()

```
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)));
```

Es posible usar **date()** y **mktime()** juntas para obtener fechas futuras o pasadas.

Ejemplo 2. Ejemplo de date() y mktime()

```
$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);
```

Para dar formato a fechas en otros idiomas, se deben usar las funciones **setlocale()** y **strftime()**.

Ver también **gmdate()** y **mktime()**.

getdate (PHP 3, PHP 4)

obtiene información de fecha y hora

```
array getdate ( int timestamp )
```

Devuelve un array asociativo que contiene la información de fecha del valor timestamp como los siguientes elementos:

- "seconds"- segundos
- "minutes"- minutos
- "hours"- horas
- "mday"- día del mes
- "wday"- día de la semana, en número
- "mon"- mes, en número
- "year"- año, en número
- "yday"- día del año, en número; por ejemplo, "299"
- "weekday"- día de la semana, en texto, completo; por ejemplo, "Friday"
- "month"- mes, en texto, completo; por ejemplo, "January"

gettimeofday (PHP 3>= 3.0.7, PHP 4 >= 4.0b4)

obtiene la hora actual

```
array gettimeofday (void)
```

Es un interfaz para gettimeofday(2). Devuelve un array asociativo que contiene los datos devueltos por esta llamada al sistema.

- "sec"- segundos
- "usec"- microsegundos
- "minuteswest"- minutos al oeste de Greenwich
- "dsttime"- tipo de corrección dst

gdate (PHP 3, PHP 4)

da formato a una fecha/hora GMT/CUT

```
string gdate (string format, int timestamp)
```

Idéntica a la función **date()** excepto en que la hora devuelta es la de Greenwich (GMT). Por ejemplo, si se utiliza en Finlandia (GMT +0200), la primera línea del ejemplo devuelve "Jan 01 1998 00:00:00", mientras la segunda imprime "Dec 31 1997 22:00:00".

Ejemplo 1. Ejemplo de gdate()

```
echo date( "M d Y H:i:s",mktime(0,0,0,1,1,1998) );
echo gdate( "M d Y H:i:s",mktime(0,0,0,1,1,1998) );
```

Ver también **date()**, **mktime()** y **gmmktime()**.

gmmktime (PHP 3, PHP 4)

obtiene el valor timestamp UNIX de una fecha GMT

```
int gmmktime (int hour, int minute, int second, int month, int day, int year [, int is_dst])
```

Idéntica a **mktime()**, excepto en que los parámetros representan una fecha GMT.

gmstrftime (PHP 3>= 3.0.12, PHP 4 >= 4.0RC2)

da formato a una fecha/hora GMT/CUT según las convenciones locales

```
string gmstrftime (string format, int timestamp)
```

Se comporta como **strftime()**, excepto en que la hora devuelta es la de Greenwich (GMT). Por ejemplo, si se utiliza en la zona horaria EST (GMT -0500), la primera línea del ejemplo imprime "Dec 31 1998 20:00:00", mientras la segunda imprime "Jan 01 1999 01:00:00".

Ejemplo 1. Ejemplo de gmstrftime()

```
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";
```

Ver también **strftime()**.

microtime (PHP 3, PHP 4)

devuelve el valor timestamp UNIX actual con microsegundos

```
string microtime(void);
```

Devuelve la cadena "msec sec", donde sec es la hora actual en número de segundos desde el valor Unix Epoch (0:00:00 del 1 de enero de 1970, hora GMT), y msec es la parte de microsegundos. Esta función sólo está disponible en sistemas operativos con admiten la llamada al sistema gettimeofday().

Ver también **time()**.

mkttime (PHP 3, PHP 4)

obtiene el timestamp UNIX de una fecha

```
int mkttime (int hour, int minute, int second, int month, int day, int year [, int is_dst])
```

Advertencia: Véase el extraño orden de los argumentos, que se diferencia del orden de argumentos en una llamada mkttime() de UNIX y que no permite eliminar parámetros de derecha a izquierda (ver abajo). Es un error común mezclar estos valores en un script.

Devuelve el valor timestamp Unix correspondiente a los argumentos dados. El timestamp es un entero de tipo long que contiene el número de segundos entre el valor Unix Epoch (1 de enero de 1970) y la hora especificada.

Se pueden eliminar argumentos en orden de derecha a izquierda; en los argumentos omitidos se toma el valor de la fecha y hora locales.

is_dst puede ponerse a 1 si la hora corresponde a horario de verano, 0 si no, o -1 (valor por omisión) si no se sabe.

Nota: *is_dst* se añadió en la versión 3.0.10.

mkttime() es útil para realizar cálculos y validaciones con fechas, ya que calcula automáticamente el valor correcto para una entrada fuera de rango. Por ejemplo, cada una de las líneas siguientes produce la cadena "Jan-01-1998".

Ejemplo 1. Ejemplo de mkttime()

```
echo date( "M-d-Y", mkttime(0,0,0,12,32,1997) );
echo date( "M-d-Y", mkttime(0,0,0,13,1,1997) );
```

```
echo date( "M-d-Y", mktime(0,0,0,1,1,1998) );
```

El último día de cada mes se puede expresar como el día "0" del mes siguiente, no el día -1. Los dos ejemplos siguientes producen la cadena "The last day in Feb 2000 is: 29".

Ejemplo 2. El último día del próximo mes

```
$lastday=mktime(0,0,0,3,0,2000);
echo strftime("Last day in Feb 2000 is: %d",$lastday);

$lastday=mktime(0,0,0,4,-31,2000);
echo strftime("Last day in Feb 2000 is: %d",$lastday);
```

Ver también **date()** y **time()**.

strftime (PHP 3, PHP 4)

da formato a la hora o fecha local de acuerdo con las convenciones locales

```
string strftime (string format, int timestamp)
```

Devuelve una cadena formateada según la cadena de formato dada utilizando el valor *timestamp* o la hora local actual. Los nombres del mes y el día de la semana y otras cadenas dependientes del idioma respetan lo establecido con **setlocale()**.

Se reconocen los siguientes especificadores de conversión en la cadena de formato:

- %a - nombre del día de la semana abreviado
- %A - nombre del día de la semana completo
- %b - nombre del mes abreviado
- %B - nombre del mes completo
- %c - representación de fecha y hora preferidas en el idioma actual
- %d - día del mes en número (de 00 a 31)
- %H - hora como un número de 00 a 23
- %I - hora como un número de 01 a 12
- %j - día del año como un número de 001 a 366
- %m - mes como un número de 01 a 12
- %M - minuto en número
- %p - 'am' o 'pm', según la hora dada, o las cadenas correspondientes en el idioma actual
- %S - segundos en número
- %U - número de la semana en el año, empezando con el primer domingo como el primer día de la primera semana
- %W - número de la semana en el año, empezando con el primer lunes como el primer día de la primera semana
- %w - día de la semana en número (el domingo es el 0)
- %x - representación preferida de la fecha sin la hora
- %X - representación preferida de la hora sin la fecha

- %y - año en número de 00 a 99
- %Y - año en número de cuatro cifras
- %Z - nombre o abreviatura de la zona horaria
- %% - carácter ‘%’

Ejemplo 1. Ejemplo de strftime()

```
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"));
```

Este ejemplo funciona si se tienen los respectivos ‘locales’ instalados en el sistema.

Ver también **setlocale()** y **mktime()**.

time (PHP 3, PHP 4)

devuelve el timestamp UNIX actual

```
int time(void);
```

Devuelve la hora actual como número de segundos transcurridos desde las 00:00:00 del 1 de enero de 1970 GMT (Unix Epoch).

Ver también **date()**.

XVI. Funciones para dBase

Estas funciones permiten el acceso a datos almacenados en formato dBase (dbf).

No hay soporte para índices o campos Memo. Tampoco hay soporte para bloqueo: si dos procesos concurrentes en el servidor modifican el mismo fichero dBase, probablemente se destruirán los datos.

A diferencia de las bases de datos SQL, las "bases de datos" dBase no pueden cambiar su definición. Una vez creado el fichero, la definición de la base de datos es fija. No hay índices que aceleren la búsqueda u organicen los datos de distinto modo. Los ficheros dBase son simples ficheros secuenciales con registros de longitud fija. Los nuevos registros se añaden al final del fichero y los registros borrados se conservan hasta que se llama a la función **dbase_pack()**.

Se recomienda no utilizar ficheros dBase como bases de datos, sino elegir cualquier servidor SQL; MySQL o Postgres son opciones habituales con PHP. El soporte para dBase se proporciona para permitir importar y exportar datos a y desde la base de datos web, ya que este formato de ficheros es aceptado habitualmente por las hojas de datos y los organizadores de Windows. La importación y exportación de datos es lo único para lo que sirve el soporte dBase.

dbase_create (PHP 3, PHP 4)

crea una base de datos dBase

```
int dbase_create (string filename, array fields)
```

El parámetro *fields* es un array de arrays, cada uno de los cuales describe el formato de un campo de la base de datos. Cada campo consiste de un nombre, un carácter que indica el tipo de campo, una longitud, y una precisión.

Los tipos de campos disponibles son:

L

Lógico. No tienen longitud ni precisión.

M

Memo. (Sin soporte en PHP.) No tienen longitud ni precisión.

D

Fecha (almacenada como AAAAMMDD). No tienen longitud ni precisión.

N

Número. Tienen longitud y precisión (número de cifras tras el punto decimal).

C

Cadena.

Si la base de datos se crea con éxito, se devuelve un dbase_identifier; en caso contrario, devuelve false.

Ejemplo 1. Crear un fichero dBase

```
// "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)

abre un fichero dBase

```
int dbase_open (string filename, int flags)
```

Los "flags" son los que utiliza la llamada al sistema open(). Normalmente, 0 significa sólo lectura, 1 sólo escritura y 2 lectura y escritura.

Devuelve un dbase_identifier del fichero abierto, o false si no pudo abrirse el fichero.

dbase_close (PHP 3, PHP 4)

cierra un fichero dBase

```
bool dbase_close (int dbase_identifier)
```

Cierra el fichero asociado con *dbase_identifier*.

dbase_pack (PHP 3, PHP 4)

"empaque" un fichero dBase

```
bool dbase_pack (int dbase_identifier)
```

Empaque el fichero especificado, borrando definitivamente todos los registros marcados con la función **dbase_delete_record()**.

dbase_add_record (PHP 3, PHP 4)

añade un registro a un fichero dBase

```
bool dbase_add_record (int dbase_identifier, array record)
```

Añade los datos de *record* a la base de datos. Si el número de elementos del registro proporcionado no es igual al número de campos de la base de datos, la operación fallará y la función devolverá false.

dbase_replace_record (PHP 3>= 3.0.11, PHP 4)

reemplaza un registro en un fichero dBase

```
bool dbase_replace_record (int dbase_identifier, array record, int dbase_record_number)
```

Reemplaza los datos asociados con el registro *record_number* con los datos de *record* en el fichero de datos. Si el número de elementos del registro proporcionado no es igual al número de campos de la base de datos, la operación fallará y la función devolverá false.

dbase_record_number es un entero en el rango de 1 al número de registros en el fichero de datos (devuelto por la función **dbase_numrecords()**).

dbase_delete_record (PHP 3, PHP 4)

borra un registro del fichero dBase

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

Marca el registro *record* para ser borrado del fichero de datos. Para eliminar realmente el registro del fichero, debe llamarse a la función **dbase_pack()**.

dbase_get_record (PHP 3, PHP 4)

lee un registro de un fichero dBase

```
array dbase_get_record (int dbase_identifier, int record)
```

Devuelve los datos del registro *record* en un array. El array se indexa a partir de 0, e incluye un elemento con el índice asociativo 'deleted', que vale 1 si el registro ha sido marcado para borrar (ver **dbase_delete_record()**).

Cada campo se convierte al tipo PHP apropiado. (Las fechas se guardan como cadenas.)

dbase_get_record_with_names (PHP 3>= 3.0.4, PHP 4)

lee un registro de un fichero dBase como array asociativo

```
array dbase_get_record_with_names (int dbase_identifier, int record)
```

Devuelve los datos del registro *record* en un array asociativo. El array incluye también un elemento con índice 'deleted' que vale 1 si el registro ha sido marcado para borrar (ver **dbase_delete_record()**).

Cada campo se convierte al tipo PHP apropiado. (Las fechas se transforman en cadenas.)

dbase_numfields (PHP 3, PHP 4)

cuenta el número de campos en un fichero dBase

```
int dbase_numfields (int dbase_identifier)
```

Devuelve el número de campos (columnas) en el fichero especificado. Los números de campo va de 0 a dbase_numfields(\$db)-1, mientras los números de registros van de 1 a dbase_numrecords(\$db).

Ejemplo 1. Uso de 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)

cuenta el número de registros en un fichero dBase

```
int dbase_numrecords (int dbase_identifier)
```

Devuelve el número de registros (filas) en el fichero especificado. Los números de registro van de 1 a dbase_numrecords(\$db), mientras los números de campo van de 0 a dbase_numfields(\$db)-1.

XVII. Funciones dbm

Estas funciones le permiten almacenar registros en una base de datos estilo dbm. Este tipo de base de datos (soportadas por las librerías db y gdbm de Berkeley, así como por algunas librerías del sistema y por una librería incluída para acceso a archivos de texto) guarda pares clave/valor (en oposición a los registros completos soportados por las bases de datos relacionales).

Ejemplo 1. ejemplo de dbm

```
$dbm = dbmopen("vistoya", "w");
if (dbmexists($dbm, $idusuario)) {
    $visto_ya = dbmfetch($dbm, $idusuario);
} else {
    dbminsert($dbm, $idusuario, time());
}
do_stuff();
dbmreplace($dbm, $idusuario, time());
dbmclose($dbm);
```


dbmopen (PHP 3, PHP 4)

abre una base de datos dbm

```
int dbmopen (string fichero, string indicadores)
```

El primer argumento es el nombre con sendero completo del archivo dbm que se va a abrir y el segundo es el modo de apertura, que puede ser "r", "n", "c" o "w", que significan sólo lectura, nuevo (implica lectura/escritura y suele truncar una base de datos si ya existía con ese nombre), crear (implica lectura/escritura, pero sin truncar la base de datos) y abrir para lectura/escritura, respectivamente.

Devuelve un identificador que se pasa al resto de funciones dbm si tiene éxito, o false si falla.

Si se utiliza el soporte de ndbm, este creará los archivos *fichero.dir* y *fichero.pag*. gdbm sólo utiliza un archivo y lo mismo hace el soporte interno de archivos de texto, mientras que el db de Berkeley crea un archivo *fichero.db*. Nótese que el PHP hace su propio bloqueo de archivo sobre el que pudiera realizar la propia librería dbm. El PHP no borra los archivos *.lck* que crea. Los utiliza simplemente como i-nodos fijos en los que hacer el bloqueo. Para más información sobre archivos dbm, vea las páginas man de su Unix o obtenga el gdbm de GNU desde <ftp://prep.ai.mit.edu/pub/gnu>.

dbmclose (PHP 3, PHP 4)

cierra una base de datos dbm

```
bool dbmclose (int identif_dbm)
```

Desbloquea y cierra la base de datos especificada.

dbmexists (PHP 3, PHP 4)

dice si existe un valor para una clave dada en la base de datos dbm

```
bool dbmexists (int identif_dbm, string clave)
```

Devuelve true si hay un valor asociado con la *clave*.

dbmfetch (PHP 3, PHP 4)

obtiene un valor para una clave desde la base de datos dbm

```
string dbmfetch (int identif_dbm, string clave)
```

Devuelve el valor asociado con la *clave*.

dbminsert (PHP 3, PHP 4)

inserta un valor para una clave en la base de datos dbm

```
int dbminsert (int identif_dbm, string clave, string valor)
```

Añade el valor a la base de datos con la clave especificada.

Devuelve -1 si la base de datos se abrió en modo sólo lectura, 0 si la inserción tuvo éxito y 1 si la clave ya existía (para sustituir el valor, utilice **dbmreplace()**.)

dbmreplace (PHP 3, PHP 4)

sustituye el valor de una clave en la base de datos dbm

```
bool dbmreplace (int identif_dbm, string clave, string valor)
```

Sustituye el valor para la clave especificada de la base de datos.

También añadirá la clave a la base de datos si no existía antes.

dbmdelete (PHP 3, PHP 4)

borra el valor de una clave de una base de datos dbm

```
bool dbmdelete (int identif_dbm, string clave)
```

Borra el valor para la *clave* en la base de datos.

Devuelve false si la clave no existía en la base de datos.

dbmfIRSTkey (PHP 3, PHP 4)

obtiene la primera clave de una base de datos dbm

```
string dbmfIRSTkey (int identif_dbm)
```

Devuelve la primera clave de la base de datos. Nótese que no se garantiza ningún orden en particular, pues la base de datos se crea utilizando una tabla hash, que no garantiza ordenación alguna.

dbmnEXTkey (PHP 3, PHP 4)

obtiene la siguiente clave de una base de datos dbm

```
string dbmnEXTkey (int identif_dbm, string clave)
```

Devuelve la clave que sigue a *clave*. Llamando a **dbmfIRSTkey()** seguida de llamadas sucesivas a **dbmnEXTkey()** se pueden visitar todos los pares clave/valor de la base de datos dbm. Por ejemplo:

Ejemplo 1. Visitando cada par clave/valor en una base de datos dbm.

```
$clave = dbmfIRSTkey($id_dbm);
while ($clave) {
    echo "$clave = " . dbmfetch($id_dbm, $clave) . "\n";
    $clave = dbmnEXTkey($id_dbm, $clave);
```

```
    $clave = dbmnextkey($id_dbm, $clave);  
}
```

dblist (PHP 3, PHP 4)

describe la librería compatible dbm que se está usando

```
string dblist (void)
```


XVIII. Funciones con directorios

chdir (PHP 3, PHP 4)

cambia de directorio

```
int chdir (string directory)
```

Cambia el directorio PHP actual a *directory*. Devuelve FALSE si no puede cambiar al directorio, TRUE si todo va bien.

dir (PHP 3, PHP 4)

clase directorio

```
new dir (string directory)
```

Un mecanismo semi-orientado a objetos para leer directorios. El parametro *directory* abre el directorio. Dos propiedades estan disponibles cuando el directorio ha sido abierto. La propiedad de manejo puede ser usada con otras funciones de directorios tal como **readdir()**, **rewinddir()** y **closedir()**. La propiedad de trayectoria (path) es fijada para encaminar el directorio que ha sido abierto. Tres metodos estan disponibles: leer, rebobinar y cerrar.

Ejemplo 1. Dir() Ejemplo

```
$d = dir("/etc");
echo "Handle: ".$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)

cierra el manejador de directorios

```
void closedir (int dir_handle)
```

Cierra la secuencia de directorio determinada por *dir_handle*. La secuencia debe de haber sido abierta previamente con **opendir()**.

opendir (PHP 3, PHP 4)

abre el manejador de directorios

```
int opendir (string path)
```

Devuelve un manejador de directorio para ser usado con las llamadas **closedir()**, **readdir()** y **rewinddir()**.

readdir (PHP 3, PHP 4)

lee las entradas del manejador de directorios

```
string readdir (int dir_handle)
```

Devuelve el nombre del siguiente fichero en el directorio. Los nombres de ficheros no son devueltos en ningun orden especial .

Ejemplo 1. Listar todos los ficheros en un directorio

```
<?php
$handle=opendir('.');
echo "Directory handle: $handle\n";
echo "Files:\n";
while ($file = readdir($handle)) {
    echo "$file\n";
}
closedir($handle);
?>
```

Tener en cuenta que **readdir()** devolvera tambien . y .. Si no quereis estas entradas podeis borrarlas:

Ejemplo 2. Listar todos los ficheros en un directorio excepto . y ..

```
<?php
$handle=opendir('.');
while ($file = readdir($handle)) {
    if ($file != "." && $file != "..") {
        echo "$file\n";
    }
}
closedir($handle);
?>
```

rewinddir (PHP 3, PHP 4)

rebobinar el manejador de directorios

```
void rewinddir (int dir_handle)
```

Inicializa la secuencia de directorio determinada por *dir_handle* al principio del directorio.

XIX. Funciones de DOM XML

Estas funciones son disponibles solamente si PHP fué configurado con `-with-dom=[DIR]`, usando al libreria de XML de GNOME. Usted va a necesitar como mínimo libxml-2.0.0 (la versión beta no trabajará). Estas funciones fueron añadidas en PHP4.

Este module define las siguientes constantes:

Tabla 1. Constantes de XML

Constante	Valor	Descripción
XML_ELEMENT_NODE	1	
XML_ATTRIBUTE_NODE	2	
XML_TEXT_NODE	3	
XML_CDATA_SECTION_NODE	4	
XML_ENTITY_REF_NODE	5	
XML_ENTITY_NODE	6	
XML_PI_NODE	7	
XML_COMMENT_NODE	8	
XML_DOCUMENT_NODE	9	
XML_DOCUMENT_TYPE_NODE	10	
XML_DOCUMENT_FRAG_NODE	11	
XML_NOTATION_NODE	12	
XML_GLOBAL_NAMESPACE	1	
XML_LOCAL_NAMESPACE	2	

Este modulo define un número de clases. Las funciones de DOM XML devuelven un árbol conteniendo la structura del documento XML, en el cual cada nodo es un objeto perteneciente a una de estas clases.

xmlDoc (PHP 4 >= 4.0b4)

Crea un objeto DOM representando un documento XML

```
object xmlDoc (string str)
```

Esta función analiza el documento XML contenido en el texto *str* y devuelve un objeto de clase "Documento Dom", que tiene las propiedades "doc"(resource), "version"(string) y "typo"(long).

xmlDocfile (PHP 4 >= 4.0b4)

Crea un objeto DOM a partir de un archivo XML

```
object xmlDocfile (string filename)
```

Esta función analiza el documento XML contenido en el archivo referido en *filename* y devuelve un objeto de clase "Documento Dom", que tiene las propiedades "doc"(resource), "version"(string) y "typo"(long).

xmlTree (PHP 4 >= 4.0b4)

Crea un árbol de objetos PHP a partir del un document XML

```
object xmlTree (string str)
```

Esta función analiza el documento XML contenido en el texto *str* y devuelve in árbol de objetos PHP representando el documento analizado.

XX. 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)

envía un mensaje de error a algún lugar

```
int error_log (string message, int message_type [, string destination [, string extra_headers]])
```

Envía un mensaje de error al log de errores del servidor web, a un puerto TCP o a un fichero. El primer parámetro, *message* (mensaje), es el mensaje de error que debe ser registrado. El segundo parámetro, *message_type* (tipo de mensaje) indica el lugar al que debe dirigirse:

Tabla 1. error_log() tipos de log

0	<i>message</i> es enviado al registro de sistema de PHP, utilizando el mecanismo de registro de sistema del Sistema Operativo, o a un fichero, dependiendo del valor de la directiva de configuración error_log
1	<i>message</i> es enviado por correo electrónico a la dirección del parámetro <i>destination</i> (destino). Este es el único tipo de mensaje donde se utiliza el cuarto parámetro, <i>extra_headers</i> . Este tipo de mensaje utiliza la misma funcionalidad interna que Mail() realiza.
2	<i>message</i> es enviado a través de la conexión de depuración de PHP. Esta opción está disponible sólo si la depuración remota ha sido activada. En este caso el parámetro <i>destination</i> especifica el nombre de host o dirección IP y, opcionalmente, el número de puerto del socket que recibe la información de depuración.
3	<i>message</i> es añadido al fichero <i>destination</i> .

Ejemplo 1. error_log() ejemplos

```
// 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)

establece que errores PHP son registrados

```
int error_reporting ([int level])
```

Establece el nivel de registro de los errores PHP y devuelve el nivel anterior. El nivel de registro es una máscara de bits de los valores siguientes (siga los enlaces a los valores internos para obtener sus significados):

Tabla 1. error_reporting() valores de bit

valor	nombre interno
1	E_ERROR
2	E_WARNING
4	E_PARSE
8	E_NOTICE
16	E_CORE_ERROR
32	E_CORE_WARNING

restore_error_handler (PHP 4 >= 4.0.1)

Restores the previous error handler function

```
void restore_error_handler (void)
```

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 >= 4.0.1)

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. The example below shows the handling of internal exceptions by triggering errors and handling them with a user defined function:

Ejemplo 1. Error handling with set_error_handler() and trigger_error()

```
<?php

// redefine the user error constants - PHP4 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, $errstr) {
    switch ($errno) {
        case FATAL:
            echo "<b>FATAL</b> [$errno] $errstr<br>\n";
            echo " Fatal error in line ".__LINE__." of file ".__FILE__;
            echo ", PHP ".PHP_VERSION." (" .PHP_OS .")<br>\n";
            echo "Aborting...<br>\n";
            exit -1;
        break;
        case ERROR:
            echo "<b>ERROR</b> [$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 undefined, you used: scale = $scale",
                      FATAL);
    if (!is_array($vect)) {
        trigger_error("Incorrect input vector, array of values expected", 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, using 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, using 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 16 of file trigger_error.php, PHP 4.0.1pl2 (Linux)<br>
Aborting...<br>

```

See also **error_reporting()**, **restore_error_handler()**, **trigger_error()**, **user_error()**

trigger_error (PHP 4 >= 4.0.1)

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()`). 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 >= 4.0RC2)

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()`

XXI. Funciones filePro

Estas funciones permiten acceso en modo de solo-lectura a datos guardados en bases de datos filePro.

filePro es una marca registrada de Fiserv, Inc. Mas informacion sobre filePro puede encontrarse en <http://www.fileproplus.com/>.

filepro (PHP 3, PHP 4)

lee y verifica el fichero de mapeo

```
bool filepro (string directory)
```

Lee y verifica el fichero de mapeo, guardando la relacion de campos e informacion.

Ningun bloqueo es realizado, por ello, no se deberia modificar la base de datos filePro cuando puede ser abierta con PHP.

filepro_fieldname (PHP 3, PHP 4)

obtiene el nombre de un campo

```
string filepro_fieldname (int field_number)
```

Devuelve el nombre del campo correspondiente a *field_number*.

filepro_fieldtype (PHP 3, PHP 4)

obtiene el tipo de campo

```
string filepro_fieldtype (int field_number)
```

Devuelve el tipo de campo del campo correspondiente a *field_number*.

filepro_fieldwidth (PHP 3, PHP 4)

obtiene la anchura de un campo

```
int filepro_fieldwidth (int field_number)
```

Devuelve la anchura de el campo correspondiente a *field_number*.

filepro_retrieve (PHP 3, PHP 4)

extrae informacion de una base de datos filePro

```
string filepro_retrieve (int row_number, int field_number)
```

Devuelve la informacion de la base de datos contenida en la localizacion especificada.

filepro_fieldcount (PHP 3, PHP 4)

encuentra cuantos campos existen en una base de datos filePro

```
int filepro_fieldcount(void);
```

Devuelve el numero de campos (columnas) existentes en la base de datos filePro abierta.

Ver tambien **filepro()**.

filepro_rowcount (PHP 3, PHP 4)

encuentra cuantas filas existen en una base de datos filePro

```
int filepro_rowcount(void);
```

Devuelve el numero de filas (entradas) existentes en la base de datos filePro abierta.

Ver tambien **filepro()**.

XXII. Funciones del sistema de ficheros

basename (PHP 3, PHP 4)

Devuelve la parte del path correspondiente al nombre del fichero

```
string basename (string path)
```

Dada una cadena (string) que contiene el path de un fichero, esta función devuelve el nombre base del fichero.

En Windows, tanto la barra (/) como la barra inversa (\) pueden usarse como carácter separador en el path. En otros entornos, se usa la barra directa (/).

Ejemplo 1. Ejemplo de basename()

```
$path = "/home/httpd/html/index.php3";
$file = basename($path); // $file toma el valor "index.php3"
```

Ver también: **dirname()**

chgrp (PHP 3, PHP 4)

Cambia el grupo de un fichero

```
int chgrp (string filename, mixed group)
```

Trata de cambiar el grupo al que pertenece el fichero *filename* al grupo *group*. Sólo el superusuario puede cambiar el grupo de un fichero arbitrariamente; otros usuarios pueden cambiar el grupo del fichero a cualquier grupo del cual el usuario sea miembro.

Devuelve true en caso de éxito; en otro caso devuelve false.

En Windows, no hace nada y devuelve true.

Ver también **chown()** y **chmod()**.

chmod (PHP 3, PHP 4)

Cambia permisos de un fichero

```
int chmod (string filename, int mode)
```

Trata de cambiar los permisos del fichero especificado por *filename* a los permisos dados por *mode*.

Cuidado que *mode* no es asumido de forma automática como un valor octal. Para asegurar que se hace la operación esperada necesitas anteponer un cero (0) como prefijo del parámetro *mode*:

```
chmod( "/somedir/somefile", 755 );    // decimal; probablemente incorrecto
chmod( "/somedir/somefile", 0755 );   // octal; valor correcto de mode
```

Devuelve true en caso de éxito y false en otro caso.

Ver también **chown()** y **chgrp()**.

chown (PHP 3, PHP 4)

Cambia el propietario de un fichero

```
int chown (string filename, mixed user)
```

Trata de cambiar el propietario del fichero *filename* al usuario *user*. Sólo el superusuario puede cambiar el propietario de un fichero.

Devuelve true en caso de éxito; en otro caso devuelve false.

Nota: En Windows, no hace nada y devuelve true.

Ver también **chown()** y **chmod()**.

clearstatcache (PHP 3, PHP 4)

Limpia la cache de estado de un fichero

```
void clearstatcache(void);
```

Invocar la llamada del sistema stat o lstat es bastante costoso en la mayoría de los sistemas. Por lo tanto, el resultado de la última llamada a cualquiera de las funciones de estado (listadas abajo) es guardado para usarlo en la próxima llamada de este tipo empleando el mismo nombre de fichero. Si deseas forzar un nuevo chequeo del estado del fichero, por ejemplo si el fichero está siendo chequeado muchas veces y puede cambiar o desaparecer, usa esta función para borrar los resultados almacenados en memoria de la última llamada.

Este valor sólo es cacheado durante el tiempo de vida de una petición simple.

Entre las funciones afectadas se incluyen **stat()**, **lstat()**, **file_exists()**, **is_writeable()**, **is_readable()**, **is_executable()**, **is_file()**, **is_dir()**, **is_link()**, **filectime()**, **fileatime()**, **filemtime()**, **fileinode()**, **filegroup()**, **fileowner()**, **filesize()**, **filetype()**, y **fileperms()**.

copy (PHP 3, PHP 4)

Copia un fichero

```
int copy (string source, string dest)
```

Hace una copia de un fichero. Devuelve true si la copia tiene éxito, y false en otro caso.

Ejemplo 1. Ejemplo de copy()

```
if (!copy($file, $file.'.bak')) {
    print("failed to copy $file...<br>\n");
}
```

Ver también: **rename()**.

delete (PHP 4 CVS only)

Una entrada manual inútil

```
void delete (string file)
```

Esto es una entrada manual inútil para satisfacer a esas personas que están buscando **unlink()** o **unset()** en el lugar equivocado.

Ver también: **unlink()** para borrar ficheros, **unset()** para borrar variables.

dirname (PHP 3, PHP 4)

Devuelve la parte del path correspondiente al directorio

```
string dirname (string path)
```

Dada una cadena (string) conteniendo el path a un fichero, esta función devolverá el nombre del directorio.

En Windows, tanto la barra (/) como la barra inversa (\) son usadas como separadores de caracteres. En otros entornos, debe usarse la barra directa (/).

Ejemplo 1. Ejemplo de dirname()

```
$path = "/etc/passwd";
$file = dirname($path); // $file toma el valor "/etc"
```

Ver también: **basename()**

diskfreespace (PHP 3>= 3.0.7, PHP 4 >= 4.0b4)

Devuelve el espacio disponible en un directorio

```
float diskfreespace (string directory)
```

Dada una cadena (string) conteniendo el nombre de un directorio, esta función devolverá el número de bytes disponibles en el disco correspondiente.

Ejemplo 1. Ejemplo de diskfreespace()

```
$df = diskfreespace("/"); // $df contiene el numero de bytes
// disponibles en "/"
```

fclose (PHP 3, PHP 4)

Cierra el apuntador a un fichero abierto

```
int fclose (int fp)
```

Se cierra el fichero apuntado por fp.

Devuelve true en caso de éxito y false en caso de fallo.

El apuntador al fichero debe ser válido y debe apuntarse a un fichero abierto con éxito con **fopen()** o con **fsckopen()**.

feof (PHP 3, PHP 4)

Verifica si el apuntador a un fichero está al final del fichero (end-of-file)

```
int feof (int fp)
```

Devuelve true si el apuntador del fichero está en EOF o si ocurre un error; en otro caso devuelve false.

The file pointer must be valid, and must point to a file El apuntador al fichero debe ser válido, y debe apuntar a un fichero abierto con éxito por **fopen()**, **popen()**, o **fsckopen()**.

fgetc (PHP 3, PHP 4)

Obtiene un carácter del fichero apuntado

```
string fgetc (int fp)
```

Devuelve una cadena (string) conteniendo un simple carácter leído del fichero apuntado por fp. Devuelve FALSE para EOF (como hace **feof()**).

El apuntador al fichero debe ser válido, y debe apuntar a un fichero abierto con éxito por **fopen()**, **popen()**, o **fsckopen()**.

Ver también **fread()**, **fopen()**, **popen()**, **fsckopen()**, y **fgets()**.

fgetcsv (PHP 3>= 3.0.8, PHP 4)

Obtiene una línea del fichero apuntado y extrae los campos CSV

```
array fgetcsv (int fp, int length [, string delimiter])
```

Parecida a **fgets()** excepto que **fgetcsv()** parsea la línea que lee buscando campos en formato CSV y devuelve un array conteniendo los campos leídos. El delimitador de campo es una coma, a menos que se especifique otro delimitador con el tercer parámetro opcional.

fp debe ser un apuntador válido a un fichero abierto con éxito por **fopen()**, **popen()**, o **fsckopen()**

la longitud debe ser mayor que la línea más larga que pueda encontrarse en el fichero CSV (permitiendo arrastrar caracteres de fin de línea)

fgetcsv() devuelve false en caso de error, incluyendo el fin de fichero.

NOTA: Una línea en blanco en un fichero CSV se devuelve como un array que contiene un único campo nulo, y esto no será tratado como un error.

Ejemplo 1. Ejemplo de Fgetcsv() - Leer e imprimir el contenido completo de un fichero CSV

```
$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)

Obtiene una línea del fichero apuntado

```
string fgets (int fp, int length)
```

Devuelve una cadena de como mucho length - 1 bytes leidos del fichero apuntado por fp. La lectura acaba cuando son leídos length - 1 bytes, cuando se llega a una nueva línea (el carácter de nueva línea se incluye en el valor devuelto), o cuando se llega a un EOF (lo que ocurría primero).

Si ocurre un error, devuelve false.

Fallos Comunes:

Los que hayan usado la semántica de 'C' de la función fgets deben darse cuenta de la diferencia que hay en como el EOF es devuelto por esta función.

El apuntador al fichero debe ser válido, y debe apuntar a un fichero abierto con éxito con **fopen()**, **popen()**, o **fsckopen()**.

A continuación un ejemplo sencillo:

Ejemplo 1. Leyendo un fichero línea por línea

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

Ver también **fread()**, **fopen()**, **popen()**, **fgetc()**, y **fsckopen()**.

fgetss (PHP 3, PHP 4)

Obtiene una línea del fichero apuntado y quita las etiquetas HTML

```
string fgetss (int fp, int length [, string allowable_tags])
```

Idéntica a **fgets()**, excepto que fgetss trata de quitar cualquier etiqueta HTML y PHP del texto que lee.

Se puede utilizar el tercer parámetro opcional para especificar etiquetas que no deben de quitarse.

Nota: *allowable_tags* fue añadido en PHP 3.0.13, PHP4B3.

Ver también **fgets()**, **fopen()**, **fsckopen()**, **popen()**, y **strip_tags()**.

file (PHP 3, PHP 4)

lee un fichero completo hacia un array

```
array file (string filename [, int use_include_path])
```

Idéntica a **readfile()**, excepto que **file()** devuelve el fichero en un array. Cada elemento del array corresponde a una línea del fichero, con el carácter de nueva línea incluido.

Se puede utilizar el segundo parámetro opcional y ponerle el valor "1", si también se quiere buscar el fichero en el [include_path](#).

Ver también **readfile()**, **fopen()**, y **popen()**.

file_exists (PHP 3, PHP 4)

Verifica si un fichero existe

```
int file_exists (string filename)
```

Devuelve true si el fichero especificado por *filename* existe; y false en otro caso.

El resultado de esta función es cacheado. Ver **clearstatcache()** para más detalles.

fileatime (PHP 3, PHP 4)

Obtiene la última fecha de acceso a un fichero

```
int fileatime (string filename)
```

Devuelve la fecha a la que el fichero fue accedido por última vez, o false en caso de error.

Los resultados de esta función son cacheados. Ver **clearstatcache()** para más detalles.

filectime (PHP 3, PHP 4)

Obtiene la fecha de cambio del inode del fichero

```
int filectime (string filename)
```

Devuelve el momento en el que el fichero fue cambiado por última vez, o false en caso de error.
Los resultados de esta función son cacheados. Ver **clearstatcache()** para más detalles.

filegroup (PHP 3, PHP 4)

Obtiene el grupo de un fichero

```
int filegroup (string filename)
```

Devuelve el identificador (ID) de grupo del propietario del fichero, o false en caso de un error. El ID del grupo es devuelto en formato numérico, usar **posix_getgrgid()** para obtener el nombre del grupo.

Los resultados de esta función son cacheados. Ver **clearstatcache()** para más detalles.

fileinode (PHP 3, PHP 4)

Obtiene el inode del fichero

```
int fileinode (string filename)
```

Devuelve el número de inode del fichero, o false en caso de un error.

Los resultados de esta función son cacheados. Ver **clearstatcache()** para más detalles.

filemtime (PHP 3, PHP 4)

Obtiene la fecha de modificación del fichero

```
int filemtime (string filename)
```

Devuelve el momento en el que el fichero fue modificado por última vez, o false en caso de un error.

Los resultados de esta función son cacheados. Ver **clearstatcache()** para más detalles.

fileowner (PHP 3, PHP 4)

Obtiene el propietario del fichero

```
int fileowner (string filename)
```

Devuelve el identificador (ID) de usuario del propietario del fichero, o false en caso de error. El ID de usuario se devuelve en formato numérico, usar **posix_getpwuid()** para obtener el nombre del usuario.

Los resultados de esta función son cacheados. Ver **clearstatcache()** para más detalles.

fileperms (PHP 3, PHP 4)

Obtiene los permisos del fichero

```
int fileperms (string filename)
```

Devuelve los permisos del fichero, o false en caso de error.

Los resultados de esta función son cacheados. Ver **clearstatcache()** para más detalles.

filesize (PHP 3, PHP 4)

Obtiene el tamaño del fichero

```
int filesize (string filename)
```

Devuelve el tamaño del fichero, o false en caso de error.

Los resultados de esta función son cacheados. Ver **clearstatcache()** para más detalles.

filetype (PHP 3, PHP 4)

Obtiene el tipo de fichero

```
string filetype (string filename)
```

Devuelve el tipo de fichero. Valores posibles son fifo, char, dir, block, link, file, y unknown.

Devuelve false si ocurre un error.

Los resultados de esta función son cacheados. Ver **clearstatcache()** para más detalles.

flock (PHP 3>= 3.0.7, PHP 4)

Bloqueo de ficheros portable y asesorado

```
bool flock (int fp, int operation)
```

PHP soporta un método portable de bloquear ficheros completos de manera asesorada (lo que significa que todos los programas que acceden tienen que usar el mismo modo de bloqueo o éste no funcionará).

flock() opera sobre *fp* el cual debe ser un apuntador a un fichero abierto. *operation* toma uno de los siguientes valores:

- Para que adquiera un bloqueo compartido (lectura), fija *operation* a 1.
- Para adquirir un bloqueo exclusivo (escritura), fija *operation* a 2.
- Para liberar un bloqueo (compartido o exclusivo), fija *operation* a 3.
- Si no quieres que **flock()** bloquee mientras está activado, suma 4 al valor de *operation*.

flock() permite establecer un modelo simple de lectura/escritura el cual puede usarse en prácticamente cualquier plataforma (incluyendo la mayoría de sistemas Unix e incluso Windows).

flock() devuelve true en caso de éxito y false en caso de error (ej. cuando no se puede establecer un bloqueo).

fopen (PHP 3, PHP 4)

Abre un fichero o una URL

```
int fopen (string filename, string mode [, int use_include_path])
```

Si *filename* comienza con "http://" (no es sensible a mayúsculas), se abre una conexión HTTP 1.0 hacia el servidor especificado y se devuelve un apuntador de fichero al comienzo del texto de respuesta.

No maneja redirecciones HTTP, por eso se debe incluir una barra final cuando se trata de directorios.

Si *filename* comienza con "ftp://" (no es sensible a mayúsculas), se abre una conexión ftp hacia el servidor especificado y se devuelve un apuntador al fichero requerido. Si el servidor no soporta ftp en modo pasivo, esto fallará. Se pueden abrir fichero via ftp para leer o para escribir (pero no ambas cosas simultáneamente).

Si *filename* no comienza con nada de lo anterior, el fichero se abre del sistema de ficheros, y se devuelve un apuntador al fichero abierto.

Si el abrir el fichero falla, la función devuelve false.

mode puede ser cualquiera de lo siguiente:

- 'r' - Abre para sólo lectura; sitúa el apuntador del fichero al comienzo del mismo.
- 'r+' - Abre para lectura y escritura; sitúa el apuntador del fichero al comienzo del fichero.
- 'w' - Abre para sólo escritura; sitúa el apuntador del fichero al comienzo del fichero y trunca el fichero con longitud cero. Si el fichero no existe, trata de crearlo.
- 'w+' - Abre el fichero para lectura y escritura; sitúa el apuntador del fichero al comienzo del fichero y trunca el fichero con longitud cero. Si el fichero no existe, trata de crearlo.
- 'a' - Abre sólo para escribir (añadir); sitúa el apuntador del fichero al final del mismo. Si el fichero no existe, trata de crearlo.
- 'a+' - Abre para lectura y escritura (añadiendo); sitúa el apuntador del fichero al final del mismo. Si el fichero no existe, trata de crearlo.

Además, *mode* puede contener la letra 'b'. Esto es útil para sistemas que diferencian entre ficheros binarios y de texto (ej. es inútil en Unix). Si no se necesita, será ignorado.

Puede usarse el tercer parámetro opcional y fijarlo a "1", si también se quiere buscar el fichero en el [include_path](#).

Ejemplo 1. Ejemplo de fopen()

```
$fp = fopen( "/home/rasmus/file.txt" , "r" );
$fp = fopen( "http://www.php.net/" , "r" );
$fp = fopen( "ftp://user:password@example.com/" , "w" );
```

Si experimentas problemas a la hora de leer y escribir a ficheros y estas usando la versión de PHP como módulo para el servidor, recuerda que debes asegurar que los ficheros y directorios que estas usando son accesibles al proceso servidor.

En la plataforma Windows, ten cuidado de escribir correctamente las barras invertidas en el path del fichero (poniéndolas dobles), o usa barras directas.

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

Ver también **fclose()**, **fsockopen()**, y **popen()**.

fpassthru (PHP 3, PHP 4)

Saca todos los datos restantes del fichero apuntado

```
int fpassthru (int fp)
```

Lee hasta el EOF del fichero apuntado y escribe los resultados a la salida estándar (stdout).

Si ocurre un error, **fpassthru()** devuelve false.

El apuntador al fichero debe ser válido, y debe apuntar a un fichero abierto con éxito por **fopen()**, **popen()**, o **fsockopen()**. El fichero se cierra cuando **fpassthru()** termina de leerlo (dejando *fp* sin ninguna utilidad).

Si sólo quieras volcar el contenido de un fichero a stdout puedes If you just want to dump the contents of a file to stdout you may usar la función **readfile()**, la cual te libra de la llamada a **fopen()**.

Ver también **readfile()**, **fopen()**, **popen()**, y **fsockopen()**

fputs (PHP 3, PHP 4)

Escribe en el fichero apuntado

```
int fputs (int fp, string str [, int length])
```

fputs() es un alias de **fwrite()**, y es idéntico a él en todo. Notar que el parámetro *length* es opcional y si no se pone la cadena entera será escrita.

fread (PHP 3, PHP 4)

Lee ficheros en plan binario

```
string fread (int fp, int length)
```

fread() lee hasta *length* bytes del apuntador de fichero referenciado por *fp*. La lectura acaba cuando *length* bytes se han leido o se alcanza EOF, lo que ocurra primero.

```
// Mete el contenido de un fichero en una cadena
$filename = "/usr/local/something.txt";
$fd = fopen ($filename, "r");
$contents = fread ($fd, filesize ($filename));
fclose ($fd);
```

Ver también **fwrite()**, **fopen()**, **fsockopen()**, **popen()**, **fgets()**, **fgetss()**, **file()**, y **fpassthru()**.

fseek (PHP 3, PHP 4)

Sitúa el apuntador a un fichero

```
int fseek (int fp, int offset)
```

Fija el indicador de posición del fichero referenciado por *fp* a tantos bytes como indica *offset*. Es equivalente a la llamada (en C) `fseek(fp, offset, SEEK_SET)`.

Si va bien, devuelve 0; en otro caso, devuelve -1. Tener en cuenta que situarse más alla de EOF no se considera un error.

No puede usarse sobre apuntadores de ficheros devueltos por **fopen()** si usan los formatos "http://"or "ftp://".

Ver también **ftell()** y **rewind()**.

ftell (PHP 3, PHP 4)

Pregunta por la posición del apuntador de lectura/escritura de un fichero

```
int ftell (int fp)
```

Devuelve la posición del apuntador de fichero referenciado por *fp*; es decir, la distancia en la secuencia del fichero.

Si ocurre un error, devuelve false.

El apuntador al fichero debe ser válido, y debe referirse a The file pointer must be valid, and must point to a file un fichero abierto con éxito por **fopen()** o **popen()**.

Ver también **fopen()**, **popen()**, **fseek()** y **rewind()**.

fwrite (PHP 3, PHP 4)

Escribe ficheros en plan binario

```
int fwrite (int fp, string string [, int length])
```

fwrite() escribe el contenido de *string* al fichero apuntado por *fp*. Si se da el argumento *length*, la escritura acaba antes de que *length* bytes sean escritos o se alcance el final de *string*, lo que ocurra primero.

Tener en cuenta que si se da el argumento *length*, entonces la opción de configuración **magic_quotes_runtime** será ignorada y los caracteres de barra no se quitarán de la cadena *string*.

Ver también **fread()**, **fopen()**, **fsckopen()**, **popen()**, y **fputs()**.

set_file_buffer (PHP 3>= 3.0.8, PHP 4 >= 4.0.1)

Fija el buffer de fichero del fichero apuntado

```
int fwrite (int fp, int buffer)
```

set_file_buffer() fija el buffer para operaciones de escritura en el apuntador de fichero *fp* con *buffer* bytes. Si *buffer* es 0 entonces las operaciones de escritura no usan un buffer intermedio.

La función devuelve 0 en caso de éxito, o EOF si la petición no se puede realizar.

Tener en cuenta que por defecto cualquier fopen hace una llamada a set_file_buffer de 8K.

Ver también **fopen()**.

is_dir (PHP 3, PHP 4)

Dice si el fichero nombrado es un directorio

```
bool is_dir (string filename)
```

Devuelve true si el nombre del fichero existe y es un directorio.

Los resultados de esta función son cacheados. Ver **clearstatcache()** para más detalles.

Ver también **is_file()** y **is_link()**.

is_executable (PHP 3, PHP 4)

Dice si el fichero nombrado es ejecutable

```
bool is_executable (string filename)
```

Devuelve true si el fichero indicado existe y es ejecutable.

Los resultados de esta función son cacheados. Ver **clearstatcache()** para más detalles.

Ver también **is_file()** y **is_link()**.

is_file (PHP 3, PHP 4)

Dice si el fichero nombrado es un fichero regular

```
bool is_file (string filename)
```

Devuelve true si el fichero nombrado existe y es un fichero regular.

Los resultados de esta función son cacheados. Ver **clearstatcache()** para más detalles.

Ver también **is_dir()** y **is_link()**.

is_link (PHP 3, PHP 4)

Dice si el fichero indicado es un enlace simbólico

```
bool is_link (string filename)
```

Devuelve true si el fichero indicado existe y es un enlace simbólico.

Los resultados de esta función son cacheados. Ver **clearstatcache()** para más detalles.

Ver también **is_dir()** y **is_file()**.

is_readable (PHP 3, PHP 4)

Dice si el fichero indicado se puede leer

```
bool is_readable (string filename)
```

Devuelve true si el fichero indicado existe y se puede leer.

Recuerda que PHP puede acceder al fichero con el identificador de usuario con el que el servidor web se ejecuta (a menudo 'nobody'). No se tienen en cuenta las limitaciones de modos seguros.

Los resultados de esta función son cacheados. Ver **clearstatcache()** para más detalles.

Ver también **is_writeable()**.

is_writeable (PHP 3, PHP 4)

Dice si se puede escribir en el fichero indicado

```
bool is_writeable (string filename)
```

Devuelve true si el fichero indicado existe y se puede escribir en él. El argumento *filename* puede ser el nombre de un directorio, lo que permite verificar si un directorio tiene permiso de escritura.

Recuerda que PHP puede acceder al fichero con el identificador de usuario con el que el servidor web se ejecuta (a menudo 'nobody'). No se tienen en cuenta las limitaciones de modos seguros.

Los resultados de esta función son cacheados. Ver **clearstatcache()** para más detalles.

Ver también **is_readable()**.

link (PHP 3, PHP 4)

Crea un enlace fuerte

```
int link (string target, string link)
```

Link() crea un enlace fuerte.

Ver también **symlink()** para crear enlaces débiles, y **readlink()** junto con **linkinfo()**.

linkinfo (PHP 3, PHP 4)

Consigue información sobre un enlace

```
int linkinfo (string path)
```

Linkinfo() da el campo st_dev de la estructura stat de UNIX C devuelto por la llamada al sistema lstat. Esta función se usa para verificar si un enlace (apuntado por *path*) existe realmente (usando el mismo método que la macro S_ISLNK definida en stat.h). Devuelve 0 o FALSE en caso de error.

Ver también **symlink()**, **link()**, y **readlink()**.

mkdir (PHP 3, PHP 4)

Crea un directorio

```
int mkdir (string pathname, int mode)
```

Trata de crear el directorio especificado por pathname.

Ten en cuenta que debes especificar el modo como un número octal, lo que significa que debes anteponerle un 0 al número.

```
mkdir ("/path/to/my/dir", 0700);
```

Devuelve true en caso de éxito y false en caso de fallo.

Ver también **rmdir()**.

pclose (PHP 3, PHP 4)

Cierra el fichero de proceso apuntado

```
int pclose (int fp)
```

Cierra un fichero que representa un tubería (pipe) abierta con **popen()**.

El apuntador al fichero debe ser válido, y debe haber sido devuelto por una llamada con éxito a **popen()**.

Devuelve el estado de terminación del proceso que estaba ejecutándose.

Ver también **popen()**.

popen (PHP 3, PHP 4)

Abre el fichero de proceso apuntado

```
int popen (string command, string mode)
```

Abre una tubería (pipe) a un proceso ejecutado haciendo fork al comando dado por command

Devuelve un apuntador de fichero idéntico al devuelto por **fopen()**, excepto que este es unidireccional (sólo puede usarse o para leer o para escribir) y debe cerrarse con **pclose()**. Este apuntador puede usarse con **fgets()**, **fgetss()**, y **fputs()**.

Si ocurre un error, devuelve false.

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

Ver también [pclose\(\)](#).

readfile (PHP 3, PHP 4)

Muestra el contenido de un fichero

```
int readfile (string filename [, int use_include_path])
```

Lee un fichero y lo escribe a la salida estándar.

Devuelve el número de bytes leidos del fichero. Si ocurre un error, se devuelve false y a menos que la función fuera llamada como @readfile, se imprime un mensaje de error

Si *filename* comienza por "http://" (no es sensible a mayúsculas), se abre una conexión HTTP 1.0 al servidor especificado y el texto de la respuesta se escribe a la salida estándar.

No maneja redirecciones HTTP, por eso se debe incluir una barra final cuando se trata de directorios.

Si *filename* comienza con "ftp://" (no es sensible a mayúsculas), se abre una conexión ftp al servidor especificado y el fichero que se pide se escribe en la salida estándar. Si el servidor no soporta ftp en modo pasivo, la función fallará.

Si *filename* no comienza con ninguna de las cadenas anteriores, el fichero será abierto del sistema de ficheros y su contenido escrito en la salida estándar.

Se puede usar el segundo parámetro opcional y fijarlo a "1", si si quieres que también se busque el fichero en el [include_path](#).

Ver también [fpassthru\(\)](#), [file\(\)](#), [fopen\(\)](#), [include\(\)](#), [require\(\)](#), y [virtual\(\)](#).

readlink (PHP 3, PHP 4)

Devuelve el objetivo de un enlace simbólico

```
string readlink (string path)
```

Readlink() hace lo mismo que la función C `readlink` C y devuelve el contenido del path del enlace simbólico o 0 en caso de error.

Ver también [symlink\(\)](#), [readlink\(\)](#) y [linkinfo\(\)](#).

rename (PHP 3, PHP 4)

Renombra un fichero

```
int rename (string oldname, string newname)
```

Trata de renombrar *oldname* como *newname*.

Devuelve true en caso de éxito y false en caso de fallo.

rewind (PHP 3, PHP 4)

Rebobina la posición del apuntador al fichero

```
int rewind (int fp)
```

Fija el indicador de posición del fichero dado por fp al comienzo de del fichero.

Si ocurre un error, devuelve 0.

El apuntador al fichero debe ser válido, y debe apuntar a un fichero abierto con éxito por **fopen()**.

Ver también **fseek()** y **ftell()**.

rmdir (PHP 3, PHP 4)

Elimina un directorio

```
int rmdir (string dirname)
```

Trata de eliminar el directorio indicado por pathname. El directorio debe estar vacío, y los permisos relevantes deben permitir esto.

Si ocurre un error, devuelve 0.

Ver también **mkdir()**.

stat (PHP 3, PHP 4)

Da información sobre un fichero

```
array stat (string filename)
```

Recoje los datos sobre el fichero indicado por filename.

Devuelve un array conteniendo los datos del fichero con los siguientes elementos:

1. dispositivo (device)
2. inode
3. modo de protección del inode
4. número de enlaces
5. id de usuario del propietario
6. id de grupo del propietario
7. tipo de dispositivo si es un inode device *
8. tamaño en bytes
9. fecha del último acceso access
10. fecha de la última modificación
11. fecha del último cambio
12. tamaño del bloque para el sistema I/O *

13. número de bloques ocupados

* - sólo válido en sistemas que soportan el tipo st_blksize –otros sistemas (como Windows) devuelven -1

Los resultados de esta función son cacheados. Ver **clearstatcache()** para más detalles.

lstat (PHP 3>= 3.0.4, PHP 4)

Da información sobre un fichero o enlace simbólico

```
array lstat (string filename)
```

Reúne los datos del fichero o enlace simbólico indicado por *filename*. Esta función es idéntica a la función **stat()** excepto que si el nombre en el parámetro *filename* es un enlace simbólico, son devueltos los datos (status) del enlace simbólico, y no los del fichero al que apunta el enlace simbólico.

Devuelve un array conteniendo los datos del fichero con los siguientes elementos:

1. dispositivo (device)
2. inode
3. número de enlaces
4. id de usuario del propietario
5. id de grupo del propietario
6. tipo de dispositivo si es un inode device *
7. tamaño en bytes
8. fecha del último acceso
9. fecha de la última modificación
10. fecha del último cambio
11. tamaño de bloque para el sistema I/O *
12. número de bloques ocupados

* - sólo válido en sistemas que soportan el tipo st_blksize –otros sistemas (como Windows) devuelven -1

Los resultados de esta función son cacheados. Ver **clearstatcache()** para más detalles.

symlink (PHP 3, PHP 4)

Crea un enlace simbólico

```
int symlink (string target, string link)
```

symlink() crea un enlace simbólico del objetivo *target* con el nombre especificado por *link*.

Ver también **link()** para crear enlaces fuertes, y **readlink()** junto con **linkinfo()**.

tempnam (PHP 3, PHP 4)

Crea un fichero de nombre único

```
string tmpnam (string dir, string prefix)
```

Crea un fichero temporal de nombre único en el directorio especificado. Si el directorio no existe **tmpnam()** puede generar un fichero en el directorio temporal del sistema.

El comportamiento de la función **tmpnam()** depende del sistema. En Windows la variable de entorno TMP se impone sobre el parámetro *dir*, en Linux la variable de entorno TMPDIR tiene preferencia, mientras que en SVR4 siempre se usará el parámetro *dir* si el directorio al que apunta existe. Consulta la documentación del sistema sobre la función **tmpnam(3)** en caso de duda.

Devuelve el nombre del nuevo fichero temporal, o una cadena nula en caso de fallo.

Ejemplo 1. Ejemplo de Tempnam()

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

touch (PHP 3, PHP 4)

Fija la fecha de modificación de un fichero

```
int touch (string filename, int time)
```

Trata de fijar la fecha de modificación del fichero indicado por *filename* al valor dado por *time*. Si no se pone la opción *time*, se utiliza la fecha actual.

Si el fichero no existe, será creado.

Devuelve true en caso de éxito y false en otro caso.

umask (PHP 3, PHP 4)

Cambia la umask actual

```
int umask (int mask)
```

Umask() fija las umask PHP con la mascara & 0777 y devuelve la antigua umask. Cuando PHP se está usando como un módulo del servidor, la umask se restaura cuando cada petición es finalizada.

Umask() sin argumentos sólamente devuelve la umask actual.

unlink (PHP 3, PHP 4)

Borra un fichero

```
int unlink (string filename)
```

Borra el fichero *filename*. Es similar a la función **unlink()** del Unix C.

Devuelve 0 o FALSE en caso de error.

Ver también **rmdir()** para borrar directorios.

XXIII. Funciones Forms Data Format (Formato de Datos de Formularios)

El Formato de Datos de Formulario (FDF) está diseñado para el manejo de formularios en archivos PDF. Se aconseja leer la información disponible en <http://partners.adobe.com/asn/developer/acrosdk/forms.html> para más información sobre lo que es FDF y cómo se usa en general.

Nota: Actualmente Adobe sólo proporciona una versión compatible con libc5 para Linux. Las pruebas con glibc2 provocaron un fallo de segmentado. Si alguien es capaz de hacerla funcionar, por favor coméntelo en esta página.

Nota: Si tiene problemas configurando php con soporte de fdftk, compruebe si el archivo de cabecera FdfTk.h y la librería libFdfTk.so están en su lugar correcto. Deberían encontrarse respectivamente en fdftk-dir/include y en fdftk-dir/lib. Este problema no se dará si se limita a desempaquetar la distribución del FdfTk.

La idea general del FDF es similar a los formularios HTML. La diferencia básicamente está en el formato en que se transmiten los datos al servidor cuando se pulsa el botón de envío (este es realmente el Formato de Datos de Formulario) y el formato del formulario en sí mismo (que es el Formato de Documento Portable, PDF). Procesar los datos del FDF es una de las características que proporcionan las funciones fdf. Pero aún hay más. Uno también puede tomar un formulario PDF y llenar los campos de entrada con datos sin modificar el formulario en sí mismo. En dicho caso, lo que se hace es crear un documento FDF (**fdf_create()**), fijar los valores de cada campo de entrada (**fdf_set_value()**) y asociarlo con un formulario PDF (**fdf_set_file()**). Finalmente, debe ser enviado al navegador con el MimeType application/vnd.fdf. El plug-in de Acrobat reader de su navegador reconoce el MimeType, lee el formulario PDF asociado y rellena los datos a partir del documento FDF.

Los siguientes ejemplos muestran cómo se evalúan los datos de los formularios.

Ejemplo 1. Evaluando un documento FDF

```
<?php
// Guarda los datos FDF en un archivo temporal
$fdffp = fopen("test.fdf", "w");
fwrite($fdffp, $HTTP_FDF_DATA, strlen($HTTP_FDF_DATA));
fclose($fdffp);

// Abre archivo temporal y evalúa los datos
// El formulario pdf contenía varios campos de texto con los nombres
// volumen, fecha, comentario, editorial, preparador, y dos casillas de verificación
// muestra_editorial y muestra_preparador.
$fdf = fdf_open("test.fdf");
$volume = fdf_get_value($fdf, "volumen");
echo "El campo volumen tiene el valor '<B>$volume</B>'<BR>';

$date = fdf_get_value($fdf, "fecha");
echo "El campo fecha tiene el valor '<B>$date</B>'<BR>';

$comment = fdf_get_value($fdf, "comentario");
echo "El campo comentario tiene el valor '<B>$comment</B>'<BR>';

if(fdf_get_value($fdf, "muestra_editorial") == "On") {
    $publisher = fdf_get_value($fdf, "editorial");
    echo "El campo editorial tiene el valor '<B>$publisher</B>'<BR>";
} else
    echo "No se debe mostrar la editorial.<BR>';

if(fdf_get_value($fdf, "muestra_preparador") == "On") {
    $preparer = fdf_get_value($fdf, "preparador");
    echo "El campo preparador tiene el valor '<B>$preparer</B>'<BR>';
```

```
echo "El campo preparador tiene el valor '<B>$preparer</B>'<BR>" ;
} else
    echo "No se debe mostrar el preparador.<BR>" ;
fdf_close($fdf);
?>
```

fdf_open (PHP 3>= 3.0.6, PHP 4)

Abrir un documento FDF

```
int fdf_open (string filename)
```

La función **fdf_open()** abre un archivo con datos de formulario. Este archivo debe contener los datos tal y como se devuelven en un formulario PDF. Actualmente dicho archivo debe crearse "manualmente" usando la función **fopen()** y escribiendo en éste el contenido de **HTTP_FDF_DATA** usando **fwrite()**. No existe un mecanismo similar al de los formularios HTML donde se crea una variable para cada campo de entrada.

Ejemplo 1. Accediendo a los datos del formulario

```
<?php
// Guarda los datos FDF en un archivo temporal
$fdffp = fopen("test.fdf", "w");
fwrite($fdffp, $HTTP_FDF_DATA, strlen($HTTP_FDF_DATA));
fclose($fdffp);

// Abre archivo temporal y evalúa los datos
$fdf = fdf_open("test.fdf");
...
fdf_close($fdf);
?>
```

Vea también **fdf_close()**.

fdf_close (PHP 3>= 3.0.6, PHP 4)

Cerrar un documento FDF

```
void fdf_close (int fdf_document)
```

La función **fdf_close()** cierra un documento FDF.

Vea también **fdf_open()**.

fdf_create (PHP 3>= 3.0.6, PHP 4)

Crear un documento FDF

```
int fdf_create (void )
```

La función **fdf_create()** crea un documento FDF nuevo. Esta función se necesita si se desea llenar los campos de entrada en un documento PDF.

Ejemplo 1. Rellenando un documento PDF

```
<?php
$outfdf = fdf_create();
```

```
fdf_set_value($outfdf, "volumen", $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");
?>
```

Vea también **fdf_close()**, **fdf_save()**, **fdf_open()**.

fdf_save (PHP 3>= 3.0.6, PHP 4)

Guardar un documeto FDF

```
int fdf_save (string filename)
```

La función **fdf_save()** guarda un documento FDF. El kit de FDF proporciona una forma de volcar el documento a stdout si el parámetro *filename* es '''. Esto no funciona si el PHP se utiliza como un módulo del apache. En tal caso se deberá escribir a un fichero y utilizar p. ej. **fpassthru()** para visualizarlo.

Vea también **fdf_close()** y el ejemplo para **fdf_create()**.

fdf_get_value (PHP 3>= 3.0.6, PHP 4)

Obtener el valor de un campo

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

La función **fdf_get_value()** devuelve el valor de un campo.

Vea también **fdf_set_value()**.

fdf_set_value (PHP 3>= 3.0.6, PHP 4)

Fijar el valor de un campo

```
void fdf_set_value (int fdf_document, string fieldname, string value, int isName)
```

La función **fdf_set_value()** fija el valor de un campo. El parámetro final determina si el valor del campo se deberá convertir a un Nombre PDF (*isName* = 1) o convertir en una Cadena PDF (*isName* = 0).

Vea también **fdf_get_value()**.

fdf_next_field_name (PHP 3>= 3.0.6, PHP 4)

Obtener el nombre del siguiente campo

```
string fdf_next_field_name (int fdf_document, string fieldname)
```

La función **fdf_next_field_name()** devuelve el nombre del campo tras el campo *fieldname* o el nombre del primer campo si el segundo parámetro es NULL.

Vea también **fdf_set_field()**, **fdf_get_field()**.

fdf_set_ap (PHP 3>= 3.0.6, PHP 4)

Ajusta la apariencia de un campo

```
void fdf_set_ap (int fdf_document, string fieldname, int face, string filename, int page_number)
```

La función **fdf_set_ap()** ajusta la apariencia de un campo (p. ej. el valor de la clave /AP). Los valores posibles de *face* son 1=FDFNormalAP, 2=FDFRolloverAP, 3=FDFDownAP.

fdf_set_status (PHP 3>= 3.0.6, PHP 4)

Fija el valor de la clave /STATUS

```
void fdf_set_status (int fdf_document, string status)
```

La función **fdf_set_status()** fija el valor de la clave /STATUS.

Vea también **fdf_get_status()**.

fdf_get_status (PHP 3>= 3.0.6, PHP 4)

Obtener el valor de la clave /STATUS

```
string fdf_get_status (int fdf_document)
```

La función **fdf_get_status()** devuelve el valor de la clave /STATUS.

Vea también **fdf_set_status()**.

fdf_set_file (PHP 3>= 3.0.6, PHP 4)

Fijar el valor de la clave /F

```
void fdf_set_file (int fdf_document, string filename)
```

La función **fdf_set_file()** fija el valor de la clave /F. La clave /F es simplemente una referencia a un formulario PDF que se va a llenar con datos. En un entorno web es un URL (p.ej. http:/testfdf/resultlabel.pdf).

Vea también **fdf_get_file()** y el ejemplo para **fdf_create()**.

fdf_get_file (PHP 3>= 3.0.6, PHP 4)

Obtener el valor de la clave /F

```
string fdf_get_file (int fdf_document)
```

La función **fdf_set_file()** devuelve el valor de la clave /F.

Vea también **fdf_set_file()**.

XXIV. Funciones FTP

FTP es el acrónimo de File Transfer Protocol (Protocolo de Transferencia de Ficheros).

Cuando se utiliza el módulo FTP, se definen las siguientes constantes: `FTP_ASCII`, y `FTP_BINARY`.

ftp_connect (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Establece una conexión FTP

```
int ftp_connect (string host [, int port])
```

Si tiene éxito, devuelve un flujo FTP. En caso de error, devuelve false.

ftp_connect() establece una conexión FTP al *host* especificado. El parámetro *port* especifica un puerto alternativo al que conectar. Si se omite o es cero, se usa el puerto FTP por defecto, 21.

ftp_login (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Comienza la sesión en una conexión FTP

```
int ftp_login (int ftp_stream, string username, string password)
```

Si tiene éxito, devuelve true. En caso de error, devuelve false.

Inicia una sesión (envía identificador de usuario y contraseña) en el flujo FTP especificado.

ftp_pwd (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Devuelve el nombre del directorio actual

```
int ftp_pwd (int ftp_stream)
```

Devuelve el directorio actual, o false en caso de error.

ftp_cdup (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Cambia al directorio padre

```
int ftp_cdup (int ftp_stream)
```

Si tiene éxito, devuelve true. En caso de error, devuelve false.

Cambia al directorio padre.

ftp_chdir (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Cambia de directorio en un servidor FTP

```
int ftp_chdir (int ftp_stream, string directory)
```

Si tiene éxito, devuelve true. En caso de error, devuelve false.

Cambia al directorio especificado por el parámetro *directory*.

ftp_mkdir (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Crea un directorio

```
string ftp_mkdir (int ftp_stream, string directory)
```

Si tiene éxito, devuelve el nombre del directorio recién creado. En caso de error, devuelve false.

Crea el directorio especificado por el parámetro *directory*.

ftp_rmdir (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Borra un directorio

```
int ftp_rmdir (int ftp_stream, string directory)
```

Si tiene éxito, devuelve true. En caso de error, devuelve false.

Borra el directorio especificado por el parámetro *directory*.

ftp_nlist (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Devuelve una lista de ficheros del directorio dado.

```
int ftp_nlist (int ftp_stream, string directory)
```

Si tiene éxito, devuelve un array de nombres de fichero. En caso de error, devuelve false.

ftp_rawlist (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Devuelve una lista detallada de ficheros del directorio dado.

```
int ftp_rawlist (int ftp_stream, string directory)
```

ftp_rawlist() ejecuta el comando FTP LIST, y devuelve el resultado como un array. Cada elemento del array corresponde a una línea de texto. La salida no se procesa de ninguna forma. Se puede utilizar el identificador de tipo de sistema devuelto por **ftp_systype()** para determinar cómo se deben interpretar los resultados.

ftp_systype (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Devuelve el identificador de tipo de sistema del servidor FTP remoto.

```
int ftp_systype (int ftp_stream)
```

Devuelve el tipo de sistema remoto, o false en caso de error.

ftp_pasv (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Activa o desactiva el modo pasivo.

```
int ftp_pasv (int ftp_stream, int pasv)
```

Si tiene éxito, devuelve true. En caso de error, devuelve false.

ftp_pasv() activa el modo pasivo si el parámetro *pasv* es true (desactiva el modo pasivo si *pasv* es false.) En modo pasivo, las conexiones de datos son iniciadas por el cliente, en lugar de ser iniciadas por el servidor.

ftp_get (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Descarga un fichero del servidor FTP.

```
int ftp_get (int ftp_stream, string local_file, string remote_file, int mode)
```

Si tiene éxito, devuelve true. En caso de error, devuelve false.

ftp_get() baja el fichero *remote_file* del servidor FTP, y lo guarda localmente como *local_file*. El modo de transferencia especificado por el parámetro *mode* debe ser FTP_ASCII o bien FTP_BINARY.

ftp_fget (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Descarga un fichero del servidor FTP y lo guarda en un fichero abierto.

```
int ftp_fget (int ftp_stream, int fp, string remote_file, int mode)
```

Si tiene éxito, devuelve true. En caso de error, devuelve false.

ftp_fget() baja el fichero *remote_file* del servidor FTP, y lo escribe en el puntero a fichero *fp*. El modo de transferencia especificado por el parámetro *mode* debe ser FTP_ASCII o bien FTP_BINARY.

ftp_put (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Sube un fichero al servidor FTP.

```
int ftp_put (int ftp_stream, string remote_file, string local_file, int mode)
```

Si tiene éxito, devuelve true. En caso de error, devuelve false.

ftp_put() sube el fichero local *local_file* al servidor FTP y lo guarda como *remote_file*. El modo de transferencia especificado por el parámetro *mode* debe ser FTP_ASCII o bien FTP_BINARY.

ftp_fput (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Sube un fichero abierto al servidor FTP.

```
int ftp_fput (int ftp_stream, string remote_file, int fp, int mode)
```

Si tiene éxito, devuelve true. En caso de error, devuelve false

ftp_fput() sube los datos apuntados por el puntero a fichero *fp* hasta alcanzar el final del fichero. Los resultados se guardan en el fichero *remote_file* del FTP remoto. El modo de transferencia especificado por el parámetro *mode* debe ser FTP_ASCII o bien FTP_BINARY.

ftp_size (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Devuelve el tamaño del fichero especificado.

```
int ftp_size (int ftp_stream, string remote_file)
```

Si tiene éxito devuelve el tamaño del fichero, o -1 en caso de error.

ftp_size() devuelve el tamaño de un fichero. Si ocurre algún error, o si el fichero no existe, devuelve -1. No todos los servidores soportan esta característica.

ftp_mdtm (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Devuelve la fecha de última modificación del fichero especificado.

```
int ftp_mdtm (int ftp_stream, string remote_file)
```

Si tiene éxito, devuelve una marca de tiempo UNIX (UNIX timestamp). En caso de error, devuelve -1.

ftp_mdtm() comprueba la fecha de última modificación de un fichero, y la devuelve como una marca de tiempo UNIX. Si se produce algún error, o el fichero no existe, devuelve -1. Tenga en cuenta que no todos los servidores soportan esta característica.

ftp_rename (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Renombra un fichero del servidor FTP.

```
int ftp_rename (int ftp_stream, string from, string to)
```

Si tiene éxito, devuelve true. En caso de error, devuelve false.

ftp_rename() renombra el fichero especificado por el parámetro *from* con el nuevo nombre *to*

ftp_delete (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Borra un fichero del servidor FTP.

```
int ftp_delete (int ftp_stream, string path)
```

Si tiene éxito, devuelve true. En caso de error, devuelve false.

ftp_delete() borra el fichero especificado por el parámetro *path* del servidor FTP.

ftp_quit (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Cierra una conexión FTP

```
int ftp_quit (int ftp_stream)
```

ftp_connect() cierra el flujo FTP *ftp_stream*.

XXV. Function Handling functions

These functions all handle various operations involved in working with functions.

call_user_func (PHP 3>= 3.0.3, PHP 4)

Call a user function given by the first parameter

```
mixed call_user_func (string function_name [, mixed parameter [, mixed ...]])
```

Call a user defined function given by the *function_name* parameter. Take the following:

```
function barber ($type) {
    print "You wanted a $type haircut, no problem";
}
call_user_func ('barber', "mushroom");
call_user_func ('barber', "shave");
```

create_function (PHP 4 >= 4.0.1)

Create an anonymous (lambda-style) function

```
string create_function (string args, string code)
```

Creates an anonymous function from the parameters passed, and returns a unique name for it. Usually the *args* will be passed as a single quote delimited string, and this is also recommended for the *code*. The reason for using single quoted strings, is to protect the variable names from parsing, otherwise, if you use double quotes there will be a need to escape the variable names, e.g. \\${avar}.

You can use this function, to (for example) create a function from information gathered at run time:

Ejemplo 1. Creating an anonymous function with `create_function()`

```
$newfunc = create_function('$a,$b','return "ln($a) + ln($b) = ".log($a * $b);');
echo "New anonymous function: $newfunc\n";
echo $newfunc(2,M_E)."\n";
// outputs
// New anonymous function: lambda_1
// ln(2) + ln(2.718281828459) = 1.6931471805599
```

Or, perhaps to have general handler function that can apply a set of operations to a list of parameters:

Ejemplo 2. Making a general processing function with `create_function()`

```
function process($var1, $var2, $farr) {
    for ($f=0; $f < count($farr); $f++)
        echo $farr[$f]($var1,$var2)."\n";
}

// create a bunch of math functions
$f1 = 'if ($a >=0) {return "b*a^2 = ".$b*sqrt($a);} else {return false;}';
$f2 = "return \"min(b^2+a, a^2,b) = \".min(\${a}\${a}+\${b},\${b}\${b}+\${a});";
$f3 = 'if ($a > 0 && $b != 0) {return "ln(a)/b = ".log($a)/$b;} else {return false;}';
$farr = array(
    create_function('$x,$y', 'return "some trig: .(sin($x) + $x*cos($y));"'),
    create_function('$x,$y', 'return "a hypotenuse: ".sqrt($x*$x + $y*$y);'),
    create_function('$a,$b', $f1),
    create_function('$a,$b', $f2),
```

```

create_function('$a,$b', '$f3')
);

echo "\nUsing the first array of anonymous functions\n";
echo "parameters: 2.3445, M_PI\n";
process(2.3445, M_PI, $farr);

// now make a bunch of string processing functions
$garr = array(
    create_function('$b,$a','if (strcmp($a,$b,3) == 0) return "*** \"$a\" \"'.
        'and \"$b\"\n** Look the same to me! (looking at the first 3 chars)\";'),
    create_function('$a,$b','; return "CRCs: ".crc32($a).", ".crc32(b);'),
    create_function('$a,$b','; return "similar(a,b) = ".similar_text($a,$b,&$p)."( $p% )\";');
);
echo "\nUsing the second array of anonymous functions\n";
process("Twas brillig and the slithy toves", "Twas the night", $garr);

```

and when you run the code above, the output will be:

```

Using the first array of anonymous functions
parameters: 2.3445, M_PI
some trig: -1.6291725057799
a hypotenuse: 3.9199852871011
b*a^2 = 4.8103313314525
min(b^2+a, a^2,b) = 8.6382729035898
ln(a/b) = 0.27122299212594

Using the second array of anonymous functions
** "Twas the night" and "Twas brillig and the slithy toves"
** Look the same to me! (looking at the first 3 chars)
CRCs: -725381282 , 1908338681
similar(a,b) = 11(45.83333333333333%)

```

But perhaps the most common use for of lambda-style (anonymous) functions is to create callback functions, for example when using **array_walk()** or **usort()**

Ejemplo 3. Using anonymous functions as callback functions

```

$av = array("the ","a ","that ","this ");
array_walk($av, create_function('&$v,$k','$v = $v."mango";'));
print_r($av); // for PHP3 use var_dump()
// outputs:
// Array
// (
//     [0] => the mango
//     [1] => a mango
//     [2] => that mango
//     [3] => this mango
// )

// an array of strings ordered from shorter to longer
$sv = array("small","larger","a big string","it is a string thing");
print_r($sv);
// outputs:
// Array
// (
//     [0] => small
//     [1] => larger
//     [2] => a big string
//     [3] => it is a string thing

```

```
// )

// sort it from longer to shorter
usort($sv, create_function('$a,$b','return strlen($b) - strlen($a);'));
print_r($sv);
// outputs:
// Array
// (
//     [0] => it is a string thing
//     [1] => a big string
//     [2] => larger
//     [3] => small
// )
```

func_get_arg (PHP 4 >= 4.0b4)

Devuelve un elemento de la lista de argumentos.

```
int func_get_arg (int arg_num)
```

Devuelve el argumento que está en la posición *arg_num* en la lista de argumentos de una función definida por el usuario. Los argumentos de la función se cuentan comenzando por la posición cero. **func_get_arg()** generará un aviso si se llama desde fuera de la definición de la función.

Si *arg_num* es mayor que el número de argumentos pasados realmente, se generará un aviso y **func_get_arg()** devolverá FALSE.

```
<?php
function foo() {
    $numargs = func_num_args();
    echo "Number of arguments: $numargs<br>\n";
    if ( $numargs >= 2 ) {
        echo "Second argument is: " . func_get_arg( 1 ) . "<br>\n";
    }
}
foo( 1, 2, 3 );
?>
```

func_get_arg() puede utilizarse conjuntamente con **func_num_args()** y **func_get_args()** para permitir a las funciones definidas por el usuario que acepten listas de argumentos de longitud variable.

Nota: Esta función fue añadida en PHP 4.

func_get_args (PHP 4 >= 4.0b4)

Devuelve un array que contiene la lista de argumentos de una función.

```
int func_get_args (void )
```

Devuelve un array en el que cada elemento es el miembro correspondiente de la lista de argumentos de la función definida por el usuario actual. **func_get_args()** generará un aviso si es llamada desde fuera de la definición de la función.

```
<?php
function foo() {
    $numargs = func_num_args();
    echo "Number of arguments: $numargs<br>\n";
    if ( $numargs >= 2 ) {
        echo "Second argument is: " . func_get_arg( 1 ) . "<br>\n";
    }
    $arg_list = func_get_args();
    for ( $i = 0; $i < $numargs; $i++ ) {
        echo "Argument $i is: " . $arg_list[$i] . "<br>\n";
    }
}
foo( 1, 2, 3 );
?>
```

func_get_args() puede utilizarse conjuntamente con **func_num_args()** y **func_get_arg()** para permitir a las funciones definidas por el usuario que acepten listas de argumentos de longitud variable.

Nota: Esta función fue añadida en PHP 4.

func_num_args (PHP 4 >= 4.0b4)

Devuelve el número de argumentos pasados a la función.

```
int func_num_args (void )
```

Devuelve el número de argumentos pasados a la función actual definida por el usuario. **func_num_args()** generará un aviso si es llamada desde fuera de la definición de la función.

```
<?php
function foo() {
    $numargs = func_num_args();
    echo "Number of arguments: $numargs\n";
}
foo( 1, 2, 3 ); // Prints 'Number of arguments: 3'
?>
```

func_num_args() puede utilizarse conjuntamente con **func_get_arg()** y **func_get_args()** para permitir a las funciones definidas por el usuario que acepten listas de argumentos de longitud variable.

Nota: Esta función fue añadida en PHP 4.

function_exists (PHP 3>= 3.0.7, PHP 4)

Devuelve true si la función dada ha sido definida

```
int function_exists (string function_name)
```

Consulta la lista de funciones definidas buscando *function_name* (nombre de función). Devuelve true si encuentra el nombre de función dado, false en otro caso.

register_shutdown_function (PHP 3>= 3.0.4, PHP 4)

Registra una función para su ejecución en el cierre.

```
int register_shutdown_function (string func)
```

Registra la función nombrada en *func* para que se ejecute cuando el script procese su finalización.

Aviso:

Debido a que no se permite ningún tipo de salida en el navegador en esta función, no será capaz de depurarla utilizando sentencias como print o echo.

XXVI. GNU Gettext

bindtextdomain (PHP 3>= 3.0.7, PHP 4)

Establece la ruta para un dominio

```
string bindtextdomain (string domain, string directory)
```

La función **bindtextdomain()** establece la ruta para el dominio.

dcgettext (PHP 3>= 3.0.7, PHP 4)

Omite el dominio para una única búsqueda

```
string dcgettext (string domain, string message, int category)
```

Esta función permite omitir el dominio actual para una búsqueda de un mensaje. Además permite especificar una categoría.

dgettext (PHP 3>= 3.0.7, PHP 4)

Omite el dominio actual

```
string dgettext (string domain, string message)
```

La función **dgettext()** permite omitir el dominio actual para una única búsqueda.

gettext (PHP 3>= 3.0.7, PHP 4)

Realiza una búsqueda del mensaje en el dominio actual

```
string gettext (string message)
```

Esta función devuelve la traducción de la cadena si encuentra una en la tabla de traducciones, o el mensaje enviado si no se encuentra ninguna. Puede usar un carácter de subrayado como un alias para esta función.

Ejemplo 1. Gettext()-check

```
<?php
// Establece el idioma en alemán
putenv ("LANG=de");

// Especifica la localización de las tablas de traducción
bindtextdomain ("myPHPApp", "./locale");

// Elige un dominio
textdomain ("myPHPApp");

// Imprime un mensaje de prueba
print (gettext ("Welcome to My PHP Application"));
?>
```

textdomain (PHP 3>= 3.0.7, PHP 4)

Establece el dominio actual

```
int textdomain ([string library])
```

Esta función establece el dominio en el que se realizarán las busquedas provocadas por las llamadas a **gettext()**, normalmente el nombre dado a la aplicación. Se devuelve el dominio anterior. Puede llamar a la función sin parámetros para obtener el dominio actual sin necesidad de cambiarlo.

XXVII. GMP functions

These functions allow you to work with arbitrary-length integers using GNU MP library. In order to have these functions available, you must compile PHP with GMP support by using the `-with-gmp` option.

You can download the GMP library from <http://www.swox.com/gmp/>. This site also has the GMP manual available.

You will need GMP version 2 or better to use these functions. Some functions may require more recent version of the GMP library.

These functions have been added in PHP 4.0.4.

Nota: Most GMP functions accept GMP number arguments, defined as `resource` below. However, most of these functions will accept also numeric and string arguments, given it's possible to convert the latter to number. Also, if there's faster function that can operate on integer arguments, it would be used instead of slower function when supplied arguments are integers. This is done transparently, so the bottom line is that you can use integers in every function that expects GMP number. See also `gmp_init()` function.

Ejemplo 1. Factorial function using GMP

```
<?php
function fact ($x) {
    if ($x <= 1)
        return 1;
    else
        return gmp_mul ($x, fact ($x-1));
}
print gmp_strval (fact (1000)) . "\n";
?>
```

This will calculate factorial of 1000 (pretty big number) very fast.

gmp_init (PHP 4 >= 4.0.4)

Create GMP number

```
resource gmp_init (mixed number)
```

Creates a GMP number from integer or string. String representation can be decimal or hexadecimal. In the latter case, the string should start with 0x.

Ejemplo 1. Creating GMP number

```
<?php
    $a = gmp_init (123456);
    $b = gmp_init ("0xFFFFDEBACDFEDF7200");
?>
```

Nota: It is not necessary to call this function if you want to use integer or string in place of GMP number in GMP functions, like **gmp_add()**. Function arguments are automatically converted to GMP numbers, if such conversion is possible and needed, using the same rules as **gmp_init()**.

gmp_intval (PHP 4 >= 4.0.4)

Convert GMP number to integer

```
int gmp_intval (resource gmpnumber)
```

This function allows to convert GMP number to integer.

Aviso

This function returns useful result only if the number actually fits the PHP integer (i.e., signed long type). If you want just to print the GMP number, use **gmp_strval()**.

gmp_strval (PHP 4 >= 4.0.4)

Convert GMP number to string

```
string gmp_strval (resource gmpnumber [, int base])
```

Convert GMP number to string representation in base *base*. The default base is 10. Allowed values for the base are from 2 to 36.

Ejemplo 1. Converting GMP number to string

```
<?php
    $a = gmp_init("0x41682179fbf5");
```

```
    printf ("Decimal: %s, 36-based: %s", gmp_strval($a), gmp_strval($a,36));
?>
```

gmp_add (PHP 4 >= 4.0.4)

Add numbers

```
resource gmp_add (resource a, resource b)
```

Add two GMP numbers. The result will be GMP number representing the sum of the arguments.

gmp_sub (PHP 4 >= 4.0.4)

Subtract numbers

```
resource gmp_sub (resource a, resource b)
```

Subtract *b* from *a* and returns the result.

gmp_mul (PHP 4 >= 4.0.4)

Multiply numbers

```
resource gmp_mul (resource a, resource b)
```

Multiplies *a* by *b* and returns the result.

gmp_div_q (PHP 4 >= 4.0.4)

Divide numbers

```
resource gmp_div_q (resource a, resource b [, int round])
```

Divides *a* by *b* and returns the integer result. The result rounding is defined by the *round*, which can have the following values:

- *GMP_ROUND_ZERO*: The result is truncated towards 0.
- *GMP_ROUND_PLUSINF*: The result is rounded towards +infinity.
- *GMP_ROUND_MINUSINF*: The result is rounded towards -infinity.

This function can also be called as **gmp_div()**.

See also **gmp_div_r()**, **gmp_div_qr()**

gmp_div_r (PHP 4 >= 4.0.4)

Remainder of the division of numbers

```
resource gmp_div_r (resource n, resource d [, int round])
```

Calculates remainder of the integer division of *n* by *d*. The remainder has the sign of the *n* argument, if not zero.

See the **gmp_div_q()** function for description of the *round* argument.

See also **gmp_div_q()**, **gmp_div_qr()**

gmp_div_qr (PHP 4 >= 4.0.4)

Divide numbers and get quotient and remainder

```
array gmp_div_qr (resource n, resource d [, int round])
```

The function divides *n* by *d* and returns array, with the first element being $[n/d]$ (the integer result of the division) and the second being $(n - [n/d] * d)$ (the remainder of the division).

See the **gmp_div_q()** function for description of the *round* argument.

Ejemplo 1. Division of GMP numbers

```
<?php
    $a = gmp_init ("0x41682179fbf5");
    $res = gmp_div_qr ($a, "0xDEFE75");
    printf("Result is: q - %s, r - %s",
        gmp_strval ($res[0]), gmp_strval ($res[1]));
?>
```

See also **gmp_div_q()**, **gmp_div_r()**.

gmp_div (PHP 4 >= 4.0.4)

Divide numbers

```
resource gmp_divexact (resource a, resource b)
```

This function is an alias to **gmp_div_q()**.

gmp_mod (PHP 4 >= 4.0.4)

Modulo operation

```
resource gmp_mod (resource n, resource d)
```

Calculates n modulo d . The result is always non-negative, the sign of d is ignored.

gmp_divexact (PHP 4 >= 4.0.4)

Exact division of numbers

```
resource gmp_divexact (resource n, resource d)
```

Divides n by d , using fast "exact division" algorithm. This function produces correct results only when it is known in advance that d divides n .

gmp_cmp (PHP 4 >= 4.0.4)

Compare numbers

```
int gmp_cmp (resource a, resource b)
```

Returns positive value if $a > b$, zero if $a = b$ and negative value if $a < b$.

gmp_neg (PHP 4 >= 4.0.4)

Negate number

```
resource gmp_neg (resource a)
```

Returns $-a$.

gmp_abs (PHP 4 >= 4.0.4)

Absolute value

```
resource gmp_abs (resource a)
```

Returns absolute value of a .

gmp_sign (PHP 4 >= 4.0.4)

Sign of number

```
int gmp_sign (resource a)
```

Return sign of a - 1 if a is positive and -1 if it's negative.

gmp_fact (PHP 4 >= 4.0.4)

Factorial

```
resource gmp_fact (int a)
```

Calculates factorial ($a!$) of a .

gmp_sqrt (PHP 4 >= 4.0.4)

Square root

```
resource gmp_sqrt (resource a)
```

Calculates square root of a .

gmp_sqrtrm (unknown)

Square root with remainder

```
array gmp_sqrtrm (resource a)
```

Returns array where first element is the integer square root of a (see also **gmp_sqrt()**), and the second is the remainder (i.e., the difference between a and the first element squared).

gmp_perfect_square (PHP 4 >= 4.0.4)

Perfect square check

```
bool gmp_perfect_square (resource a)
```

Returns true if a is a perfect square, false otherwise.

See also: **gmp_sqrt()**, **gmp_sqrtrm()**.

gmp_pow (PHP 4 >= 4.0.4)

Raise number into power

```
resource gmp_pow (resource base, int exp)
```

Raise $base$ into power exp . The case of 0^0 yields 1. exp cannot be negative.

gmp_powm (PHP 4 >= 4.0.4)

Raise number into power with modulo

```
resource gmp_powm (resource base, resource exp, resource mod)
```

Calculate (*base* raised into power *exp*) modulo *mod*. If *exp* is negative, result is undefined.

gmp_prob_prime (PHP 4 >= 4.0.4)

Check if number is "probably prime"

```
int gmp_prob_prime (resource a [, int reps])
```

If this function returns 0, *a* is definitely not prime. If it returns 1, then *a* is "probably"prime. If it returns 2, then *a* is surely prime. Reasonable values of *reps* vary from 5 to 10 (default being 10); a higher value lowers the probability for a non-prime to pass as a "probable"prime.

The function uses Miller-Rabin's probabilistic test.

gmp_gcd (PHP 4 >= 4.0.4)

Calculate GCD

```
resource gmp_gcd (resource a, resource b)
```

Calculate greatest common divisor of *a* and *b*. The result is always positive even if either of or both input operands are negative.

gmp_gcdext (PHP 4 >= 4.0.4)

Calculate GCD and multipliers

```
array gmp_gcdext (resource a, resource b)
```

Calculates *g*, *s*, and *t*, such that $a*s + b*t = g = \text{gcd}(a,b)$, where gcd is gretest common divisor. Returns array with respective elements *g*, *s* and *t*.

gmp_invert (PHP 4 >= 4.0.4)

Inverse by modulo

```
resource gmp_invert (resource a, resource b)
```

Computes the inverse of *a* modulo *b*. Returns false if an inverse does not exist.

gmp_legendre (PHP 4 >= 4.0.4)

Legendre symbol

```
int gmp_legendre (resource a, resource p)
```

Compute the Legendre symbol (<http://www.utm.edu/research/primes/glossary/LegendreSymbol.html>) of *a* and *p*. *p* should be odd and must be positive.

gmp_jacobi (PHP 4 >= 4.0.4)

Jacobi symbol

```
int gmp_jacobi (resource a, resource p)
```

Computes Jacobi symbol (<http://www.utm.edu/research/primes/glossary/JacobiSymbol.html>) of *a* and *p*. *p* should be odd and must be positive.

gmp_random (PHP 4 >= 4.0.4)

Random number

```
resource gmp_random (int limiter)
```

Generate a random number. The number will be up to *limiter* words long. If *limiter* is negative, negative numbers are generated.

gmp_and (PHP 4 >= 4.0.4)

Logical AND

```
resource gmp_and (resource a, resource b)
```

Calculates logical AND of two GMP numbers.

gmp_or (PHP 4 >= 4.0.4)

Logical OR

```
resource gmp_or (resource a, resource b)
```

Calculates logical inclusive OR of two GMP numbers.

gmp_xor (PHP 4 >= 4.0.4)

Logical XOR

```
resource gmp_xor (resource a, resource b)
```

Calculates logical exclusive OR (XOR) of two GMP numbers.

gmp_setbit (PHP 4 >= 4.0.4)

Set bit

```
resource gmp_setbit (resource &a, int index [, bool set_clear])
```

Sets bit *index* in *a*. *set_clear* defines if the bit is set to 0 or 1. By default the bit is set to 1.

gmp_clrbit (PHP 4 >= 4.0.4)

Clear bit

```
resource gmp_clrbit (resource &a, int index)
```

Clears (sets to 0) bit *index* in *a*.

gmp_scan0 (PHP 4 >= 4.0.4)

Scan for 0

```
int gmp_scan0 (resource a, int start)
```

Scans *a*, starting with bit *start*, towards more significant bits, until the first clear bit is found. Returns the index of the found bit.

gmp_scan1 (PHP 4 >= 4.0.4)

Scan for 1

```
int gmp_scan1 (resource a, int start)
```

Scans *a*, starting with bit *start*, towards more significant bits, until the first set bit is found. Returns the index of the found bit.

gmp_popcount (PHP 4 >= 4.0.4)

Population count

```
int gmp_popcount (resource a)
```

Return the population count of *a*.

gmp_hamdist (PHP 4 >= 4.0.4)

Hamming distance

```
int gmp_hamdist (resource a, resource b)
```

Returns the hamming distance between *a* and *a*. Both operands should be non-negative.

XXVIII. Funciones HTTP

Estas funciones permiten manipular la salida que se envía al navegador remoto a nivel de protocolo HTTP.

header (PHP 3, PHP 4)

Manda una cabecera HTTP

```
int header (string string)
```

La función **Header()** se utiliza al comienzo de un fichero HTML para enviar las cadenas de texto de la cabecera HTTP. Consulte la Especificación HTTP 1.1 (<http://www.w3.org/Protocols/rfc2616/rfc2616>) para obtener más información sobre las cabeceras http. *Nota:* Recuerde que la función **Header()** debe llamarse antes de que se genere salida alguna, bien con etiquetas HTML normales o con PHP. Un error muy frecuente consiste en leer código con **include()** o con **auto_prepend**, y que dicho código inserte espacios o líneas en blanco antes de llamar a **header()**.

Hay dos casos especiales de llamadas a header. La primera es la cabecera "Location". No sólo envía esta cabecera al navegador, sino que también devuelve un código de estado REDIRECT a Apache. Desde el punto de vista del programador de scripts esto no debería ser importante, pero para la gente que comprende las interioridades de Apache sí puede serlo.

```
header("Location: http://www.php.net"); /* Redirect browser to PHP web site */
exit; /* Make sure that code below does not get executed when we redirect. */
```

El segundo caso especial se produce con cualquier cabecera que comience con la cadena, "HTTP/"(las mayúsculas no son significativas). Por ejemplo, si tiene la directiva ErrorDocument 404 de Apache apuntando a un script PHP, es una buena idea asegurarse de que su script de PHP genera realmente un 404. La primera cosa que debe hacer en su script sería:

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

Los scripts de PHP a menudo generan HTML dinámico que no debe almacenarse en la caché del navegador cliente o en la caché de cualquier proxy situado entre el servidor y el navegador cliente. Se puede obligar a muchos proxies y clientes a que deshabiliten el almacenamiento en caché con

```
header("Expires: Mon, 26 Jul 1997 05:00:00 GMT"); // Date in the past
header("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT"); // always modified
header("Cache-Control: no-cache, must-revalidate"); // HTTP/1.1
header("Pragma: no-cache"); // HTTP/1.0
```

setcookie (PHP 3, PHP 4)

Envía una cookie

```
int setcookie (string name, string value, int expire, string path, string domain, int secure)
```

setcookie() define una cookie para ser enviada con el resto de la información de la cabecera. Las cookies deben enviarse *antes* de mandar cualquier otra cabecera (esta es una restricción de las cookies, no de PHP). Esto requiere que sitúe las llamadas a esta función *antes* de cualquier etiqueta <html> o <head>.

Todos los parámetros excepto *name* son opcionales. Si sólo se especifica el parámetro *name*, la cookie con ese nombre se borrará del cliente remoto. También puede sustituir cualquier parámetro por una cadena de texto vacía ("") y saltar así ese parámetro. Los parámetros *expire* y *secure* son números enteros y no se pueden saltar con una cadena de texto vacía. En su lugar utilice un cero (0). El parámetro *expire* es un entero de tiempo típico de UNIX tal como lo devuelven las

funciones **time()** o **mktime()**. El parámetro **secure** indica que la cookie se debe transmitir única y exclusivamente sobre una conexión segura HTTPS.

Fallos habituales:

Las cookies no se hacen visibles hasta la siguiente carga de una página para la que la cookie deba estar visible.

Las llamadas múltiples a **setcookie()** en el mismo script se ejecutarán en orden inverso. Si está intentando borrar una cookie antes de insertar otra, debe situar la llamada de inserción antes de la de borrado.

A continuación se muestran algunos ejemplos::

Ejemplo 1. setcookie(), ejemplos

```
setcookie( "TestCookie", "Test Value");
setcookie( "TestCookie", $value, time() + 3600); /* expire in 1 hour */
setcookie( "TestCookie", $value, time() + 3600, "/~rasmus/", ".utoronto.ca", 1);
```

Tenga en cuenta que el campo value de la cookie se codifica como URL (urlencode) automáticamente cuando envía la cookie. Cuando ésta se recibe, se descodifica automáticamente y se asigna a una variable con el mismo nombre que el nombre de la cookie. Para ver el contenido de nuestra cookie de prueba en un script, simplemente utilice uno de los siguientes ejemplos:

```
echo $TestCookie;
echo $HTTP_COOKIE_VARS[ "TestCookie" ];
```

También puede utilizar arrays de cookies empleando la notación de array en el nombre de la cookie. Esto tiene como efecto establecer tantas cookies como elementos de array, pero cuando el script recibe la cookie, se guardan los valores en un array con el nombre de la cookie:

```
setcookie( "cookie[three]", "cookiethree" );
setcookie( "cookie[two]", "cookietwo" );
setcookie( "cookie[one]", "cookieone" );
if ( isset( $cookie ) ) {
    while( list( $name, $value ) = each( $cookie ) ) {
        echo "$name == $value<br>\n";
    }
}
```

Para obtener más información sobre las cookies, consulte la especificación de cookies de Netscape, que se encuentra en http://www.netscape.com/newsref/std/cookie_spec.html.

Microsoft Internet Explorer 4 con Service Pack 1 no funciona correctamente con las cookies que tienen asociado el parámetro path.

Netscape Communicator 4.05 y Microsoft Internet Explorer 3.x funcionan aparentemente de manera incorrecta cuando no se especifican los parámetros path y time.

XXIX. Funciones para Hyperwave

Introducción

Hyperwave ha sido desarrollado en el IICM (<http://www.iicm.edu>) en Graz. Comenzó con el nombre Hyper-G y cambió a Hyperwave cuando fue comercializado (Si lo recuerdo bien, fue en 1996).

Hyperwave no es software gratuito. La versión actual, 4.1, está disponible en www.hyperwave.com (<http://www.hyperwave.com/>). Se puede solicitar gratuitamente una versión limitada (30 días).

Hyperwave es un sistema de información similar a una base de datos (HIS, Hyperwave Information Server - Servidor Hyperwave de Información). Su objetivo es el almacenamiento y manipulación de documentos. Un documento puede ser cualquier bloque posible de datos que también puede ser almacenado en un archivo. Cada documento se acompaña por su registro de objeto. El registro de objeto contiene metadatos para el documento. Los metadatos son una lista de atributos que pueden ser extendidos por el usuario. Ciertos atributos siempre son fijados por el servidor Hyperwave, otros pueden ser modificados por el usuario. Un atributo es un par nombre/valor de la forma nombre=valor. El registro completo del objeto tiene tantos de estos pares como guste el usuario. El nombre de un atributo no tiene porqué ser único, p. ej. un título puede aparecer varias veces en el registro de un objeto. Esto tiene sentido si se desea especificar un título en diferentes idiomas. En dicho caso existe la convención de que cada valor de título esté precedido por la abreviatura de dos letras del idioma, seguida por dos puntos, como p. ej. 'en:Title in English' o 'es:Título en Español'. Otros atributos tales como descripciones o palabras clave son candidatos potenciales a esta diferenciación. También se pueden reemplazar las abreviaturas de idioma por cualquier otra cadena siempre y cuando estén separadas por los dos puntos del resto del valor del atributo.

Cada registro de objeto tiene una representación nativa como cadena con cada par nombre/valor separado por una línea nueva. La extensión Hyperwave también conoce una segunda representación que consiste en un array asociativo donde el nombre del atributo es la clave. Los valores de atributo multilingües en sí mismos forman otro array asociativo donde la clave es la abreviatura del idioma. Realmente cualquier atributo múltiple forma una tabla asociativa donde la cadena a la izquierda de los dos puntos en el valor de atributo es la clave. (Esto no se ha implementado por completo. Sólo los atributos Title, Description y Keyword son tratados adecuadamente.)

Aparte de los documentos, todos los hiper-enlaces contenidos en un documento son almacenados también como registros de objeto. Cuando el documento sea insertado en la base de datos, los hiper-enlaces que haya en un documento serán borrados del mismo y almacenados como objetos individuales. El registro de objeto del enlace contiene información acerca de dónde comienza y dónde termina. Para recuperar el documento original se deberá recuperar el documento sin los enlaces y la lista de los mismos para reinsertarla (Las funciones **hw_pipedocument()** y **hw_gettext()** hacen esto para usted). La ventaja de separar los enlaces del documento es obvia. Una vez un documento al que apunta un enlace cambia de nombre, el enlace puede modificarse fácilmente. El documento que contiene el enlace no se ve afectado. Incluso se puede añadir un enlace a un documento sin alterarlo.

Decir que **hw_pipedocument()** y **hw_gettext()** hacen automáticamente la inserción de enlaces no es tan simple como suena. Insertar los enlaces implica una cierta jerarquía en los documentos. En un servidor web esto viene dado por el sistema de archivos, pero el Hyperwave tiene su propia jerarquía y los nombres no representan la posición de un objeto en dicha jerarquía. Por tanto, la creación de los enlaces precisa primeramente de realizar un mapeado entre el espacio de nombres y la jerarquía del Hyperwave y el espacio de nombres respectivo de una jerarquía de web. La diferencia fundamental entre Hyperwave y la web es la distinción clara entre nombres y jerarquía que se da en el primero. El nombre no contiene ninguna información sobre la posición del objeto en la jerarquía. En la web, el nombre también contiene la información sobre la posición en la jerarquía del objeto. Esto nos lleva a dos posibles formas de mapeo. O bien se reflejan la jerarquía del Hyperwave y el nombre del objeto Hyperwave en el URL o sólo el nombre. Para facilitar las cosas, se utiliza el segundo método. El objeto Hyperwave de nombre 'mi_objeto' es mapeado a 'http://host/mi_objeto' sin importar dónde reside dentro de la jerarquía de Hyperwave. Un objeto con el nombre 'padre/mi_objeto' podría ser el hijo de 'mi_objeto' en la jerarquía Hyperwave, aunque en el espacio de nombres web aparezca justamente lo opuesto y el usuario pueda ser llevado a confusión. Esto sólo se puede evitar seleccionando nombres de objetos razonables.

Hecha esta decisión surge un segundo problema. ¿Cómo implicar al PHP? el URL http://host/mi_objeto no llamará a ningún script PHP a no ser que se le diga al servidor que lo transforme en p. ej. 'http://host/script_php3/mi_objeto' y que el 'script_php3' luego evalúe la variable \$PATH_INFO y recupere el objeto con nombre 'mi_objeto' del servidor Hyperwave. Hay sólo un pequeño inconveniente que se puede resolver fácilmente. Cuando se reescribe cualquier URL no se permite el acceso a ningún otro documento en el servidor web. Un script de PHP para buscar en el servidor Hyperwave sería

imposible. Por lo tanto se necesitará al menos una segunda regla de reescritura para que excluya ciertos URL, como los que empiecen p. ej. por `http://host/Hyperwave`. Básicamente esto sería compartir un espacio de nombres entre el servidor web y el servidor Hyperwave.

Los enlaces se insertan en los documentos basándose en el mecanismo citado más arriba.

Se vuelve más complicado si el PHP no se está ejecutando como módulo del servidor o como script CGI, sino que se ejecuta como aplicación, p. ej. para volcar el contenido del servidor de Hyperwave a un CD-ROM. En dicho caso tiene sentido mantener la jerarquía Hyperwave y mapearla en el sistema de archivos. Esto entra conflicto con los nombres de los objetos si estos reflejan su propia jerarquía (p. ej. eligiendo nombres que comienzan por '/'). Por tanto, la '/' tendrá que ser reemplazada por otro carácter, p. ej. '_' para continuar.

El protocolo de red para comunicarse con el servidor Hyperwave se denomina HG-CSP (<http://www.hyperwave.de/7.17-hg-prot>) (Hyper-G Client/Server Protocol, Protocolo Hyper-G Cliente/Servidor). Está basado en mensajes que inician ciertas acciones, p. ej. obtener el registro de un objeto. En versiones anteriores del Servidor Hyperwave se distribuyeron dos clientes nativos (Harmony, Amadeus) para la comunicación con el servidor. Ambos desaparecieron cuando se comercializó el Hyperwave. Para sustituirlo se proporcionó el llamado wavemaster. El wavemaster es como un conversor de protocolo de HTTP a HG-CSP. La idea es realizar toda la administración de la base de datos y la visualización de documentos con una interfaz web. El wavemaster implementa una serie de posicionadores para acciones que permiten personalizar la interfaz. Dicho conjunto de posicionadores se denomina el Lenguaje PLACE. El PLACE no posee muchas de las características de un lenguaje de programación real y las extensiones al mismo únicamente alargan la lista de posicionadores. Esto ha obligado al uso de JavaScript que, en mi opinión, no hace la vida más fácil.

Añadir soporte de Hyperwave al PHP llenaría el espacio que deja la falta de un lenguaje de programación que permita personalizar la interfaz. El PHP implementa todos los mensajes definidos en el HG-CSP pero además proporciona comandos más poderosos, p. ej. recuperar documentos completos.

El Hyperwave tiene su propia terminología para dar nombre a ciertos tipos de información. Esta ha sido ampliamente extendida y anulada. Casi todas las funciones operan en uno de los siguientes tipos de datos.

- ID de objeto: Un valor entero único para cada objeto del servidor Hyperwave. También es uno de los atributos del registro de objeto (ObjectID). Los ID de objeto se usan generalmente como parámetros de entrada que especifican a un objeto.
- registro de objeto: Una cadena con pares atributo=valor con la forma atributo=valos. Los pares están separados unos de otros por un retorno de carro. Un registro de objeto puede convertirse fácilmente en una tabla (array) de objetos usando `hw_object2array()`. Varias funciones devuelven registros de objeto. Los nombres de dichas funciones terminan en obj.
- tabla de objetos: Una tabla asociativa con todos los atributos de un objeto. La clave es el nombre del atributo. Si un atributo aparece más de una vez en un registro de objeto será convertido en otra tabla asociativa o indizada. Los atributos que dependen del idioma (como el título, claves o descripción) se convertirán en una tabla asociativa con la abreviatura del idioma como clave. El resto de los atributos múltiples crearán una tabla indizada. Las funciones de PHP nunca devuelven tablas de objetos.
- hw_document: Este es un nuevo tipo de datos que almacena el documento actual, p. ej. HTML, PDF, etc. Está algo optimizado para documentos HTML pero puede usarse para cualquier formato.

Varias funciones que devuelven una tabla de registros de objeto también devuelven una tabla asociativa con información estadística sobre los mismos. La tabla es el último elemento del registro de objeto. La tabla estadística contiene los siguientes elementos:

Hidden

Número de registros de objeto con el atributo PresentationHints puesto a Hidden.

CollectionHead

Número de registros de objeto con el atributo PresentationHints puesto a CollectionHead.

FullCollectionHead

Número de registros de objeto con el atributo PresentationHints puesto a FullCollectionHead.

CollectionHeadNr

Índice a una tabla de registros de objeto con el atributo PresentationHints puesto a CollectionHead.

FullCollectionHeadNr

Índice a una tabla de registros de objeto con el atributo PresentationHints puesto a FullCollectionHead.

Total

Total: Número de registros de objeto.

Integración con Apache

La extensión Hyperwave se utiliza mejor cuando el PHP se compila como un módulo de Apache. En tal caso el servidor Hyperwave subyacente puede ser ocultado casi por completo de los usuarios si el Apache utiliza su motor de re-escritura. Las siguientes instrucciones explicarán esto.

Como el PHP con soporte Hyperwave incluido en el Apache se ha diseñado para sustituir la solución nativa de Hyperwave basada en Wavemaster, asumiré que el servidor Apache sólo sirve como interfaz web para el Hyperwave. Esto no es necesario, pero simplifica la configuración. El concepto es bastante sencillo. Primeramente necesita un script PHP que evalúe la variable PATH_INFO y que trate su valor como el nombre de un objeto Hyperwave. Llámalo a este script 'Hyperwave'. El URL `http://nombre.servidor/Hyperwave/nombre_de_objeto` devolvería entonces el objeto Hyperwave llamado 'nombre_de_objeto'. Dependiendo del tipo del objeto, así reaccionaría el script. Si es una colección, probablemente devolverá una lista de hijos. Si es un documento devolverá el tipo MIME y el contenido. Se puede mejorar ligeramente si se usa el motor de re-escritura del Apache. Desde el punto de vista del usuario será más sencillo si el URL `http://nombre.servidor/nombre de objeto` devuelve el objeto. La regla de reescritura es muy sencilla:

```
RewriteRule ^/(.*) /usr/local/apache/htdocs/HyperWave/$1 [L]
```

Ahora todo URL apunta a un objeto en el servidor Hyperwave. Esto provoca un problema sencillo de resolver. No hay forma de ejecutar otro script, p. ej. para buscar, salvo el script 'Hyperwave'. Esto se puede solucionar con otra regla de reescritura como la siguiente:

```
RewriteRule ^/hw/(.*) /usr/local/apache/htdocs/hw/$1 [L]
```

Esta reservará el directorio `/usr/local/apache/htdocs/hw` para scripts adicionales y otros archivos. Sólo hay que asegurarse que esta regla se evalúa antes de la otra. Sólo hay un pequeño inconveniente: todos los objetos Hyperwave cuyo nombre comienza por 'hw/' serán ocultados, así que asegúrese de no utilizar dichos nombres. Si necesita más directorios, p. ej. para imágenes, simplemente añada más reglas o sitúe los archivos en un solo directorio. Por último, no olvide conectar el motor de re-escritura con

```
RewriteEngine on
```

Mi experiencia me ha demostrado que necesitará los siguientes scripts:

- para devolver el script en sí
- para permitir las búsquedas
- para identificarse
- para ajustar su perfil

- uno para cada función adicional como mostrar los atributos del objeto, mostrar información sobre usuarios, mostrar el estado del servidor, etc.

Pendientes

Aún hay varias cosas pendientes:

- hw_InsertDocument debe dividirse en **hw_InsertObject()** y **hw_PutDocument()**.
- Los nombres de algunas funciones aún no están fijados.
- Muchas funciones precisan la conexión actual como primer parámetro. Esto obliga a escribir mucho, lo cual no es con frecuencia necesario si sólo hay una conexión abierta. Una conexión por defecto mejoraría esto.
- La conversión de registro de objeto a tabla de objeto necesita manipular cualquier atributo múltiple.

hw_Array2Objrec (PHP 3>= 3.0.4, PHP 4)

convierte atributos de tabla de objeto a registro de objeto

```
string hw_array2objrec (array tabla_objetos)
```

Convierte una *tabla_objetos* en un registro de objeto. Los atributos múltiples como 'Título' en distintos idiomas son tratados correctamente.

Vea también **hw_objrec2array()**.

hw_Children (PHP 3>= 3.0.3, PHP 4)

id de objeto de los hijos

```
array hw_children (int conexión, int IDobjeto)
```

Devuelve una tabla de id de objeto. Cada uno de ellos pertenece a un hijo de la colección cuyo ID es *IDobjeto*. La tabla contiene tanto los documentos como las colecciones hijas.

hw_ChildrenObj (PHP 3>= 3.0.3, PHP 4)

registros de objeto de los hijos

```
array hw_childrenobj (int conexión, int IDobjeto)
```

Devuelve una tabla de registros de objeto. Cada uno de ellos pertenece a un hijo de la colección cuyo ID es *IDobjeto*. La tabla contiene tanto los documentos como las colecciones hijas.

hw_Close (PHP 3>= 3.0.3, PHP 4)

cierra la conexión Hyperwave

```
int hw_close (int conexión)
```

Devuelve false si la conexión no es un índice válido de conexión, y true en caso contrario. Cierra la conexión dada con el servidor de Hyperwave.

hw_Connect (PHP 3>= 3.0.3, PHP 4)

abre una conexión

```
int hw_connect (string servidor, int puerto, string usuario, string clave)
```

Abre una conexión con un servidor Hyperwave y devuelve un índice de conexión si hay éxito, o false si la conexión no se pudo realizar. Cada argumento debe ser una cadena entrecomillada salvo el número de puerto. Los argumentos de

usuario y *clave* son opcionales y pueden omitirse. En tal caso no se realizará identificación alguna con el servidor. Es similar a identificarse como el usuario 'anonymous'. Esta función devuelve un índice de conexión que se necesita para otras funciones Hyperwave. Puede tener varias conexiones abiertas a la vez. Recuerde que la clave no está encriptada.

Vea también **hw_pConnect()**.

hw_Cp (PHP 3>= 3.0.3, PHP 4)

copia objetos

```
int hw_cp (int conexión, array tabla_id_objeto, int id destino)
```

Copia los objetos cuyos id se especifican en el segundo parámetro a la colección identificada como *id destino*.

El valor devuelto es el número de objetos copiados.

Vea también **hw_mv()**.

hw_Deleteobject (PHP 3>= 3.0.3, PHP 4)

borra objetos

```
int hw_deleteobject (int conexión, int objeto_a_borrar)
```

Borra el objeto con el id dado como segundo parámetro. Borrará todas las instancias del mismo.

Devuelve TRUE si no hay error o FALSE en caso contrario.

Vea también **hw_mv()**.

hw_DocByAnchor (PHP 3>= 3.0.3, PHP 4)

id del objeto al que pertenece un enlace

```
int hw_docbyanchor (int conexión, int IDenlace)
```

Devuelve el id de objeto del documento al que pertenece el *IDenlace*.

hw_DocByAnchorObj (PHP 3>= 3.0.3, PHP 4)

registro de objeto al que pertenece un enlace

```
string hw_docbyanchorobj (int conexión, int IDenlace)
```

Devuelve el registro de objeto del documento al que pertenece el *IDenlace*.

hw_DocumentAttributes (PHP 3>= 3.0.3, PHP 4 <= 4.0b1)

registro de objeto de un documento_hw

```
string hw_documentattributes (int documento_hw)
```

Devuelve el registro de objeto del documento dado.

Vea también **hw_DocumentBodyTag()**, **hw_DocumentSize()**.

hw_DocumentBodyTag (PHP 3>= 3.0.3, PHP 4 <= 4.0b1)

marca 'BODY' de un documento_hw

```
string hw_documentbodytag (int documento_hw)
```

Devuelve la marca BODY del documento. Si el documento es de tipo HTML la etiqueta BODY deberá imprimirse antes que el documento.

Vea también **hw_DocumentAttributes()**, **hw_DocumentSize()**.

hw_DocumentContent (PHP 3>= 3.0.8)

devuelve el contenido de un documento_hw

```
string hw_documentcontent (int documento_hw)
```

Devuelve el contenido del documento. Si el documento es de tipo HTML el contenido es todo lo que hay tras la etiqueta BODY. La información de las etiquetas HEAD y BODY se almacenan en el registro de objeto.

Vea también **hw_DocumentAttributes()**, **hw_DocumentSize()**, **hw_DocumentSetContent()**.

hw_DocumentSetContent (PHP 3>= 3.0.8)

fija/sustituye el contenido de un documento_hw

```
string hw_documentsetcontent (int documento_hw, string contenido)
```

Fija o sustituye el contenido del documento. Si el documento es de tipo HTML el contenido es todo lo que hay tras la etiqueta BODY. La información de las etiquetas HEAD y BODY se almacenan en el registro de objeto. Si proporciona también esta información en el contenido del documento, el servidor Hyperwave modificará el registro del objeto cuando sea insertado el documento. Probablemente no es una buena idea. Si esta función falla, el documento retendrá su anterior contenido.

Vea también **hw_DocumentAttributes()**, **hw_DocumentSize()**, **hw_DocumentContent()**.

hw_DocumentSize (PHP 3>= 3.0.3, PHP 4 <= 4.0b1)

tamaño de un documento_hw

```
int hw_document_size (int documento_hw)
```

Devuelve el tamaño en bytes del documento.

Vea también **hw_DocumentBodyTag()**, **hw_DocumentAttributes()**.

hw_ErrorMsg (PHP 3>= 3.0.3, PHP 4)

devuelve un mensaje de error

```
string hw_errormsg (int conexión)
```

Devuelve una cadena que contiene el último mensaje de error o 'No Error'. Si devuelve false es que la función fracasó. El mensaje está relacionado con el último comando.

hw_EditText (PHP 3>= 3.0.3, PHP 4)

recupera documento de texto

```
int hw_edittext (int conexión, int documento_hw)
```

Envía el documento de texto al servidor. El registro de objeto del documento no puede ser modificado mientras el documento es editado. Esta función sólo funcionará para objetos puros de texto. No abrirá ninguna conexión especial de datos y por tanto bloquea la conexión de control durante la transferencia.

Vea también **hw_PipeDocument()**, **hw_FreeDocument()**, **hw_DocumentBodyTag()**, **hw_DocumentSize()**, **hw_OutputDocument()**, **hw_GetText()**.

hw_Error (PHP 3>= 3.0.3, PHP 4)

número de error

```
int hw_error (int conexión)
```

Devuelve el último número de error. Si el valor devuelto es 0 no ha habido errores. El error está relacionado con el último comando.

hw_Free_Document (PHP 3>= 3.0.3, PHP 4)

libera un documento_hw

```
int hw_free_document (int documento_hw)
```

Libera la memoria ocupada por el documento Hyperwave.

hw_GetParents (PHP 3>= 3.0.3, PHP 4)

id de objeto de los padres

```
array hw_getparentsobj (int conexión, int IDobjeto)
```

Devuelve una tabla indizada de id de objeto. Cada una pertenece a un parent del objeto con ID *IDobjeto*.

hw_GetParentsObj (PHP 3>= 3.0.3, PHP 4)

registros de objeto de los padres

```
array hw_getparentsobj (int conexión, int IDobjeto)
```

Devuelve una tabla indizada de registros de objeto junto a una tabla asociativa con información estadística sobre estos. La tabla asociativa es el último elemento de la tabla devuelta. Cada registro de objeto pertenece a un parent del objeto con ID *IDobjeto*.

hw_GetChildColl (PHP 3>= 3.0.3, PHP 4)

id de objeto de colecciones hijas

```
array hw_getchildcoll (int conexión, int IDobjeto)
```

Devuelve una tabla de id de objeto. Cada ID de objeto pertenece a una colección hija de la colección con ID *IDobjeto*. La función no devolverá documentos hijos.

Vea también **hw_GetChildren()**, **hw_GetChildDocColl()**.

hw_GetChildCollObj (PHP 3>= 3.0.3, PHP 4)

registros de objeto de colecciones hijas

```
array hw_getchildcollobj (int conexión, int IDobjeto)
```

Devuelve una tabla de registros de objeto. Cada uno de ellos pertenece a una colección hija de la colección con ID *IDobjeto*. La función no devolverá documentos hijos.

Vea también **hw_ChildrenObj()**, **hw_GetChildDocCollObj()**.

hw_GetRemote (PHP 3>= 3.0.3, PHP 4)

Obtiene un documento remoto

```
int hw_getremote (int conexión, int IDobjeto)
```

Devuelve un documento remoto. Los documentos remotos en la notación de Hyperwave son los obtenidos de una fuente externa. Los documentos remotos típicos son páginas web externas o consultas a bases de datos. Para poder acceder a las fuentes externas a través de documentos remotos, el Hyperwave presenta el HGI (Hyperwave Gateway Interface - Interfaz de Pasarela de Hyperwave) que es similar al CGI. Actualmente, sólo se puede acceder a servidores ftp y http y a algunas bases de datos. Llamar a **hw_GetRemote()** devuelve el documento de la fuente externa. Si desea usar esta función debe familiarizarse con los HGI. Debería considerar el usar PHP en lugar del Hyperwave para acceder a fuentes externas. Añadir soporte de bases de datos a través de una pasarela Hyperwave sería más difícil que hacerlo en PHP.

Vea también **hw_GetRemoteChildren()**.

hw_GetRemoteChildren (PHP 3>= 3.0.3, PHP 4)

Obtiene el hijo del documento remoto

```
int hw_getremotechildren (int conexión, string registro de objeto)
```

Devuelve el hijo de un documento remoto. Los hijos de documentos remotos son en sí mismos documentos remotos. Esto tiene sentido cuando se afina una consulta de bases de datos y se explica en la Guía de Programación de Hyperwave. Si el número de hijos es 1 la función devolverá el documento en sí mismo formateado con la Interfaz de Pasarela de Hyperwave (HGI). Si el número de hijos es mayor de 1 devolverá una tabla de registros de objeto, con cada uno posible candidato para otra llamada a **hw_GetRemoteChildren()**. Dichos registros de objeto son virtuales y no existen en el servidor Hyperwave, y por lo tanto no poseen un ID de objeto válido. La apariencia exacta de dicho registro de objeto depende del HGI. Si desea usar esta función deberá estar muy familiarizado con los HGI. También debería considerar el uso del PHP en lugar de Hyperwave para acceder a fuentes externas. Añadir soporte de bases de datos a través de una pasarela de Hyperwave resulta más difícil que hacerlo en PHP.

Vea también **hw_GetRemote()**.

hw_GetSrcByDestObj (PHP 3>= 3.0.3, PHP 4)

Devuelve los enlaces que apuntan al objeto

```
array hw_getsrcbydestobj (int conexión, int IDobjeto)
```

Devuelve los registros de objeto de todos los enlaces que apuntan al objeto con ID *IDobjeto*. El objeto puede ser tanto un documento como un enlace de tipo destino.

Vea también **hw_GetAnchors()**.

hw_GetObject (PHP 3>= 3.0.3, PHP 4)

registro de objeto

```
array hw_getobject (int conexión, [int|array] IDobjeto, string consulta)
```

Devuelve el registro de objeto para el objeto cuyo ID es *IDobjeto* si el segundo parámetro es un entero. Si es una tabla la función devolverá una tabla de registros de objeto. En tal caso, el último parámetro, que es una cadena de consulta, también es evaluado.

La cadena de consulta tiene la sintaxis siguiente:

<expr> ::= "("<expr>"")|

```

"!"<expr> /* NO */
<expr> "||"<expr> /* O */
<expr> "&&"<expr> /* Y */
<atributo> <operador> <valor>
<atributo> ::= /* cualquier atributo (Título, Autor, TipoDocumento ...) */
<operador> ::= "="| /* igual */
"<"| /* menor que (comparación de cadenas) */
">"| /* mayor que (comparación de cadenas) */
"~"/* expresión regular */

```

La consulta permite seleccionar elementos de la lista de objetos dada. Al contrario de otras funciones de búsqueda, esta consulta no puede utilizar atributos indizados. El número de registros de objeto devueltos depende de la consulta y de si está permitido el acceso al objeto.

Vea también **hw_GetAndLock()**, **hw_GetObjectByQuery()**.

hw_GetAndLock (PHP 3>= 3.0.3, PHP 4)

devuelve registro de objeto y lo bloquea

```
string hw_getandlock (int conexión, int IDobjeto)
```

Devuelve el registro de objeto para el objeto con ID *IDobjeto*. También bloqueará el objeto, de modo que otros usuarios no podrán acceder al mismo hasta que sea desbloqueado.

Vea también **hw_Unlock()**, **hw_GetObject()**.

hw_GetText (PHP 3>= 3.0.3, PHP 4)

obtiene un documento de texto

```
int hw_gettext (int conexión, int IDobjeto [, mixed IDraiz/prefijo])
```

Devuelve el documento con ID de objeto *IDobjeto*. Si el documento tiene enlaces que pueden ser insertados, serán insertados ahora. El parámetro opcional *IDraiz/prefijo* puede ser una cadena o un entero. Si es un entero determina la forma en que los enlaces se insertan en el documento. Por defecto es 0 y los enlaces se crean a partir del nombre del objeto de destino de los mismos. Esto es útil para aplicaciones web. Si un enlace apunta a un objeto con nombre 'pelicula_internet' el enlace HTML será . La posición actual del objeto de destino en la jerarquía de documentos es despreciada. Tendrá que ajustar su navegador web para que reescriba dicho URL a, por ejemplo, '/mi_script.php3/pelicula_internet'. 'mi_script.php3' deberá evaluar \$PATH_INFO y recuperar el documento. Todos los enlaces tendrán el prefijo '/mi_script.php3'. Si no desea este efecto puede fijar el parámetro opcional *IDraiz/prefijo* al prefijo que desee en su lugar. En este caso deberá ser una cadena.

Si el *IDraiz/prefijo* es un entero distinto de 0, el enlace se construye con todos los nombres de objeto comenzando con el objeto de id *IDraiz/prefijo*, separado por una barra relativa al objeto actual. Si por ejemplo el documento anterior 'pelicula_internet' está situado en 'a-b-c-pelicula_internet' donde '-' es el separador entre niveles jerárquicos en el servidor Hyperwave y el documento fuente está situado en 'a-b-d-fuente', el enlace HTML resultante sería: . Esto es útil cuando desea bajarse el contenido completo del servidor al disco y mapear la jerarquía de documentos en el sistema de archivos.

Esta función sólo trabajará en documentos de texto puros. No se abrirá una conexión de datos especial y por tanto bloqueará la conexión de control durante la transferencia.

Vea también **hw_PipeDocument()**, **hw_FreeDocument()**, **hw_DocumentBodyTag()**, **hw_DocumentSize()**, **hw_OutputDocument()**.

hw_GetObjectByQuery (PHP 3>= 3.0.3, PHP 4)

buscar objeto

```
array hw_getobjectbyquery (int conexión, string consulta, int max_resultados)
```

Busca objetos en todo el servidor y devuelve una tabla de id de objeto. El número máximo de resultados será *max_resultados*. Si *max_resultados* vale -1, el número máximo de resultados es ilimitado.

La consulta funcionará sólo con atributos indizados.

Vea también **hw_GetObjectByQueryObj()**.

hw_GetObjectByQueryObj (PHP 3>= 3.0.3, PHP 4)

buscar objeto

```
array hw_getobjectbyqueryobj (int conexión, string consulta, int max_resultados)
```

Busca objetos en todo el servidor y devuelve una tabla de registros de objeto. El número máximo de resultados será *max_resultados*. Si *max_resultados* vale -1, el número máximo de resultados es ilimitado.

La consulta funcionará sólo con atributos indizados.

Vea también **hw_GetObjectByQuery()**.

hw_GetObjectByQueryColl (PHP 3>= 3.0.3, PHP 4)

buscar objeto en colección

```
array hw_getobjectbyquerycoll (int conexión, int IDobjeto, string consulta, int max_resultados)
```

Busca objetos en la colección cuyo ID es *IDobjeto* y devuelve una tabla de ID de objeto. El número máximo de resultados será *max_resultados*. Si *max_resultados* vale -1, el número máximo de resultados es ilimitado.

La consulta funcionará sólo con atributos indizados.

Vea también **hw_GetObjectByQueryCollObj()**.

hw_GetObjectByQueryCollObj (PHP 3>= 3.0.3, PHP 4)

buscar objeto en colección

```
array hw_getobjectbyquerycollobj (int conexión, int IDobjeto, string consulta, int max_resultados)
```

Busca objetos en la colección cuyo ID es *IDobjeto* y devuelve una tabla de registros de objeto. El número máximo de resultados será *max_resultados*. Si *max_resultados* vale -1, el número máximo de resultados es ilimitado.

La consulta funcionará sólo con atributos indizados.

Vea también **hw_GetObjectByQueryColl()**.

hw_GetChildDocColl (PHP 3>= 3.0.3, PHP 4)

id de objeto de documentos hijos de una colección

```
array hw_getchilddoccoll (int conexión, int IDobjeto)
```

Devuelve una tabla de id de objeto para los documentos hijos de una colección.

Vea también **hw_GetChildren()**, **hw_GetChildColl()**.

hw_GetChildDocCollObj (PHP 3>= 3.0.3, PHP 4)

registros de objeto de documentos hijos de una colección

```
array hw_getchilddoccollobj (int conexión, int IDobjeto)
```

Devuelve una tabla de registros de objeto para los documentos hijos de una colección.

Vea también **hw_ChildrenObj()**, **hw_GetChildCollObj()**.

hw_GetAnchors (PHP 3>= 3.0.3, PHP 4)

id de objeto de los enlaces de un documento

```
array hw_getanchors (int conexión, int IDobjeto)
```

Devuelve una tabla de id de objeto con los enlaces del documento cuyo ID es *IDobjeto*.

hw_GetAnchorsObj (PHP 3>= 3.0.3, PHP 4)

registros de objeto de los enlaces de un documento

```
array hw_getanchorsobj (int conexión, int IDobjeto)
```

Devuelve una tabla de registros de objeto con los enlaces del documento cuyo ID es *IDobjeto*.

hw_Mv (PHP 3>= 3.0.3, PHP 4)

mueve objetos

```
int hw_mv (int conexión, array tabla de id de objeto, int id origen, int id destino)
```

Mueve los objetos cuyas id se especifican en el segundo parámetro desde la colección con id *id origen* hasta la colección con el id *id destino*. Si el id de destino es 0, los objetos serán disociados de la colección origen. Si esta fuese la última instancia del objeto, este sería borrado. Si desea borrar todas las instancias de una vez, utilice **hw_deleteobject()**.

El valor devuelto es el número de objetos movidos.

Vea también **hw_cp()**, **hw_deleteobject()**.

hw_Identify (PHP 3>= 3.0.3, PHP 4)

identificarse como usuario

```
int hw_identify (string nombre, string clave)
```

Le identifica como el usuario *nombre* y *clave*. La identificación sólo es válida para la sesión actual. No pienso que esta función sea necesaria con mucha frecuencia. En muchos casos será más fácil identificarse al abrir la conexión.

Vea también **hw_Connect()**.

hw_InCollections (PHP 3>= 3.0.3, PHP 4)

comprueba si los id de objeto están en las colecciones

```
array hw_incollections (int conexión, array tabla_id_objeto, array tabla_id_colecciones,
int colecciones_devueltas)
```

Comprueba si el conjunto de objetos (documentos o colecciones) especificado por el parámetro *tabla_id_objeto* es parte de las colecciones enumeradas en *tabla_id_colecciones*. Cuando el cuarto parámetro *colecciones_devueltas* es 0, el subconjunto de id de objeto que es parte de las colecciones (es decir, los documentos o colecciones hijos de una o más colecciones de id de colecciones o de sus subcolecciones, recursivamente) es devuelto en forma de tabla. Cuando el cuarto parámetro es 1, sin embargo, el conjunto de colecciones que tienen uno o más objetos de este subconjunto como hijos es devuelto como tabla. Esta opción permite a un cliente, p. ej., remarcar en una visualización gráfica la parte de la jerarquía de colecciones que contiene las coincidencias de una consulta previa.

hw_Info (PHP 3>= 3.0.3, PHP 4)

información sobre conexión

```
string hw_info (int conexión)
```

Devuelve información sobre la conexión actual. La cadena devuelta tiene el siguiente formato: <Cadenaservidor>, <Anfitrión>, <Puerto>, <Usuario>, <Puerto del Usuario>, <Intercambio de bytes>

hw_InsColl (PHP 3>= 3.0.3, PHP 4)

insertar colección

```
int hw_inscoll (int conexión, int IDobjeto, array tabla_objetos)
```

Inserta una nueva colección con los atributos de la *tabla_objetos* en la colección cuyo ID de objeto es *IDobjeto*.

hw_InsDoc (PHP 3>= 3.0.3, PHP 4)

insertar documento

```
int hw_insdoc (int conexión, int IDpadre, string registro_de_objeto, string texto)
```

Inserta un nuevo documento con los atributos del *registro_de_objeto* en la colección cuyo ID de objeto es *IDpadre*. Esta función inserta tanto un registro de objeto sólo como un registro de objeto y el texto puro en ASCII dado en *texto* si este es especificado. si desea insertar un documento general de cualquier tipo, utilice en su lugar **hw_insertdocument()**.

Vea también **hw_InsertDocument()**, **hw_InsColl()**.

hw_InsertDocument (PHP 3>= 3.0.3, PHP 4)

subir cualquier objeto

```
int hw_insertdocument (int conexión, int id_padre, int documento_hw)
```

Sube un documento a la colección dada por *id_padre*. El documento debe ser creado antes con la función **hw_NewDocument()**. Asegúrese que el registro de objeto del nuevo documento contenga al menos los atributos: Type, DocumentType, Title y Name (así, en inglés). Posiblemente desee fijar también el MimeType. La función devuelve la id de objeto del nuevo documento, o false.

Vea también **hw_PipeDocument()**.

hw_InsertObject (PHP 3>= 3.0.3, PHP 4)

inserta un registro de objeto

```
int hw_insertobject (int conexión, string reg de objeto, string parametro)
```

Inserta un objeto en el servidor. Este puede consistir en cualquier objeto hyperwave válido. Vea la documentación sobre el HG-CSP si desea información detallada sobre cuáles tienen que ser los parámetros.

Nota: Si se desea insertar un enlace, el atributo Position siempre se fijará a un valor comienzo/final o a 'invisible'. Las posiciones invisibles se necesitan si la anotación no tiene enlace correspondiente en el texto anotado.

Vea también **hw_PipeDocument()**, **hw_InsertDocument()**, **hw_InsDoc()**, **hw_InsColl()**.

hw_mapid (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Mapea in id global a un id virtual local

```
int hw_mapid (int conexión, int id servidor, int id objeto)
```

Mapea un id de objeto global en un servidor hyperwave, incluso con aquellos a los que no se ha conectado con **hw_connect()**, sobre un id virtual de objeto. Este id virtual se puede utilizar como cualquier otro id de objeto, p. ej. para obtener el registro de objeto por medio de **hw_getobject()**. El id de servidor es la primera parte del id global de objeto (GOid) de aquel que es realmente el número IP expresado como entero.

Nota: Para usar esta función deberá activar el indicador F_DISTRIBUTED, que actualmente sólo se puede fijar en tiempo de compilación desde hg_comm.c. Por defecto está inactivo. Lea el comentario al principio de hg_comm.c

hw_Modifyobject (PHP 3>= 3.0.7, PHP 4 >= 4.0b2)

modifica el registro de objeto

```
int hw_modifyobject (int conexión, int objeto_a_cambiar, array eliminar, array añadir, int modo)
```

Este comando permite eliminar, añadir, o modificar atributos individuales de un registro de objeto. El objeto está especificado por el ID de objeto *objeto_a_cambiar*. La primera tabla, *eliminar*, es la lista de atributos a eliminar. La segunda tabla, *añadir*, es la lista de atributos a añadir. Para modificar un atributo, hay que borrar el antiguo y añadir uno nuevo. **hw_modifyobject()** siempre eliminará los atributos antes de añadir los nuevos excepto si el valor del atributo a eliminar no es una cadena o una tabla.

El último parámetro determina si la modificación se realiza de manera recursiva. 1 quiere decir que sea recursiva. Si alguno de los objetos no se puede modificar, será ignorado sin avisar. Incluso **hw_error()** podría no indicar un error aunque alguno de los objetos no pueda ser modificado.

Las claves de ambas tablas son los nombres de los atributos. El valor de cada elemento de la tabla puede ser una tabla, una cadena o cualquier otra cosa. Si es una tabla, cada valor de atributo se construye como la clave de cada elemento más dos puntos y el valor de cada elemento. Si es una cadena se toma como valor del atributo. Una cadena vacía producirá la supresión completa del atributo. Si el valor no es ni cadena ni tabla, sino otra cosa, p. ej. un entero, no se realizará operación alguna en el atributo. Esto es necesario se desea añadir un atributo completamente nuevo, no solamente un nuevo valor para un atributo existente. Si la tabla eliminar contuviera una cadena vacía para dicho atributo, este se intentaría suprimir, lo que fallaría porque este no existe. La siguiente adición de un nuevo valor para dicho atributo también fallará. Fijando el valor para dicho atributo p. ej. a 0 hará que ni siquiera se intente eliminar, pero funcionará la adición del mismo.

Si desea cambiar el atributo 'Nombre' con el valor actual 'libros' por el de 'artículos' deberá crear dos tablas y llamar a **hw_modifyobject()**.

Ejemplo 1. modificando un atributo

```
// $conexion es una conexión con el servidor Hyperwave
// $idobj es la ID del objeto a modificar
$tablasupr = array( "Nombre" => "libros" );
$stablaanad = array( "Nombre" => "artículos" );
$hw_modifyobject($conexion, $idobj, $tablasupr, $stablaanad);
```

Para borrar/añadir un par nombre=valor de/a el registro de objeto, simplemente pase la tabla eliminar/añadir y fije el último/tercer parámetro a tabla vacía. Si el atributo es el primero con dicho nombre que se añade, fije el valor del atributo en la tabla eliminar a un valor entero.

Ejemplo 2. añadiendo un atributo completamente nuevo

```
// $conexion es una conexión con el servidor Hyperwave
// $idobj es la ID del objeto a modificar
$tablasupr = array( "Nombre" => 0 );
$stablaanad = array( "Nombre" => "artículos" );
```

```
$hw_modifyobject($conexion, $idobj, $stablasupr, $stablaanad);
```

Nota: Los atributos plurilingües, p. ej. 'Título', se pueden modificar de dos maneras. O bien proporcionando los valores de los atributos en su forma nativa 'lenguaje':'título', bien proporcionando una tabla con los elementos para cada lenguaje según se describe más arriba. El ejemplo anterior podría quedar entonces:

Ejemplo 3. modificando el atributo Título

```
$stablasupr = array("Título" => "es:Libros");
$stablaanad = array("Título" => "es:Artículos");
$hw_modifyobject($conexion, $idobj, $stablasupr, $stablaanad);
```

O

Ejemplo 4. modificando el atributo Título

```
$stablasupr = array("Título" => array("es" => "Libros"));
$stablaanad = array("Título" => array("es" => "Artículos", "ge"=>"Artikel"));
$hw_modifyobject($conexion, $idobj, $stablasupr, $stablaanad);
```

Esto elimina el título español 'Libros' y añade el título español 'Artículos' y el título alemán 'Artikel'.

Ejemplo 5. eliminando atributos

```
$stablasupr = array("Título" => "");
$stablaanad = array("Título" => "es:Artículos");
$hw_modifyobject($conexion, $idobj, $stablasupr, $stablaanad);
```

Nota: Esto eliminará todos los atributos con el nombre 'Título' y añadirá un nuevo atributo 'Título'. Esto es útil cuando se desea eliminar atributos de forma recursiva.

Nota: Si necesita eliminar todos los atributos con un cierto nombre tendrá que pasar una cadena vacía como el valor del atributo.

Nota: Sólo los atributos 'Title', 'Description' y 'Keyword' (así, en inglés) manejarán de forma apropiada el prefijo de idioma. Si estos atributos no llevan prefijo de idioma, se les asignará el prefijo 'xx'.

Nota: El atributo 'Name' es bastante especial. En algunos casos no puede ser completamente eliminado. Obtendrá un mensaje de error 'Change of base attribute' ('Cambio de atributo base', no está muy claro cuando ocurre). Por tanto, tendrá siempre que añadir un nuevo atributo Name primero y luego eliminar el anterior.

Nota: No debe rodear esta función de llamadas a **hw_getandlock()** ni a **hw_unlock()**. **hw_modifyobject()** ya lo hace internamente.

Devuelve TRUE si no hay error o FALSE en caso contrario.

hw_New_Document (PHP 3>= 3.0.3, PHP 4)

crear nuevo documento

```
int hw_new_document (string registro_de_objeto, string datos_documento, int tama_documento)
```

Devuelve un nuevo documento Hyperwave en el que los datos del documento están fijados a *datos_documento* y el registro de objeto a *registro_de_objeto*. La longitud de los *datos_documento* debe pasarse en *tama_documento*. Esta función no inserta el documento en el servidor Hyperwave.

Vea también **hw_FreeDocument()**, **hw_DocumentSize()**, **hw_DocumentBodyTag()**, **hw_OutputDocument()**, **hw_InsertDocument()**.

hw_Objrec2Array (PHP 3>= 3.0.3, PHP 4)

convierte atributos de registro de objeto a tabla de objetos

```
array hw_objrec2array (string registro_de_objeto)
```

Convierte un *registro_de_objeto* en una tabla de objetos. Las claves de la tabla resultante son los nombres de los atributos. Los atributos múltiples como 'Título' en distintos idiomas forman su propia tabla. Las claves de esta tabla son las partes a la izquierda de los dos puntos del valor del atributo. Actualmente, sólo los atributos 'Title', 'Description' y 'Keyword' (así, en inglés) son correctamente tratados. Otros atributos múltiples generan una tabla indizada. Actualmente, sólo el atributo 'Group' es tratado correctamente.

Vea también **hw_array2objrec()**.

hw_OutputDocument (PHP 3>= 3.0.3, PHP 4 <= 4.0b1)

muestra el documento_hw

```
int hw_outputdocument (int documento_hw)
```

Muestra el documento sin la etiqueta BODY.

hw_pConnect (PHP 3>= 3.0.3, PHP 4)

hacer una conexión de base de datos permanente

```
int hw_pconnect (string servidor, int puerto, string usuario, string clave)
```

Devuelve un índice de conexión si hay éxito, o false si la conexión no puede hacerse. Abre una conexión permanente a un servidor Hyperwave. Cada uno de los argumentos debe ser una cadena entrecomillada excepto el número de puerto. El argumento *usuario* y la *clave* son opcionales y pueden omitirse. En tal caso no se realizará ninguna identificación con el servidor. Es similar a la identificación anónima del usuario. Esta función devuelve un índice de conexión que se necesita para otras funciones de Hyperwave. Se pueden tener varias conexiones permanentes abiertas a la vez.

Vea también **hw_Connect()**.

hw_PipeDocument (PHP 3>= 3.0.3, PHP 4)

recupera cualquier documento

```
int hw_pipedocument (int conexión, int IDobjeto)
```

Devuelve el documento Hyperwave cuyo ID de objeto es *IDobjeto*. Si el documento tiene enlaces que pueden ser insertados, deberán haberse insertado ya. El documento será transferido a través de una conexión de datos especial que no bloquea la conexión de control.

Vea también **hw_GetText()** para saber más sobre inserción de enlaces, **hw_FreeDocument()**, **hw_DocumentSize()**, **hw_DocumentBodyTag()**, **hw_OutputDocument()**.

hw_Root (PHP 3>= 3.0.3, PHP 4)

ID del objeto raíz

```
int hw_root ()
```

Devuelve la ID de objeto de la colección hiper-raíz. Actualmente siempre vale 0. La colección hija de la hiper-raíz es la colección raíz del servidor al que se ha conectado.

hw_Unlock (PHP 3>= 3.0.3, PHP 4)

desbloquea objeto

```
int hw_unlock (int conexión, int IDobjeto)
```

Desbloquea un documento para que otros usuarios puedan acceder al mismo de nuevo.

Vea también **hw_GetAndLock()**.

hw_Who (PHP 3>= 3.0.3, PHP 4)

Lista de los usuarios actualmente conectados

```
int hw_who (int conexión)
```

Devuelve una tabla de los usuarios actualmente conectados al servidor Hyperwave. Cada elemento de esta tabla contiene en sí mismo los elementos ID, nombre, sistema, onSinceDate (conectadoDesdeFecha), onSinceTime (conectadoDesdeHora), TotalTime (TiempoTotal) y self (propio). 'self' es 1 si esta línea corresponde al usuario que realizó la petición.

hw_Username (unknown)

nombre del usuario actualmente conectado

```
string hw_getusername (int conexión)
```

Devuelve el nombre de usuario de la conexión.

XXX. Funciones para ICAP - Internet Calendar Application Protocol

Para hacer funcionar estas funciones, deberá compilar el PHP con `-with-icap`. Eso indicará que se precisa la instalación de la librería ICAP. Obtenga la última versión en <http://icap.chek.com/>, compílela e instálela.

icap_open (PHP 4 >= 4.0b4)

Abre una conexión ICAP

```
stream icap_open (string calendario, string usuario, string clave, string opciones)
```

Si hay éxito devuelve un stream (flujo) ICAP, o false en caso de error.

icap_open() abre una conexión ICAP con el servidor de *calendario* especificado. Si se especifica el parámetro opcional *opciones*, también éste es pasado a dicho buzón.

icap_close (unknown)

Cierra un stream ICAP

```
int icap_close (int stream_icap, int banderas)
```

Cierra el stream ICAP dado.

icap_fetch_event (PHP 4 >= 4.0b4)

Obtiene un evento del stream de calendario/

```
object icap_fetch_event (stream stream_icap, int id_evento, options opciones)
```

icap_fetch_event() obtiene el evento del stream de calendario especificado por el parámetro *id_evento*.

Devuelve un objeto de evento compuesto por:

- int id - ID de dicho evento.
- int public - TRUE si el evento es público, FALSE si es privado.
- string category - Cadena con la categoría del evento.
- string title - Cadena de título del evento.
- string description - Cadena de descripción del evento.
- int alarm - Número de minutos antes que el evento envíe una alarma/recordatorio.
- object start - Objeto que contiene una entrada datetime (fecha/hora).
- object end - Objeto que contiene una entrada datetime.

Todas las entradas datetime consisten en un objeto compuesto por:

- int year - año
- int month - mes
- int mday - día del mes
- int hour - hora
- int min - minutos
- int sec - segundos

icap_list_events (PHP 4 >= 4.0RC1)

Devuelve una lista de eventos entre dos instantes dados

```
array icap_list_events (stream stream_icap, datetime instante_inicio, datetime instante_final)
```

Devuelve una tabla de ID de evento que están entre los dos instantes dados.

La función **icap_list_events()** toma un instante de inicio y uno de final para un stream de calendario. Se devuelve una tabla de ID de evento que están entre los instantes dados.

Todas las entradas datetime consisten en un objeto compuesto por:

- int year - año
- int month - mes
- int mday - día del mes
- int hour - hora
- int min - minutos
- int sec - segundos

icap_store_event (PHP 4 >= 4.0b4)

Almacena un evento en un calendario ICAP

```
int icap_store_event (int stream_icap, object evento)
```

icap_store_event() Guarda un evento en un calendario ICAP. Un objeto de evento consiste en:

- int public - 1 si es público, 0 si es privado;
- string category - Cadena con la categoría del evento.
- string title - Cadena de título del evento.
- string description - Cadena de descripción del evento.
- int alarm - Número de minutos antes que el evento envíe una alarma/recordatorio.
- object start - Objeto que contiene una entrada datetime (fecha/hora).
- object end - Objeto que contiene una entrada datetime.

Todas las entradas datetime consisten en un objeto compuesto por:

- int year - año
- int month - mes
- int mday - día del mes

- int hour - hora
- int min - minutos
- int sec - segundos

Devuelve true en caso de éxito y false en caso de error.

icap_delete_event (PHP 4 >= 4.0b4)

Borra un evento de un calendario ICAP

```
int icap_delete_event (int id_evento)
```

icap_delete_event() borra el evento de calendario especificado por el *id_evento*.

Devuelve true.

icap_snooze (PHP 4 >= 4.0b4)

Apaga la alarma de un evento

```
int icap_snooze (int id_evento)
```

icap_snooze() apaga la alarma del evento de calendario especificado por el *id_evento*.

Returns true.

icap_list_alarms (PHP 4 >= 4.0b4)

Devuelve una lista de los eventos que una alarma ha disparado en el instante dado

```
array icap_list_alarms (stream stream_icap, datetime instante_alarma)
```

Devuelve una tabla de identificadores de evento para los que una alarma debiera apagarse en el instante indicado.

La función **icap_list_alarms()** toma una estructura datetime para un stream de calendario. Se devuelve una tabla de los identificadores de evento de todas las alarmas que debieran apagarse en el instante indicado.

Todas las entradas datetime consisten en un objeto compuesto por:

- int year - año
- int month - mes
- int mday - día del mes
- int hour - hora
- int min - minutos
- int sec - segundos

XXXI. Funciones de imágenes

Puede usar las funciones de imágenes de PHP para obtener el tamaño de imágenes JPEG, GIF y PNG, y si tiene la librería GD (disponible en <http://www.boutell.com/gd/>) además será capaz de crear y manipular imágenes.

GetImageSize (PHP 3, PHP 4)

Obtiene el tamaño de una imagen GIF, JPG o PNG

```
array getimagesize (string filename [, array imageinfo])
```

La función **GetImageSize()** determinará el tamaño de cualquier fichero de imagen GIF, JPG o PNG y devolverá sus dimensiones junto al tipo de fichero en una cadena de texto que pueda ser usada en una marca HTML IMG normal.

Devuelve una matriz con 4 elementos. El índice 0 contiene la anchura de la imagen en pixels. El índice 1 contiene la altura. El índice 2 es una marca indicando el tipo de imagen. 1 = GIF, 2 = JPG, 3 = PNG. El índice 3 es una cadena de texto con el string correcto "height=xxx width=xxx" para ser usado directamente en una marca IMG.

Ejemplo 1. GetImageSize

```
<?php $size = GetImageSize("img/flag.jpg"); ?>
<IMG SRC="img/flag.jpg" <?php echo $size[3]; ?>
```

El parámetro opcional *imageinfo* permite extraer información adicional del fichero de imagen. Actualmente esto devolverá las diferentes marcas APP de los JPG en una matriz asociada. Algunos programas usan estas marcas APP para incluir información textual en las imágenes. Uno bastante común incluye información IPTC <http://www.iptc.org/> en la marca APP13. Puede usar la función **iptcparse()** para convertir la marca binaria APP13 en algo leible.

Ejemplo 2. GetImageSize devolviendo IPTC

```
<?php
    $size = GetImageSize("testimg.jpg",&$info);
    if (isset($info["APP13"])) {
        $iptc = iptcparse($info["APP13"]);
        var_dump($iptc);
    }
?>
```

Nota: Esta función no requiere la librería de imágenes GD.

ImageArc (PHP 3, PHP 4)

Dibuja una elipse parcial

```
int imagearc (int im, int cx, int cy, int w, int h, int s, int e, int col)
```

ImageArc dibuja una elipse parcial centrada en cx, cy (la esquina superior izquierda es 0,0) en la imagen que representa im. w y h especifican la anchura y altura respectivamente mientras que los puntos de inicio y final vienen indicados por los parámetros s y e en grados.

ImageChar (PHP 3, PHP 4)

Dibuja un carácter horizontalmente

```
int imagechar (int im, int font, int x, int y, string c, int col)
```

ImageChar dibuja el primer carácter de *c* en la imagen identificada como *id* con su esquina superior izquierda en *x*, *y* (arriba izquierda es 0,0) con el color *col*. Si la fuente es 1, 2, 3, 4 o 5 se usa una fuente predefinida (números mayores corresponden con tamaños mayores).

Vea también **imageloadfont()**.

ImageCharUp (PHP 3, PHP 4)

Dibuja un carácter vertical

```
int imagecharup (int im, int font, int x, int y, string c, int col)
```

ImageCharUp dibuja el carácter *c* verticalmente en la imagen identificada como *im* en las coordenadas *x*, *y* (arriba izquierda es 0,0) con el color *col*. Si la fuente es 1, 2, 3, 4 o 5 se usa una fuente predefinida.

Vea también **imageloadfont()**.

ImageColorAllocate (PHP 3, PHP 4)

Reserva un color para una imagen

```
int imagecolorallocate (int im, int red, int green, int blue)
```

ImageColorAllocate devuelve un identificador del color representado por la mezcla de los componentes RGB dados. El parámetro *im* es el resultado de la función **imagecreate()**. ImageColorAllocate tiene que ser invocada para crear cada color que va a ser usado por la imagen que representa *im*.

```
$white = ImageColorAllocate($im, 255,255,255);
$black = ImageColorAllocate($im, 0,0,0);
```

ImageColorAt (PHP 3, PHP 4)

Obtiene el índice del color de un pixel

```
int imagecolorat (int im, int x, int y)
```

Devuelve el índice del color del pixel especificado en la posición de la imagen.

Vea también **imagecolorset()** y **imagecolorsforindex()**.

ImageColorClosest (PHP 3, PHP 4)

Obtiene el índice del color más cercano al color especificado

```
int imagecolorclosest (int im, int red, int green, int blue)
```

Devuelve el índice del color de la paleta de la imagen que sea más "cercano" al valor RGB especificado.

La "distancia" entre el color deseado y cada color de la paleta es calculada como si los valores RGB representasen puntos en un espacio tridimensional.

Vea también **imagecolorexact()**.

ImageColorExact (PHP 3, PHP 4)

Devuelve el índice del color especificado

```
int imagecolorexact (int im, int red, int green, int blue)
```

Devuelve el índice del color especificado de la paleta de la imagen.

Si el color no existe en la paleta de la imagen, se devuelve el valor -1.

Vea también **imagecolorclosest()**.

ImageColorResolve (PHP 3>= 3.0.2, PHP 4)

Devuelve el índice del color especificado o su posible alternativa más cercana

```
int imagecolorresolve (int im, int red, int green, int blue)
```

Esta función garantiza el resultado de un índice de color para un color solicitado, ya sea el color exacto o su alternativa más cercana.

Vea también **imagecolorclosest()**.

ImageColorSet (PHP 3, PHP 4)

Establece el color para el índice de la paleta especificado

```
bool imagecolorset (int im, int index, int red, int green, int blue)
```

Establece el índice especificado de la paleta con el color introducid. Esto es útil para crear efectos de relleno en imágenes con paletas sin la sobrecarga de tener que realizar el relleno.

Vea también **imagecolorat()**.

ImageColorsForIndex (PHP 3, PHP 4)

Obtiene los colores de un índice

```
array imagecolorsforindex (int im, int index)
```

Devuelve una matriz asociativa con las claves red, green y blue que contienen los valores apropiados para el color especificado en el índice.

Vea también **imagecolorat()** y **imagecolorexact()**.

ImageColorsTotal (PHP 3, PHP 4)

Encuentra el número de colores de la paleta de una imagen

```
int imagecolorstotal ( int im )
```

Encuentra el número de colores de la paleta de una imagen.

Vea también **imagecolorat()** y **imagecolorsforindex()**.

ImageColorTransparent (PHP 3, PHP 4)

Define un color como transparente

```
int imagecolortransparent ( int im [, int col] )
```

ImageColorTransparent establece como color transparente a *col* en la imagen *im*. *im* es el identificador de imagen devuelto por **imagecreate()** y *col* es el identificador de color devuelto por **imagecolorallocate()**.

Se devuelve el identificador del color transparente (o el actual, si no se especifica ninguno).

ImageCopyResized (PHP 3, PHP 4)

Copia y redimensiona parte de una imagen

```
int imagecopyresized ( int dst_im, int src_im, int dstX, int dstY, int srcX, int srcY, int dstW, int dstH, int srcW, int srcH )
```

ImageCopyResize copia una porción rectangular de una imagen hacia otra imagen. *dst_im* es la imagen de destino, *src_im* es el identificador de la imagen origen. Si la altura y anchura de las coordenadas de origen y destino difieren se realizará un estrechamiento o un estiramiento apropiado del fragmento de la imagen. Las coordenadas van localizadas sobre la esquina superior izquierda. Esta función se puede usar para copiar regiones dentro de la misma imagen (si *dst_im* es igual que *src_im*) pero si las regiones se solapan los resultados serán impredecibles.

ImageCreate (PHP 3, PHP 4)

Crea una nueva imagen

```
int imagecreate ( int x_size, int y_size )
```

ImageCreate devuelve un identificador de imagen representando una imagen en blanco de tamaño *x_size* por *y_size*.

ImageCreateFromGif (PHP 3, PHP 4)

Crea una nueva imagen desde un fichero o una URL

```
int imagecreatefromgif (string filename)
```

imagecreatefromgif() devuelve un identificador de imagen representando la imagen obtenida del nombre del fichero dado.

imagecreatefromgif() devuelve una cadena vacía si hay algún fallo. Además muestra un mensaje de error, que desafortunadamente se representa como un link roto en un navegador. Para depurarlo fácilmente el siguiente ejemplo producirá un error de GIF:

Ejemplo 1. Ejemplo de control de un error durante la creación (cortesía vic@zymsys.com)

```
function LoadGif($imgname)
{
    $im = @imagecreatefromgif($imgname); /* Attempt to open */
    if ($im == "") { /* See if it failed */
        $im = ImageCreate(150,30); /* Create a blank image */
        $bgc = ImageColorAllocate($im,255,255,255);
        $tc = ImageColorAllocate($im,0,0,0);
        ImageFilledRectangle($im,0,0,150,30,$bgc);
        ImageString($im,1,5,5,"Error loading $imgname",$tc); /* Output an errmsg */
    }
    return $im;
}
```

Nota: Desde que todo el soporte a GIFs ha sido eliminado en la librería GD a partir de la versión 1.6, esta función no está disponible si está usando esa versión de la librería GD.

ImageDashedLine (PHP 3, PHP 4)

Dibuja una línea discontinua

```
int imagedashedline (int im, int x1, int y1, int x2, int y2, int col)
```

ImageLine dibuja una línea discontinua desde x1,y1 hasta x2, y2 (arriba izquierda es 0.0) en la imagen im con el color col.

Vea también **imageline()**.

ImageDestroy (PHP 3, PHP 4)

Destruye una imagen

```
int imagedestroy (int im)
```

ImageDestroy libera la memoria asociada a la imagen im. im es la imagen devuelta por la función **imagecreate()**.

ImageFill (PHP 3, PHP 4)

Relleno

```
int imagefill (int im, int x, int y, int col)
```

ImageFill realiza un relleno empezando en la coordenada x,y (arriba izquierda es 0,0) con el color col en la imagen im.

ImageFilledPolygon (PHP 3, PHP 4)

Dibuja un polígono relleno

```
int imagefilledpolygon (int im, array points, int num_points, int col)
```

ImageFilledPolygon crea un polígono relleno en la imagen im, points es una matriz PHP conteniendo los vértices del polígono, por ejemplo. points[0] = x0, points[1] = y0, points[2] = x1, points[3] = y1, etc. num_points es el número total de vértices.

ImageFilledRectangle (PHP 3, PHP 4)

dibuja un rectángulo relleno

```
int imagefilledrectangle (int im, int x1, int y1, int x2, int y2, int col)
```

ImageFilledRectangle crea un rectángulo relleno con color col en la imagen im comenzando con las coordenadas superiores izquierdas x1, y1 y finalizando en las coordenadas inferiores derechas x2, y2. 0,0 es la esquina superior izquierda de la imagen.

ImageFillToBorder (PHP 3, PHP 4)

Relleno de un color específico

```
int imagefilltoborder (int im, int x, int y, int border, int col)
```

ImageFillToBorder realiza un relleno hasta el color del borde que está definido por border. El punto de inicio para el relleno es x,y (arriba izquierda es 0,0) y la región se rellena con el color col.

ImageFontHeight (PHP 3, PHP 4)

Devuelve la altura de una fuente

```
int imagefontheight (int font)
```

Devuelve la altura en pixels de un carácter en un fuente específica.

Vea también **imagefontwidth()** y **imageloadfont()**.

ImageFontWidth (PHP 3, PHP 4)

Devuelve la anchura de una fuente

```
int imagefontwidth (int font)
```

Devuelve la anchura en pixels de un carácter en un fuente específica.

Vea también **imagefontheight()** y **imageloadfont()**.

ImageGif (PHP 3, PHP 4)

Envía una imagen al navegador o a un fichero

```
int imagegif (int im, string filename)
```

imagegif() crea el fichero GIF en *filename* a partir de la imagen *im*. El parámetro *im* es el resultado de usar la función **imagecreate()**.

El formato de la imagen será GIF87a a menos que la imagen se halla hecho transparente con **imagecolortransparent()**, en cuyo caso el formato de la imagen será GIF89a.

El parametro del nombre del fuichero es opcional, y si se deja en blanco, la imagen será mostrada directamente. Enviando un tipo de imagen/gif usando la función **header()**, puede crear un script PHP que muestre imágenes GIF directamente.

Nota: Desde que todo el soporte a GIFs ha sido eliminado en la libreria GD a partir de la versión 1.6, esta función no está disponible si está usando esa versión de la libreria GD.

ImageInterlace (PHP 3, PHP 4)

Activa o desactiva el entrelazado

```
int imageinterlace (int im [, int interlace])
```

ImageInterlace() activa o desactiva el bit de entrelazado. Si interlace es 1 la imagen im será entrelazada, y si interlace es 0 el bit de entrelazado se desactiva.

Esta función devuelve como ha cambiado el estado del bit de entrelazado de la imagen.

ImageLine (PHP 3, PHP 4)

Dibuja una línea

```
int imageline (int im, int x1, int y1, int x2, int y2, int col)
```

ImageLine dibuja una línea desde x1,y1 hasta x2,y2 (arriba izquierda es 0,0) en la imagen im con el color col.

Vea también **imagecreate()** y **imagecolorallocate()**.

ImageLoadFont (PHP 3, PHP 4)

Carga una fuente nueva

```
int imagedloadfont (string file)
```

ImageLoadFont carga una fuente de bitmaps definida por el usuario y devuelve un identificador para esa fuente (que siempre es mayor de 5, de forma que no pueda entrar en conflicto con las fuentes predefinidas)..

El formato del fichero de la fuente es actualmente binario y dependiente de la arquitectura. Esto significa que tiene que generar los ficheros de las fuentes en el mismo tipo de CPU que la que tiene la máquina que está ejecutando PHP.

Tabla 1. Formato del fichero de fuentes

Posición en bytes	tipo de datos C	Descripción
byte 0-3	int	número de caracteres en la fuente
byte 4-7	int	valor del primer carácter de la fuente (normalmente 32 para el espacio)
byte 8-11	int	Anchura en pixels de cada carácter
byte 12-15	int	Altura en pixels de cada carácter
byte 16-	char	matriz con los datos del carácter, un byte por pixel en cada carácter, haciendo un total de (número caracteres* altura*anchura) bytes.

Vea también **ImageFontWidth()** y **ImageFontHeight()**.

ImagePolygon (PHP 3, PHP 4)

Dibuja un polígono

```
int imagepolygon (int im, array points, int num_points, int col)
```

ImagePolygon crea un polígono en la imagen id. points es un array PHP contenido los vértices del polígono. de la siguiente forma points[0] = x0, points1 = y0, points[2] = x1, points[3] = y1, etc. num_points es el número total de vértices.

Vea también **imagecreate()**.

ImagePSBBox (PHP 3>= 3.0.9, PHP 4 >= 4.0RC1)

Devuelve el borde que rodea un rectángulo de texto usando fuentes PostScript Type1

```
array imagepsbbox (string text, int font, int size, int space, int width, float angle)
```

size representa pixels.

space permite cambiar el valor por defecto de un espacio en una fuentes. Este valor es añadido al valor normal y puede ser negativo.

tightness permite controlar la cantidad de espacio en blanco entre caracteres. Este valor se añade a la anchura normal del carácter y puede ser negativo.

angle viene dado en grados.

Los parámetros *space* y *tightness* vienen expresados en unidades de espacio de caracteres, donde una unidad es 1/1000 el borde de una M.

Los parámetros *space*, *tightness* y *angle* son opcionales.

El borde es calculado usando la información disponible de las métricas del carácter, y desafortunadamente tiende a diferir ligeramente de los resultados obtenidos de digitalizar el texto. Si el ángulo es de 0 grados, puede esperar que el texto necesite un pixel más en cada dirección.

Esta función devuelve un array conteniendo los siguientes elementos:

0	coordenada x inferior izquierda
1	coordenada y inferior izquierda
2	coordenada x superior derecha
3	coordenada y superior derecha

Vea también **imagepstext()**.

ImagePSEncodeFont (PHP 3>= 3.0.9, PHP 4 >= 4.0RC1)

Cambia el vector de codificación de caracteres de una fuente

```
int imagepsencodefont (string encodingfile)
```

Carga un vector de codificación de caracteres desde un archivo y cambia el vector de codificación de las fuentes a él. Loads a character encoding vector from a file and changes the fonts encoding vector to it. En las fuentes PostScript normalmente faltan muchos caracteres por encima de 127, seguramente quiera cambiar esto si emplea un idioma distinto del inglés. El formato exacto de este archivo está descrito en la documentación de T1libs. T1lib viene con dos archivos listos para usar, IsoLatin1.enc y IsoLatin2.enc.

Si se encuentra usando esta función todo el tiempo, una forma mucho mejor de definir la codificación es establecer ps.default_encoding en el [archivo de configuración](#) para que apunte al archivo de codificación correcto y todas las fuentes que cargue tendrán de forma automática la codificación correcta.

ImagePSFreeFont (PHP 3>= 3.0.9, PHP 4 >= 4.0RC1)

Libera la memoria usada por un fuente PostScript Type 1

```
void imagepsfreefont (int fontindex)
```

Vea también **imagepsloadfont()**.

ImagePSLoadFont (PHP 3>= 3.0.9, PHP 4 >= 4.0RC1)

Carga una fuente PostScript Type 1 desde un fichero

```
int imagepsloadfont (string filename)
```

En el caso de que todo vaya bien, tiene que devolver un índice de fuente correcto que puede ser usado para futuras operaciones. En caso contrario la función devuelve false e imprime un mensaje describiendo donde ha fallado

Vea también **imagepsfreefont()**.

ImagePSText (PHP 3>= 3.0.9, PHP 4 >= 4.0RC1)

Para dibujar una cadena de texto sobre una imagen usando fuentes PostScript Type1

```
array imagepstext (int image, string text, int font, int size, int foreground, int background, int x, int y [, int space [, int tightness [, float angle [, int antialias_steps]]]])
```

size viene expresado en pixels.

foreground es el color en el cual el texto será pintado. *background* es el color en el que el texto tratará de resaltar con antialiasing. Los pixels con el color *background* no se pintan, de forma que la imagen de fondo no necesita ser de un color sólido.

Las coordenadas dadas por *x*, *y* definirán el origen (o punto de referencia) del primer carácter (la esquina superior izquierda del carácter). Esto es diferente de la función **ImageString()**, donde *x*, *y* definen la esquina superior derecha del primer carácter. Consulte la documentación de PostScript sobre fuentes y su sistema de medidas si tiene algún problema entendiendo como funcion.

space permite cambiar el valor por defecto de un espacio en la fuente. Esta cantidad es sumada al valor normal y puede ser negativa.

tightness permite controlar la cantidad de espacio en blanco entre caracteres. Esta cantidad es sumada al valor normal y puede ser negativa.

angle viene en grados.

antialias_steps permite controlar el número de colores usados para el antialiasing del texto. Los valores permitidos son 4 y 16. El valor superior está recomendado para textos con tamaños inferiores a 20, donde el efecto en la calidad del texto es bastante visible. Con tamaños superiores use 4. Hace un menor uso de cálculo.

Parameters *space* y *tightness* están expresados en unidades de espacio de caracteres, donde 1 unidad es 1/1000 de una M mayúscula.

Los parámetros *space*, *tightness*, *angle* y *antialias* son opcionales.

Esta función devuelve una matriz conteniendo los siguientes elementos:

0	coordenada x inferior izquierda
1	coordenada y inferior izquierda
2	coordenada x superior derecha
3	coordenada y superior derecha

Vea también **imagepsbbox()**.

ImageRectangle (PHP 3, PHP 4)

Dibuja un rectángulo

```
int imagerectangle (int im, int x1, int y1, int x2, int y2, int col)
```

ImageRectangle crea un rectángulo de color col en la imagen im comenzando en la coordenada superior izquierda x1,y1 y finalizando en la coordenada inferior derecha x2,y2. 0,0 es la esquina superior izquierda de la imagen.

ImageSetPixel (PHP 3, PHP 4)

Dibuja un pixel

```
int imagesetpixel (int im, int x, int y, int col)
```

ImageSetPixel dibuja un pixel x,y (arriba izquierda 0,0) en la imagen im con color col.

Vea también **imagecreate()** y **imagecolorallocate()**.

ImageString (PHP 3, PHP 4)

Dibuja una cadena de texto horizontalmente

```
int imagestring (int im, int font, int x, int y, string s, int col)
```

ImageString dibuja la cadena s en la imagen identificada por im en las coordenadas x,y (arriba izquierda es 0,0) en el color col. Si la fuente es 1, 2, 3, 4 o 5, se emplea una fuente interna.

Vea también **imageloadfont()**.

ImageStringUp (PHP 3, PHP 4)

Dibuja una cadena de texto verticalmente

```
int imagestringup (int im, int font, int x, int y, string s, int col)
```

ImageStringUp dibuja la cadena s verticalmente en la imagen identificada por im en las coordenadas x,y (arriba izquierda es 0,0) en el color col. Si la fuente es 1, 2, 3, 4 o 5, se usa una fuente interna.

Vea también **imageloadfont()**.

ImageSX (PHP 3, PHP 4)

Obtiene la anchura de la imagen

```
int imagesx (int im)
```

ImageSX devuelva la anchura de la imagen identificado por im.

Vea también **imagecreate()** y **imagesy()**.

ImageSY (PHP 3, PHP 4)

Obtiene la altura de la imagen

```
int imagesy (int im)
```

ImageSY devuelve la altura de la imagen identificada por *im*.

Vea también **imagecreate()** y **imagesx()**.

ImageTTFBBox (PHP 3>= 3.0.1, PHP 4)

Devuelve un caja que rodea al texto usando fuentes TypeType

```
array ImageTTFBBox (int size, int angle, string fontfile, string text)
```

Esta función calcula y devuelve un rectángulo en pixels que encierra un texto con TrueType.

text

La cadena que ha de ser medida.

size

El tamaño de la fuente.

fontfile

El nombre del archivo de fuente TrueType. (Puede ser también una URL.)

angle

Ángulo en grados en el *text* que va a ser medido.

ImageTTFBBox() devuelve una matriz con 8 elementos representando cuatro puntos que hacen una caja rodeando al texto:

0	esquina inferior izquierda, posición X
1	esquina inferior izquierda, posición Y
2	esquina inferior derecha, posición X
3	esquina inferior derecha, posición Y
4	esquina superior derecha, posición X
5	esquina superior derecha, posición Y
6	esquina superior izquierda, posición X
7	esquina superior izquierda, posición Y

Los puntos son relativos a *text* a pesar del ángulo, de forma que "superior izquierda" significa la esquina superior izquierda viendo el texto horizontalmente.

Esta función requiere tanto la librería GD como la librería Freetype.

Vea también **ImageTTFFText()**.

ImageTTFFText (PHP 3, PHP 4)

Escribe texto en la imagen usando fuentes TrueType

```
array ImageTTFTText (int im, int size, int angle, int x, int y, int col, string fontfile,  
string text)
```

ImageTTFTText escribe la cadena *text* en la imagen identificada por *im*, comenzando en las coordenadas *x,y* (arriba izquierda es 0,0), con un ángulo de *angle* en el color *col*, usando el ficheor de fuente TrueType identificado por *fontfile*.

Las coordenadas dadas por *x,y* definirán el punto base del primer carácter. (la esquina inferior izquierda del carácter). Esto es diferente de la función **ImageString()**, donde *x,y* definen la esquina superior derecha del primer carácter.

El *angle* viene dado en grados, donde 0 grados representa el texto de izquierda a derecha (dirección las 3 en punto), y valores superiores representan una rotación en el sentido de las agujas de un reloj. (ej. un valor de 90 representaría un texto que fuese de abajo hacia arriba).

fontfile es la ruta hacia la fuente TrueType que desee usar.

text es la cadena de texto que puede incluir secuencias de caracteres UTF-8 (de la forma: {) para acceder a caracteres de la fuente más allá de los primeros 255.

col es el índice de color. El uso de un índice de color negativo tiene el efecto de desactivar el antialiasing.

ImageTTFTText() devuelve una matriz con 8 elementos representando cuatro puntos que hace una caja que cubre el texto. El orden de los puntos en arriba izquierda, arriba derecha, abajo derecha, abajo izquierda. Los puntos son relativos al texto a pesar del ángulo, por lo que "arriba izquierda" significa en la esquina superior izquierda cuando ve el texto horizontalmente.

Este script de ejemplo producirá un GIF negro de 400x30 pixels, con las palabras "Testing..." en blanco con la fuente Arial.

Ejemplo 1. **ImageTTFTText**

```
<?php  
Header("Content-type: image/gif");  
$im = imagecreate(400,30);  
$black = ImageColorAllocate($im, 0,0,0);  
$white = ImageColorAllocate($im, 255,255,255);  
ImageTTFTText($im, 20, 0, 10, 20, $white, "/path/arial.ttf", "Testing... Omega: &#937;");  
ImageGif($im);  
ImageDestroy($im);  
?>
```

Esta función requiere la librería GD y la librería FreeType (<http://www.freetype.org/>).

Vea también **ImageTTFBBox()**.

XXXII. Funciones IMAP

Para hacer funcionar estas funciones, debe compilar PHP con `-with-imap`. Esto requiere que la librería c-client esté instalada. Obtenga la última versión de `ftp://ftp.cac.washington.edu/imap/` y compílela. Luego copie `c-client/c-client.a` al directorio `/usr/local/lib` o a cualquier otro directorio de su LINK path y copie `c-client/rfc822.h`, `mail.h` y `linkage.h` al directorio `/usr/local/include` o a cualquier otro de su INCLUDE path.

Decir que estas funciones no están limitadas al protocolo IMAP, a pesar de sus nombres. La librería subyacente c-client también soporta NNTP, POP3 y métodos de acceso local a buzones de correo. Vea `imap_open()` para una mayor información.

imap_append (PHP 3, PHP 4)

Agrega una cadena de mensaje al buzón especificado

```
int imap_append (int imap_stream, string mbox, string message, string flags)
```

Devuelve true si no hay error y false en caso contrario.

imap_append() agrega una cadena de mensaje al buzón especificado *mbox*. Si se especifica el parámetro *flags*, escribe las opciones o condiciones establecidas en el parámetro *flags* al buzón.

Cuando conecte con el servidor Cyrus IMAP, debe usar "\r\n" como finalizador de linea en vez de "\n" o la operación fallará.

imap_base64 (PHP 3, PHP 4)

Decodifica texto codificado en BASE64

```
string imap_base64 (string text)
```

imap_base64() decodifica texto codificado en BASE-64. El mensaje decodificado es devuelto como una cadena.

imap_body (PHP 3, PHP 4)

Lee el cuerpo del mensaje

```
string imap_body (int imap_stream, int msg_number, int flags)
```

imap_body() devuelve el cuerpo del mensaje, numerado *msg_number* del buzón actual. Los *flags* opcionales son una máscara de bit con una o mas de las siguientes:

- FT_UID - El msgno es un UID
- FT_PEEK - No activar \Seen flag si no está ya activa
- FT_INTERNAL - La cadena devuelta está en formato interno, no canoniza a CRLF.

imap_check (PHP 3, PHP 4)

Comprueba el estado del buzón actual

```
object imap_check (int imap_stream)
```

Devuelve información acerca del buzón actual. Devuelve FALSE si falla.

La función **imap_check()** comprueba el estado del buzón actual en el servidor y devuelve la información en un objeto con las siguientes propiedades.

Date : fecha del mensaje

Driver : controlador

Mailbox : nombre del buzón

Nmsgs : número de mensajes

Recent : número de mensajes recientes

imap_close (PHP 3, PHP 4)

Cierra una sesión IMAP

```
int imap_close (int imap_stream, int flags)
```

Cierra una sesión imap. Toma un parámetro *flag* opcional, CL_EXPUNGE, el cual purgará el buzón de forma transparente antes de cerrarla.

imap_createmailbox (PHP 3, PHP 4)

Crea un buzón nuevo

```
int imap_createmailbox (int imap_stream, string mbox)
```

imap_createmailbox() crea un buzón nuevo especificado por *mbox* (ver **imap_open()** para el formato del parámetro *mbox*).

Devuelve true si no hay error y false en caso contrario.

Ver También **imap_renamemailbox()** y **imap_deletemailbox()**.

imap_delete (PHP 3, PHP 4)

Marca un mensaje para ser borrado en el buzón actual

```
int imap_delete (int imap_stream, int msg_number)
```

Devuelve true.

La función **imap_delete()** marca el mensaje referenciado por *msg_number* para su eliminación. El borrado físico de los mensajes es realizado por **imap_expunge()**.

imap_deletemailbox (PHP 3, PHP 4)

Elimina un buzón

```
int imap_deletemailbox (int imap_stream, string mbox)
```

imap_deletemailbox() elimina el buzón especificado (ver **imap_open()** para el formato del *mbox*).

Devuelve true si no hay error y false en caso contrario.

Ver También **imap_createmailbox()** y **imap_reanmemailbox()**.

imap_expunge (PHP 3, PHP 4)

Elimina todos los mensajes marcados como borrados

```
int imap_expunge (int imap_stream)
```

imap_expunge() elimina todos los mensajes marcados por la función **imap_delete()**.

Devuleve true.

imap_fetchbody (PHP 3, PHP 4)

Localiza una sección particular en el cuerpo del mensaje

```
string imap_fetchbody (int imap_stream, int msg_number, string part_number, flags flags)
```

Esta función busca una sección particular en el cuerpo de los mensajes especificados, como una cadena de texto y devuleve esa cadena. La especificación de la sección es una cadena de enteros delimitados por comas,los cuales indexan las partes del cuerpo como indica la especificación IMAP4. Partes del cuerpo no son decodificadas por esta función.

Las opciones para **imap_fetchbody ()** son una máscara de bit con una o más de las siguientes

- FT_UID - El msgono es un UID
- FT_PEEK - No activar \Seen flag si no esta ya activa
- FT_INTERNAL - La cadena devuelta está en formato "interno", sin ningún intento por canonizar CRLF

imap_fetchstructure (PHP 3, PHP 4)

Lee la estructuta de un mensaje concreto

```
object imap_fetchstructure (int imap_stream, int msg_number [, int flags])
```

Esta función busca toda la información estructurada en el mensaje especificado. El parámetro opcional *flags* sólo tiene una opcion, *FT_UID*, la cual indica a la función que trate el argumento *msg_number* como un *UID*. El objeto devuelto incluye el sobre, la fecha interna, el tamaño, flags y la estructura del cuerpo con un objeto similar por cada mime adjunto al mensaje. La estructura de los objetos devueltos es como sigue:

Tabla 1. Objetos Devueltos para imap_fetchstructure()

type	Tipo primario del cuerpo
encoding	Body transfer encoding
ifsubtype	True si hay una cadena de subtipo
subtype	MIME subtype
ifDescripción	True si hay una cadena de Descripción
Description	Conenido de la cadena de Descripción
ifid	True si hay una cadena de identificación

id	Cadena de Identificación
lines	Número de lineas
bytes	Número de bytes
ifdisposition	True si hay una cadena de configuración
disposition	Cadena de configuración
ifdparameters	True si el array dparameters existe
dparameters a	Array de parametro de configuración
ifparameters	True si el array de parámetros existe
parameters b	MIME parameters array
parts c	Array de objetos describiendo cada parte del mensaje

Notas: a. dparameters es un array de objetos donde cada objeto tiene un "atributo" y una propiedad "valor". b. parameter es un

Tabla 2. Tipo primario del cuerpo

0	texto
1	multiparte
2	mensaje
3	aplicación
4	audio
5	imagen
6	video
7	otro

Tabla 3. Codificacion para tranferencia

0	7BIT
1	8BIT
2	BINARY
3	BASE64
4	QUOTED-PRINTABLE
5	OTRO

imap_header (PHP 3, PHP 4)

Lee la cabecera del mensaje

```
object imap_header (int imap_stream, int msg_number [, int fromlength [, int subjectlength [, string defaulthost]]])
```

Esta función devuelve un objeto con varios elementos de la cabecera.

remail, date, Date, subject, Subject, in_reply_to, message_id,
newsgroups, followup_to, references

message flags:

- Recent - 'R' si es reciente y ha sido leido,
- 'N' si es reciente y no ha sido leido,
- '' si no es reciente
- Unseen - 'U' si no ha sido leido Y no es reciente,
- '' si ha sido leido O no y es reciente
- Answered -'A' si ha sido contestado,
- '' si no ha sido contestado
- Deleted - 'D' si ha sido borrado,
- '' si no ha sido borrado
- Draft - 'X' if draft,
- '' if not draft
- Flagged - 'F' si esta if flagged,
- '' if not flagged

OBSERVE que el comportamiento Recent/Unseen es un poco extraño. Si quiere conocer si un mensaje es Unseen, debe comprobarlo asi

Unseen == 'U' || Recent == 'N'

toaddress (la linea to: al completo, hasta 1024 caracteres)

to[] (devuelve un array de objetos a partir de la linea To, conteniendo:)

- personal
- adl
- mailbox
- host

fromaddress (la linea from: al completo, hasta 1024 caracteres)

from[] (devuelve un array de objetos a partir de la linea From, conteniendo:)

- personal
- adl
- mailbox
- host

ccaddress (la linea cc: al completo, hasta 1024 caracteres)

cc[] (devuelve un array de objetos a partir de la linea Cc:, conteniendo:)

- personal
- adl
- mailbox
- host

bccaddress (la linea bcc al completo, hasta 1024 caracteres)

bcc[] (devuelve un array de objetos a partir de la linea Bcc, conteniendo:)

- personal
- adl
- mailbox
- host

reply_toaddress (la linea reply_to: al completo, hasta 1024 caracteres)

reply_to[] (devuelve un array de objetos a partir de la linea Reply_to,
conteniendo:)

personal
adl
mailbox
host

senderaddress (la linea sender: al completo, hasta 1024 caracteres)
sender[] (devuelve un array de objetos a partir de la linea sender, conteniendo:)

personal
adl
mailbox
host

return_path (la linea return-path: al completo, hasta 1024 caracteres)
return_path[] (devuelve un array de objetos a partir de la linea return_path,
conteniendo:)

personal
adl
mailbox
host

update (fecha del mensaje en formato unix)

fetchfrom (la linea from formateada hasta ajustarse a los caracteres
indicados en *fromlength*)

fetchsubject (la linea subject formateada hasta ajustarse a los caracteres
indicados en *subjectlength*)

imap_headers (PHP 3, PHP 4)

Returns headers for all messages in a mailbox

```
array imap_headers (int imap_stream)
```

Devuelve un array de cadenas formateadas con informacion de la cabecera. Un elemento por mensaje de correo.

imap_listmailbox (PHP 3, PHP 4)

Lee la lista de buzones

```
array imap_listmailbox (int imap_stream, string ref, string pat)
```

Devuelve un array que contiene los nombres de los buzones.

imap_getmailboxes (PHP 3>= 3.0.12, PHP 4 >= 4.0b4)

Lee la lista de buzones, devolviendo información detallada de cada uno

```
array imap_getmailboxes (int imap_stream, string ref, string pat)
```

Devuelve un array de objetos coneniendo información del buzón. Cada objeto tiene los atributos *name*, especificando el nombre completo del buzón; *delimiter*, que es el delimitador jerárquico para la parte de la jerarquía dónde está este buzón; y *attributes*. *Attributes* es una máscara de bits contra la que se puede probar:

- LATT_NOINFERIORS - Este buzón no tiene "hijos"(No ha buzones por debajo de él)
- LATT_NOSELECT - Esto es sólo un contenedor, no un buzón - No puede abrirlo.
- LATT_MARKED - Este buzón está marcado. Unicamente usado por UW-IMAPD.
- LATT_UNMARKED - Este buzón no está marcado. Unicamente usado por UW-IMAPD.

ref normalmente debería ser solo el servidor IMAP, de la forma: {imap_server:imap_port}, y *pattern* especifica, dónde en la estructura jerárquica del buzón, para comenzar a buscar. Si quiere todo los buzones, pase el parámetro *pattern* como una cadena vacía.

Hay dos caracteres especiales que puede pasar como parte del parámetro *pattern*: '*' and '%'. '*' significa que devuelva todos los buzones. Si pasa el parámetro *pattern* como '*', obtendrá una lista con la jerarquía completa del buzón. '%' significa que devuelva sólo el nivel actual. Pasar '%' en el parámetro *pattern* devolverá sólo el nivel más alto de los buzones; '~/mail/%' en UW_IMAPD devolverá cada buzón del directorio ~/mail, pero ninguno de los subdirectorios de ese directorio.

imap_listsubscribed (PHP 3, PHP 4)

Lista todos los buzones subscriptos

```
array imap_listsubscribed (int imap_stream, string ref, string pattern)
```

Devuelve un array de todos los buzones que usted tiene subscriptos. Los parámetros *ref* y *pattern* especifican la localización desde donde comenzará a buscar y el patrón que el nombre del buzón debe encontrar.

imap_getsubscribed (PHP 3>= 3.0.12, PHP 4 >= 4.0b4)

Lista todos los buzones subscriptos

```
array imap_getsubscribed (int imap_stream, string ref, string pattern)
```

Esta función es idéntica a **imap_getmailboxes()**, excepto que esta sólo devuelve los buzones a los que está subscripto el usuario.

imap_mail_copy (PHP 3, PHP 4)

Copia los mensajes especificados a un buzón

```
int imap_mail_copy (int imap_stream, string msglist, string mbox, int flags)
```

Devuelve true si no hay error y false en caso contrario.

Copia los mensajes especificados por *msglist* a un buzón especificado. *msglist* es un rango no números de mensajes.

Flags es una máscara de bit de uno o más

- CP_UID - los números de secuencia contienen UIDS
- CP_MOVE - Elimina los mensajes del buzon actual despues de copiarlos

imap_mail_move (PHP 3, PHP 4)

Mueve los mensajes especificados a un buzón

```
int imap_mail_move (int imap_stream, string msglist, string mbox)
```

Mueve los mensajes especificados por *msglist* al buzón especicifado. *msglist* es un rango no números de mensajes.

Devuelve true si no hay error y false en caso contrario.

imap_num_msg (PHP 3, PHP 4)

Informa del número de mensajes en el buzón actual

```
int imap_num_msg (int imap_stream)
```

Devuelve el número de mensajes en el buzón actual.

imap_num_recent (PHP 3, PHP 4)

Informa el número de mensajes recientes en el buzón actual

```
int imap_num_recent (int imap_stream)
```

Devuleve el número de mensajes recientes en el buzón actual.

imap_open (PHP 3, PHP 4)

Abre una sesión IMAP

```
int imap_open (string mailbox, string username, string password, int flags)
```

Devuleve la sesion IMAP si no hay error y false en caso contrario. Esta función también puede ser usada para abrir sesiones con servidores POP3 y NNTP. Para conectarse a un servidor IMAP escuchando por el puerto 143 en una máquina local, haga lo siguiente:

```
$mbox = imap_open("{localhost:143}INBOX", "user_id", "password");
```

Para conectarse a un servidor POP3 escuchando por el puerto 110, use:

```
$mbox = imap_open("{localhost/pop3:110}INBOX", "user_id", "password");
```

Para conectarse a un servidor NNTP escuchando por el puerto 119, use:

```
$nntp = imap_open("{localhost/nntp:119}comp.test", "", "");
```

Para conectarse a un servidor remoto sustituya "localhost", por el nombre o dirección IP del servidor al cual quiere conectarse.

Las opciones son una máscara de bit con una o más de los siguientes:

- OP_READONLY - Abre el buzón en modo de sólo lectura
- OP_ANONYMOUS - No usa o actualiza un .newsr para las noticias
- OP_HALFOPEN - Para nombres IMAP y NNTP, abre una conexión pero no abre un buzón
- CL_EXPUNGE - Purga automáticamente el buzón antes de cerrar la sesión

imap_ping (PHP 3, PHP 4)

Comprueba si la sesión IMAP está aún activa

```
int imap_ping (int imap_stream)
```

Devuelve true si la sesión está activa, false en caso contrario.

La función **imap_ping()** pings the stream to see if it is still active. Esto puede descubrir que hay correo nuevo; este es el método preferido para hacer una comprobación periódica del buzón, así como para mantener activa sesiones en servidores que tienen inactivity timeout.

imap_renamemailbox (PHP 3, PHP 4)

Renombra un buzón

```
int imap_renamemailbox (int imap_stream, string old_mbox, string new_mbox)
```

Esta función renombra un buzón (ver **imap_open()** para el formato del parámetro *mbox*).

Devuelve true si no hay error y false en caso contrario.

Ver También **imap_createmailbox()** and **imap_deletemailbox()**.

imap_reopen (PHP 3, PHP 4)

Reabre una sesión IMAP a un nuevo buzón

```
int imap_reopen (string imap_stream, string mailbox [, string flags])
```

Devuelve true si no hay error y false en caso contrario.

Esta función reabre la sesión especificada con un nuevo buzón.

Las opciones son máscaras de bit con una o más de las siguientes:

- OP_READONLY - Abre el buzón en modo de sólo lectura
- OP_ANONYMOUS - No usa o actualiza .newsrc para noticias
- OP_HALFOPEN - Para nombres IMAP y NNTP, abre una conexión pero no abre el buzón.
- CL_EXPUNGE - Expurga automáticamente el buzón antes de cerrar la sesión

imap_subscribe (PHP 3, PHP 4)

Subscribe to a mailbox

```
int imap_subscribe (int imap_stream, string mbox)
```

Da de alta un nuevo buzón.

Devuelve true si no hay error y false en caso contrario.

imap_undelete (PHP 3, PHP 4)

Desmarca los mensajes que están marcados como borrados

```
int imap_undelete (int imap_stream, int msg_number)
```

Esta función elimina la marca de borrado de un mensaje específico, puesta por la función **imap_delete()**.

Devuelve true si no hay error y false en caso contrario.

imap_unsubscribe (PHP 3, PHP 4)

Unsubscribe from a mailbox

```
int imap_unsubscribe (int imap_stream, string mbox)
```

Da de baja el buzón especificado.

Devuelve true si no hay error y false en caso contrario.

imap_qprint (PHP 3, PHP 4)

Convierte una cadena quoted-printable a una cadena de 8 bit

```
string imap_qprint (string string)
```

Convierte una cadena quoted-printable a una cadena de 8 bit
Devuelve una cadena de 8 bit (binary)

imap_8bit (PHP 3, PHP 4)

Convierte una cadena de 8bit a una cadena quoted-printable

```
string imap_8bit (string string)
```

Convierte una cadena de 8bit a una cadena quoted-printable.
Devuelve una cadena quoted-printable

imap_binary (PHP 3>= 3.0.2, PHP 4)

Convierte una cadena de 8bit a una cadena base64

```
string imap_binary (string string)
```

Convierte una cadena de 8bit a una cadena base64.
Devuleve una cadena base64.

imap_scanmailbox (PHP 3, PHP 4)

Lee la lista de buzones y toma una cadena para buscar en el texto del buzón

```
array imap_scanmailbox (int imap_stream, string string)
```

Devuelve un array que contiene los nombres de los buzones que tienen el parámetro *string* en el texto del buzón.

imap_mailboxmsginfo (PHP 3>= 3.0.2, PHP 4)

Obtiene información acerca del buzón actual

```
object imap_mailboxmsginfo (int imap_stream)
```

Devuelve información acerca del buzón actual. Devuelve FALSE en caso de fallo.

La función **imap_mailboxmsginfo()** comprueba el estado del buzón actual en el servidor y devuelve la información en un objeto con las siguientes propiedades.

Date : fecha del mensaje

Driver : driver

Mailbox : nombre del buzón

Nmsgs : número de mensajes
 Recent : número de los mensajes recientes
 Unread : número de los mensajes no leidos
 Size : tamaño del buzón

imap_rfc822_write_address (PHP 3>= 3.0.2, PHP 4)

Devuelve una dirección de correo correctamente formateada dado el buzón, host, e información personal.

```
string imap_rfc822_write_address (string mailbox, string host, string personal)
```

Devuelve una dirección de correo correctamente formateada, dado el buzón, host, e información personal.

imap_rfc822_parse_adrlist (PHP 3>= 3.0.2, PHP 4)

Examina la cadena dirección

```
string imap_rfc822_parse_adrlist (string address, string default_host)
```

Esta función examina la cadena dirección y para cada dirección, devuelve un array de objetos. Los 4 objetos son:

mailbox - el nombre del buzón (username)
 host - el nombre del ordenador
 personal - el nombre personal
 adl - ruta del dominio

imap_setflag_full (PHP 3>= 3.0.3, PHP 4)

Activa flags en los mensajes

```
string imap_setflag_full (int stream, string sequence, string flag, string options)
```

Esta función añade el flag especificado al conjunto de flags activos para los mensajes en la secuencia especificada.

Los flags que puede seleccionar son "\Seen", "\Answered", "\Flagged", "\Deleted", "\Draft", y "\Recent"(definidos en el RFC2060)

Las opciones son una máscara de bit con uno o más de los siguientes:

ST_UID El argumento sequence contiene UIDs en vez de números secuenciales

imap_clearflag_full (PHP 3>= 3.0.3, PHP 4)

Limpia los flags de los mensajes

```
string imap_clearflag_full (int stream, string sequence, string flag, string options)
```

Esta funcion elimina el flag especificado del conjunto de flags activos para los mensajes en la secuencia especificada.

Las opciones son una máscara de bit con uno o más de los siguientes:

ST_UID El argumento sequence contiene UIDs en vez de
números secuenciales

imap_sort (PHP 3>= 3.0.3, PHP 4)

Ordena un array de cabeceras de mensajes

```
string imap_sort (int stream, int criteria, int reverse, int options)
```

Devuelve un array de números de mensajes ordenados por los parametros dados

Rev es 1 para una ordenación inversa.

Criteria puede ser uno (y sólo uno) de los siguientes:

SORTDATE Fecha del mensaje
SORTARRIVAL Fecha de llegada
SORTFROM mailbox in first From address
SORTSUBJECT Asunto del mensaje
SORTTO mailbox in first To address
SORTCC mailbox in first cc address
SORTSIZE tamaño del mensaje en bytes

Las opciones son una máscara de bit con uno o más de los siguientes:

SE_UID Devuelve UIDs en vez de números secuenciales
SE_NOPREFETCH No preselecciona los mensajes buscados.

imap_fetchheader (PHP 3>= 3.0.3, PHP 4)

Devuelve la cabecera del mensaje

```
stringimap_fetchheader (int imap_stream, int msgno, int flags)
```

Esta función localiza el formato de la cabecera RFC 822 del mensaje especificado como una cadena de texto y devuelve esa cadena de texto.

The options are:

FT_UID El argumento msgno es un UID
 FT_INTERNAL La cadena devuelta esta en formato "interno",
 sin ningún intento de canonizar CRLF

FT_PREFETCHTEXT The RFC822.TEXT should be pre-fetched at the same time. Esto evita un extra RTT en una conexión IMAP si se desea un mensaje completo de texto (e.g. en una operación de "guardar a un fichero local")

imap_uid (PHP 3>= 3.0.3, PHP 4)

Esta función devuelve el UID del número de secuencia del mensaje dado

```
int imap_uid (int imap_stream, int msgno)
```

Esta función devuelve el UID del número de secuencia del mensaje dado. Esta función es la inversa a **imap_msgno()**.

imap_msgno (PHP 3>= 3.0.3, PHP 4)

Esta función devuelve el número de secuencia del mensaje para el UID dado.

```
int imap_msgno (int imap_stream, int uid)
```

Esta función devuelve el número de secuencia del mensaje para el UID dado. Esta función es la inversa a **imap_uid()**.

imap_search (PHP 3>= 3.0.12, PHP 4 >= 4.0b4)

Esta función devuelve un array de mensajes que coinciden con el criterio de búsqueda dado.

```
array imap_search (int imap_stream, string criteria, int flags)
```

Esta función realiza una búsqueda en el buzón actualmente abierto indicado por *imap_stream*. *criteria* es una cadena, delimitada por espacios, en la cual las siguientes palabras claves son permitidas. Cualquier argumento múltiple (ej. FROM "joey smith") debe estar entre comillas.

- ALL - devuelve todos los mensajes que coinciden con el resto del criterio
- ANSWERED - busca mensajes con el flag \ANSWERED activado
- BCC "string"- busca mensajes con "cadena"en el campo Bcc:
- BEFORE "date"- busca mensajes con Date: antes de "date"
- BODY "string"- busca mensajes con "cadena"en el cuerpo del mensaje

- CC "string"- busca mensajes con "cadena"en el campo Cc:
- DELETED - busca mensajes eliminados
- FLAGGED - busca mensajes con el flag \FLAGGED (sometimes referred to as Important or Urgent) activado
- FROM "string"- busca mensajes con "cadena"en el campo From:
- KEYWORD "string"- busca mensajes con "cadena"como una palabra clave
- NEW - busca mensajes nuevos
- OLD - busca mensajes viejos
- ON "date"- busca mensajes con "date"igual a Date:
- RECENT - busca mensajes con el flag \RECENT activado
- SEEN - busca mensajes que han sido leidos (la opcion \SEEN activada)
- SINCE "date"- busca mensajes conwith Date: after "date"
- SUBJECT "string"- busca mensajes con "string"en el campo Subject:
- TEXT "string"- busca mensajes con el texto "string"
- TO "string"- busca mensajes con "string"en el campo To:
- UNANSWERED - busca mensajes que no han sido respondidos
- UNDELETED - busca mensajes que no han sido eliminados
- UNFLAGGED - busca mensajes que no estan flagged
- UNKEYWORD "string"- busca mensajes que no coinciden con la palabra clave "string"
- UNSEEN - busca mensajes que no han sido leidos aún

Por ejemplo, para buscar todos los mensajes no contestados enviados por Mamá, usaría: "UNANSWERED FROM mamá". La búsqueda parece ser no sensitiva. Esta lista de criterios está tomada del código fuente del UW c-client y puede que este incompleta o sea inexacta.

Valores validos para los flags son SE_UID, que provoca que el array devuelto contenga UIDs en vez de los numeros de secuencia de los mensajes

imap_last_error (PHP 3>= 3.0.12, PHP 4 >= 4.0b4)

Esta función devuelve el último error IMAP (si se produjo) que ocurrió durante la petición de esta página.

```
string imap_last_error (void)
```

Esta función devuelve el texto completo del último error IMAP que ocurrió en la pagina actual. La pila de errores The error stack is untouched; llamando despues a la función **imap_last_error()**, sin que se produzca un error, devolverá el mismo error.

ATENCIÀN: esta función no esta disponible aún en PHP4.

imap_errors (PHP 3>= 3.0.12, PHP 4 >= 4.0b4)

Esta función devuelve todos los errores IMAP (si hubo) que han ocurrido durante la petición de la página o desde que la pila de errores se inicializó.

```
array imap_errors (void)
```

Esta función devuelve un array de todos los mensajes de error IMAP generados desde la última llamada a **imap_errors()**, o el principio de la página. Cuando se llama a **imap_errors()**, la pila de errores se inicializa.

ATENCIaN: esta función no está disponible aún en PHP4.

imap_alerts (PHP 3>= 3.0.12, PHP 4 >= 4.0b4)

Esta función devuelve todos los mensajes de alerta IMAP (si hubo) que han ocurrido durante la petición de la pagina o desde que la pila de alertas fue inicializada.

```
array imap_alerts (void)
```

Esta función devuelve un array con todos los mensajes de alerta IMAP generados desde la última llamada a **imap_alerts()**, o el comienzo de la pagina. Cuando se llama a **imap_alerts()**, la pila de alerta es inicializada. La especificación IMAP requiere que estos mensajes sean pasados al usuario.

imap_status (PHP 3>= 3.0.4, PHP 4)

Esta función devuelve el información de estado de otro buzón distinto al actual.

```
object imap_status (int imap_stream, string mailbox, int options)
```

Esta función devuelve un objeto que contiene información de estado. Las opciones válidas son:

- SA_MESSAGES - activa status->messages con el número de mensajes en el buzón
- SA_RECENT - activa status->recent con el número de mensajes recientes en el buzón
- SA_UNSEEN - activa status->unseen con el número de mensajes no leidos (nuevos) en el buzón
- SA_UIDNEXT - activa status->uidnext con el próximo uid a usar en el buzón
- SA_UIDVALIDITY - activa status->uidvalidity con una constante que cambia cuando los uids del buzón ya no son válidos
- SA_ALL - activa todos los de arriba

status->flags contienen una máscara de bits la cual puede ser comprobada contra cualquiera de las propiedades de arriba.

XXXIII. Funciones para Informix

El conector para Informix Online (ODS) 7.x, SE 7.x y Universal Server (IUS) 9.x se encuentra implementado en "functions/ifx.ec" y "functions/php3_ifx.h". Para ODS 7.x está completado, con total soporte para columnas de tipo BYTE y TEXT. Para IUS 9.x está parcialmente finalizado: los tipos de datos nuevos están allí (en el IUS 9.x), pero la funcionalidad para SLOB y CLOB se encuentra bajo construcción todavía.

Notas de configuración:

Antes de ejecutar el guión (script) "configure", asegúrate que la variable "INFORMIXDIR" ha sido definida.

Si ejecutas "configure --with_informix=yes" entonces el guión de configuración detectará automáticamente las librerías y los directorios include. Puedes obviar esta detección definiendo las variables de entorno "IFX_LIBDIR", "IFX_LIBS" y "IFX_INCDIR". Definirás la variable de compilación condicional "HAVE_IFX_IUS" si la versión de Informix es 9.00 o superior.

Algunas notas sobre el uso de BLOBs (columnas de tipo TEXT y BYTE):

BLOBs son normalmente manipulados por enteros, los cuales representan identificadores de BLOB. Las consultas de selección devuelven un "blob id" para columnas de tipo BYTE y TEXT. Si eliges trabajar con los BLOBs en memoria (con: "ifx_blobinfile(0);") entonces puedes obtener el contenido con "string_var = ifx_get_blob(\$blob_id);". Si prefieres manipularlos en ficheros usa "ifx_blobinfile(1);" y "ifx_get_blob(\$blob_id);" devolverá el nombre del archivo. En este caso, utiliza las funciones habituales de entrada y salida de ficheros para obtener el contenido de los blob.

Para consultas de inserción y actualización debes crear estos identificadores de blob con "ifx_create_blob(..);". Entonces pondrás los identificadores de blob en un array y sustituirás en la cadena de la consulta las columnas de tipo blob por una interrogación (?). Para inserciones y actualizaciones eres responsable de definir el contenido de los blob con ifx_update_blob(...).

La conducta de columnas BLOB puede ser modificada mediante variables de configuración, las cuales pueden ser definidas en tiempo de ejecución mediante funciones.

variable de configuración: ifx.textasvarchar

variable de configuración: ifx.byteasvarchar

funciones en tiempo de ejecución:

ifx_textasvarchar(0): usa identificadores de blob para columnas de tipo TEXT en las consultas de selección

ifx_byteasvarchar(0): usa identificadores de blob para columnas de tipo BYTE en las consultas de selección

ifx_textasvarchar(1): devuelve columnas de tipo TEXT como si fueran de tipo VARCHAR, sin tener que usar identificadores de blob en las consultas de selección

ifx_byteasvarchar(1): devuelve columnas de tipo BYTE como si fueran de tipo VARCHAR, sin tener que usar identificadores de blob en las consultas de selección.

variable de configuración: ifx.blobinfile

función en tiempo de ejecución:

ifx_blobinfile_mode(0): devuelve columnas de tipo BYTE en memoria, el identificador de blob te permite obtener el contenido.

ifx_blobinfile_mode(1): devuelve columnas de tipo BYTE en un fichero, el identificador te permite saber el nombre de dicho archivo.

Si defines ifx_text/byteasvarchar a 1 entonces puedes usar columnas de tipo TEXT y BYTE en las consultas de selección como campos de tipo VARCHAR, pero teniendo en cuenta que tendrán un mayor tamaño que el habitual. Ya que en PHP todas las cadenas son posibles, esto permite datos binarios. De esta forma, se pueden manejar correctamente. La información devuelta puede contener cualquier cosa, tú eres responsable del contenido.

Si defines ifx_blobinfile a 1, utiliza el nombre del archivo devuelto por ifx_get_blob(..) para acceder a los contenidos del blobs. En este caso, ERES REPONSABLE DE ELIMINAR EL ARCHIVO TEMPORAL GENERADO POR INFORMIX

cuando accedas a los registros. Cada nueva fila obtenida creará un nuevo archivo temporal para cada columna de tipo BYTE.

El directorio donde se guardan los archivos temporales puede ser definido por la variable de entorno blobdir, por defecto es ".", es decir, el directorio actual. Así, `putenv(blobdir=tmpblob")`; definirá un directorio donde se localizarán todos los ficheros temporales y facilitará su borrado. Todos los nombres de los archivos comienzan por "blb".

Recortado (trimming) automático de datos de tipo "char" (SQLCHAR y SQLNCHAR):

Puede ser definido con la variable de configuración

`ifx.charasvarchar`: si se define a 1 eliminará automáticamente los espacios en blanco al final de la cadena.

Valores NULL:

La variable de configuración `ifx.nullformat` (y en tiempo de ejecución `ifx_nullformat()`) cuando sea definida a true devolverá columnas NULL como la cadena "NULL", si es definida a false entonces la cadena vacía. Esto permite distinguir entre columnas NULL y vacías.

ifx_connect (PHP 3>= 3.0.3, PHP 4)

Abre una conexión con un servidor Informix

```
int ifx_connect ([string database [, string userid [, string password] ]])
```

Si tuvo éxito, devuelve un identificador de conexión en otro caso FALSE.

ifx_connect() establece una conexión con un servidor INFORMIX. Todos los argumentos son opcionales, y si no se pasan, se toman los valores del [fichero de configuración](#) (ifx.default_host para el ordenador donde se encuentra el servidor (si no es definida, las librerías de Infomix usarán la variable de entorno INFORMIXSERVER), ifx.default_user para el usuario (*userid*), ifx.default_password para la contraseña (*password*) (ninguna, si no es definida).

Para una segunda llamada a **ifx_connect()** con los mismos argumentos, no se establecerá una nueva conexión, en vez de eso, el identificador de enlace de la conexión abierta será devuelto.

La conexión con el servidor será cerrada tan pronto como la ejecución del guión (script) finalice, a menos que anteriormente se haya llamando a **ifx_close()**.

Examina también **ifx_pconnect()**, y **ifx_close()**.

Ejemplo 1. Conexión a una base de datos Informix

```
$conn_id = ifx_pconnect ("mydb@ol_srvl", "imyself", "mypassword");
```

ifx_pconnect (PHP 3>= 3.0.3, PHP 4)

Abre una conexión permanente con Informix

```
int ifx_pconnect ([string database [, string userid [, string password] ]])
```

Devuelve un identificador positivo de enlace persistente si hubo conexión, o false si se produjo un error.

ifx_pconnect() actúa muy parecido a **ifx_connect()** con dos principales diferencias.

Esta función se comporta exactamente igual que **ifx_connect()** cuando PHP no es ejecutado como un módulo de Apache. La primera diferencia es cuando se conecta, la función intentará encontrar un enlace (persistente) que exista con el mismo servidor, usuario y contraseña. Si es hallado, el identificador del enlace será devuelto en vez de abrir una nueva conexión.

Segundo, la conexión al servidor no se cerrará cuando la ejecución del guión (script) finalice. En vez de esto, la conexión permanecerá abierta para usos futuros (**ifx_close()** no cerrará el enlace creado por **ifx_pconnect()**).

Este tipo de enlace es, por tanto, llamado 'persistente'

Examina también: **ifx_connect()**.

ifx_close (PHP 3>= 3.0.3, PHP 4)

Cierra una conexión con Informix

```
int ifx_close ([int link_identifier])
```

Devuelve: true siempre.

ifx_close() cierra un enlace a una base de datos Informix que esté asociado con el identificador de enlace (link_identifier). Si el identificador de enlace no es especificado, el último enlace abierto es asumido.

Observa que esto no es necesario habitualmente ya que las conexiones no permanentes son cerradas automáticamente al finalizar el guión (script).

ifx_close() no cerrará enlaces persistentes generados por **ifx_pconnect()**.

Examina también: **ifx_connect()**, y **ifx_pconnect()**.

Ejemplo 1. Cierre de una conexión a Informix

```
$conn_id = ifx_connect ("mydb@ol_srv", "itsme", "mypassword");
... algunas consultas y código ...
ifx_close($conn_id);
```

ifx_query (PHP 3>= 3.0.3, PHP 4)

Envía una consulta a Informix

```
int ifx_query (string query [, int link_identifier [, int cursor_type [, mixed blobidarray]]])
```

Devuelve un identificador positivo de resultado si tuvo éxito, false en otro caso.

Un entero (integer) "result_id" usado por otras funciones para obtener el resultado de la consulta. Es definido "affected_rows"(registros procesados) y se puede obtener mediante la función **ifx_affected_rows()**.

ifx_query() envía una consulta a la base de datos activa actualmente en el servidor, la cual está representada por el identificador de enlace especificado (link_identifier). Si el identificador no es definido, el último enlace abierto es asumido. Si el enlace no se encuentra abierto, **ifx_connect()** es llamado y utilizado.

Ejecuta una consulta (*query*) sobre una conexión (*link_identifier*). Un cursor es definido y abierto para las consultas de selección. El parámetro opcional tipo de cursor (*cursor_type*) te permite que sea un cursor de tipo "scroll"y/o "hold". Es una máscara y puede ser IFX_SCROLL, IFX_HOLD o ambos. Las consultas que no son de selección son ejecutadas inmediatamente.

Para cualquier tipo de consulta el número (estimado o real) de registros procesados es guardo y se puede obtener mediante **ifx_affected_rows()**.

Si tienes columnas BLOB (BYTE o TEXT) en una consulta de actualización, puedes añadir un parámetro *blobidarray* conteniendo los identificadores de blob y sustituir los valores de esas columnas por una "?"en el texto de la consulta.

Si el contenido de la columna de tipo TEXT (o BYTE) lo permite, también puedes usar "ifx_textasvarchar(1)"y "ifx_byteasvarchar(1)". Esto supone manejar columnas de tipo TEXT (o BYTE) como si fueran columnas normales de tipo VARCHAR (pero teniendo en cuenta que tendrán un mayor tamaño que el habitual), para consultas de selección y no necesitas preocuparte por los identificadores de blob.

La opción por defecto ifx_textasvarchar(0) o ifx_byteasvarchar(0) devuelve identificadores de blob (valores enteros) para las consultas de selección. Puedes obtener el contenido del blob como una cadena o un fichero con las funciones para blob (ver más adelante).

Examina también: **ifx_connect()**.

Ejemplo 1. Mostrar todos los registros de la tabla "orders"como una tabla html

```
ifx_textasvarchar(1);      // usa "modo texto" para blobs
$res_id = ifx_query("select * from orders", $conn_id);
if (! $res_id) {
```

```

printf("Can't select orders : %s\n<br>%s<br>\n", ifx_error());
ifx_errormsg();
die;
}
ifx_htmltbl_result($res_id, "border=\"1\"");
ifx_free_result($res_id);

```

Ejemplo 2. Inserta valores en la tabla "catalog"

```

// crea identificadores de blob para una columna de tipo byte y otra text
$textid = ifx_create_blob(0, 0, "Text column in memory");
$byteid = ifx_create_blob(1, 0, "Byte column in memory");
// almacena los identificadores de blob en un array llamado blobid
$blobidarray[] = $textid;
$blobidarray[] = $byteid;
// lanza la consulta
$query = "insert into catalog (stock_num, manu_code, ".
          "cat_descr,cat_picture) values(1,'HRO',?,?)";
$res_id = ifx_query($query, $conn_id, $blobidarray);
if (! $res_id) {
    ... error ...
}
// libera el resultado
ifx_free_result($res_id);

```

ifx_prepare (PHP 3>= 3.0.4, PHP 4)

Prepara una sentencia SQL para su ejecución

```
int ifx_prepare (string query, int conn_id [, int cursor_def, mixed blobidarray])
```

Devuelve un entero (integer) *result_id* para usarlo con **ifx_do()**. Es definido "affected_rows"(registros procesados) y se puede obtener mediante la función **ifx_affected_rows()**.

Prepara una consulta (*query*) sobre una conexión (*link_identifier*). Un cursor es definido y abierto para las consultas de selección. El parámetro opcional tipo de cursor (*cursor_type*) te permite que sea un cursor de tipo "scroll"y/o "hold". Es una máscara y puede ser IFX_SCROLL, IFX_HOLD o ambos.

Para cualquier tipo de consulta el número estimado de registros afectados (procesados) es guardado y puede ser obtenido mediante **ifx_affected_rows()**.

Si tienes columnas BLOB (BYTE o TEXT) en una consulta, puedes añadir un parámetro *blobidarray* conteniendo los identificadores de blob y sustituir los valores de esas columnas por una "?"en el texto de la consulta.

Si el contenido de la columna de tipo TEXT (o BYTE) lo permite, puedes también usar "ifx_textasvarchar(1)"y "ifx_byteasvarchar(1)". Esto supone manejar columnas de tipo TEXT (o BYTE) como si fueran columnas normales de tipo VARCHAR (pero teniendo en cuenta que tendrán un mayor tamaño que el habitual), para consultas de selección y no necesitas preocuparte por los identificadores de blob.

La opción por defecto ifx_textasvarchar(0) o ifx_byteasvarchar(0) devuelve identificadores de blob (valores enteros) para las consultas de selección. Puedes obtener el contenido del blob como una cadena o un fichero con las funciones para blob (ver más adelante).

Examina también: **ifx_do()**.

ifx_do (PHP 3>= 3.0.4, PHP 4)

Ejecuta una sentencia SQL preparada previamente

```
int ifx_do (int result_id)
```

Devuelve TRUE si se realizó, FALSE si hubo algún error.

Ejecuta una consulta preparada anteriormente o abre un cursor para ella.

No libera *result_id* si hubo un error.

También define el número real de registros procesados para consultas que no sean de selección y se puede obtener mediante **ifx_affected_rows()**.

Examina también: **ifx_prepare()** (hay un ejemplo).

ifx_error (PHP 3>= 3.0.3, PHP 4)

Devuelve el código de error de la última llamada a Informix

```
string ifx_error(void);
```

Los códigos de error de Informix (SQLSTATE & SQLCODE) son representados como se especifica a continuación:

x [SQLSTATE = aa bbb SQLCODE=cccc]

donde x = un espacio : no hubo error

E : hubo error

N : no hay más datos

W : aviso

? : no definido

Si el carácter "x"es cualquier otra cosa diferente a un espacio, SQLSTATE y SQLCODE describen el error con mayor detalle.

Examina el manual de Informix para el significado de SQLSTATE y SQLCODE.

Devuelve en una cadena un carácter describiendo el resultado de una sentencia y los valores SQLSTATE y SQLCODE asociados con la última sentencia SQL ejecutada. El formato de la cadena es "(char) [SQLSTATE=(dos dígitos) (tres dígitos) SQLCODE=(un dígitos)]". El primer carácter puede ser ' ' (un espacio) (no hubo error), 'W' (la sentencia provocó un aviso), 'E' (la consulta produjo un error) o 'N' (la sentencia no devolvió ningún dato).

Examina también: **ifx_errormsg()**

ifx_errormsg (PHP 3>= 3.0.4, PHP 4)

Devuelve el mensaje de error de la última llamada a Informix

```
string ifx_errormsg ([int errorcode])
```

Devuelve el mensaje de error asociado con el error más reciente de Informix. Si definimos el parámetro opcional "errorcode"(código de error), nos dará el mensaje de error correspondiente a ese código.

Examina también: **ifx_error()**

```
printf("%s\n<br>", ifx_errormsg(-201));
```

ifx_affected_rows (PHP 3>= 3.0.3, PHP 4)

Obtiene el número de registros procesados por una consulta

```
int ifx_affected_rows (int result_id)
```

result_id es un identificador válido del resultado de **ifx_query()** o **ifx_prepare()**.

Devuelve el número de filas procesadas por una consulta representada por un *result_id* (identificador de resultado).

Para inserciones, actualizaciones y borrados el número es exactamente los registros procesados (sqlerrd[2]). Para las consultas de selección es una estimación (sqlerrd[0]). No confíes en él.

Es útil llamarla después de ejecutar **ifx_prepare()** pues así podemos limitar las consultas a número razonable de registros.

Examina también: **ifx_num_rows()**

Ejemplo 1. Número de registros procesados por una consulta

```
$rid = ifx_prepare ("select * from emp where name like " . $name, $connid);
if (! $rid) {
    ... error ...
}
$rowcount = ifx_affected_rows ($rid);
if ($rowcount > 1000) {
    printf ("Too many rows in result set (%d)\n<br>", $rowcount); // Demasiados reg-
istros en el resultado
    die ("Please restrict your query<br>\n"); // Por favor, re-
stringe tu consulta
}
```

ifx_getsqlca (PHP 3>= 3.0.8, PHP 4)

Después de una consulta, obtiene el contenido de sqlca.sqlerrd[0..5]

```
array ifx_getsqlca (int result_id)
```

result_id es un identificador válido del resultado de **ifx_query()** o **ifx_prepare()**.

Devuelve una seudo fila (array asociativo) con los valores de sqlca.sqlerrd[0] a sqlca.sqlerrd[5] de una consulta ejecutada, representada ésta con un identificador de resultado *result_id*.

Para inserciones, actualizaciones y borrados los valores devueltos son aquellos definidos por el servidor después de que la consulta sea ejecutada. Esto da acceso al número de registros procesados y al valor de una columna de tipo serial en una consulta de inserción. Para consultas de selección, los valores son guardados cuando se prepara la sentencia. También permite conocer el número estimado de registros procesados. El uso de esta función evita el sobrecoste de ejecutar la consulta "select dbinfo('sqlca.sqlerrdx')", como obtener los valores guardados por el conector para Informix en el momento apropiado.

Ejemplo 1. Obtener los valores sqlca.sqlerrd[x]

```
/* suponiendo que la primera columna de la tabla 'sometable' es de tipo serial */
```

```
$qid = ifx_query("insert into sometable values(0, '2nd column', 'another col-
umn' ", $connid);
if (! $qid) {
    ... error ...
}
$sqlca = ifx_getsqlca ($qid);
$serial_value = $sqlca["sqlerrd1"];
echo "The serial value of the inserted row is : " . $se-
rial_value<br>\n"; // El valor de tipo serial del registro insertado es:
```

ifx_fetch_row (PHP 3>= 3.0.3, PHP 4)

Obtiene registros como un array (vector) enumerado

```
array ifx_fetch_row (int result_id [, mixed position])
```

Devuelve un array (vector) correspondiente a la fila leída o false si no hay más registros.

Las columnas blob son devueltas como identificadores de blob enteros (integer) para usarlos con **ifx_get_blob()** a menos que hayas usado ifx_textasvarchar(1) o ifx_byteasvarchar(1), en cuyo caso los blobs son devueltos como cadenas de texto. Devuelve FALSE si hubo error.

result_id es un identificador válido del resultado de **ifx_query()** o **ifx_prepare()** (sólo para consultas de selección).

position es un parámetro opcional para una operación de lectura sobre un cursor de tipo "scroll": "NEXT"(siguiente), "PREVIOUS"(anterior), "CURRENT"(actual), "FIRST"(primero), "LAST"(último) o un número. Si se especifica un número, un registro concreto es leído. Este parámetro opcional es sólo válido para cursos de tipo scroll.

ifx_fetch_row() lee el contenido de un registro de la consulta representada por el identificador de resultado indicado. La fila (registro) es devuelta en un array. Cada columna es guardada en un array, empezando éste desde cero.

Las llamadas posteriores a **ifx_fetch_row()** devolverán el registro siguiente en el resultado de la consulta, o false si no hay más filas.

Ejemplo 1. Leer registros

```
$rid = ifx_prepare ("select * from emp where name like " . $name,
                    $connid, IFX_SCROLL);
if (! $rid) {
    ... error ...
}
$rowcount = ifx_affected_rows($rid);
if ($rowcount > 1000) {
    printf ("Too many rows in result set (%d)\n<br>", $rowcount); // Demasiados reg-
istros en el resultado
    die ("Please restrict your query<br>\n"); // Por favor, re-
stringe tu consulta
}
if (! ifx_do ($rid)) {
    ... error ...
}
$row = ifx_fetch_row ($rid, "NEXT");
while (is_array($row)) {
    for(reset($row); $fieldname=key($row); next($row)) {
        $fieldvalue = $row[$fieldname];
        printf ("%s = %s,", $fieldname, $fieldvalue);
    }
    printf("\n<br>");
    $row = ifx_fetch_row ($rid, "NEXT");
```

```

}
ifx_free_result ($rid);

```

ifx_htmltbl_result (PHP 3>= 3.0.3, PHP 4)

Muestra todos los registros de una consulta en una tabla HTML

```
int ifx_htmltbl_result (int result_id [, string html_table_options])
```

Devuelve el número de registros leídos o FALSE si hubo error.

Muestra todas las filas de la consulta *result_id* dentro de una tabla html. El argumento segundo, opcional, es una cadena de parámetros del tag <table>

Ejemplo 1. Mostrar resultado como una tabla HTML

```

$rid = ifx_prepare ("select * from emp where name like " . $name,
                     $connid, IFX_SCROLL);
if (! $rid) {
    ... error ...
}
$rowcount = ifx_affected_rows ($rid);
if ($rowcount > 1000) {
    printf ("Too many rows in result set (%d)\n<br>", $rowcount); // Demasiados reg-
    istros en el resultado
    die ("Please restrict your query<br>\n"); // Por favor, re-
    stringe tu consulta
}
if (! ifx_do($rid)) {
    ... error ...
}

ifx_htmltbl_result ($rid, "border=\"2\"");
ifx_free_result($rid);

```

ifx_fieldtypes (PHP 3>= 3.0.3, PHP 4)

Obtiene los campos de una consulta SQL

```
array ifx_fieldtypes (int result_id)
```

Dada una consulta representada por *result_id* devuelve un array con los nombres de campo como llaves y los tipos como datos. Si no tuvo éxito da FALSE.

Ejemplo 1. Nombres y tipos de campos de una consulta SQL

```

$types = ifx_fieldtypes ($resultid);
if (! isset ($types)) {
    ... error ...
}
for ($i = 0; $i < count($types); $i++) {
    $fname = key($types);

```

```

    printf("%s :\t type = %s\n", $fname, $types[$fname]);
    next($types);
}

```

ifx_fieldproperties (PHP 3>= 3.0.3, PHP 4)

Indica las propiedades de los campos de una consulta SQL

```
array ifx_fieldproperties (int result_id)
```

Dada una consulta representada por *result_id* devuelve un array con los nombres de campo como llaves y las propiedades como datos. FALSE es devuelto si hubo error.

Devuelve las propiedades SQL de cada campo como un array. Las propiedades son codificadas así: "SQLTYPE;longitud;precisión;escala;ISNULLABLE" siendo SQLTYPE el tipo de dato definido en Informix como puede ser "SQLVCHAR"etc. e ISNULLABLE (puede ser nulo) igual a "Y"sí o "N"no.

Ejemplo 1. Propiedades de los campos de una consulta SQL

```

$properties = ifx_fieldtypes ($resultid);
if (! isset($properties)) {
    ... error ...
}
for ($i = 0; $i < count($properties); $i++) {
    $fname = key ($properties);
    printf ("%s:\t type = %s\n", $fname, $properties[$fname]);
    next ($properties);
}

```

ifx_num_fields (PHP 3>= 3.0.3, PHP 4)

Devuelve el número de columnas en una consulta

```
int ifx_num_fields (int result_id)
```

Dada una consulta representada por *result_id* devuelve el número de columnas o FALSE si se produjo un error.

Después de preparar o ejecutar una consulta, una llamada a esta función te da el número de columnas en la consulta.

ifx_num_rows (PHP 3>= 3.0.3, PHP 4)

Cuenta los registros ya leídos de una consulta

```
int ifx_num_rows (int result_id)
```

Da el número de registros ya leídos de una consulta representada por un *result_id* después de llamar a **ifx_query()** o **ifx_do()**.

ifx_free_result (PHP 3>= 3.0.3, PHP 4)

Libera los recursos de una consulta

```
int ifx_free_result (int result_id)
```

Libera los recursos representados por el identificador *result_id* de una consulta. Devuelve FALSE si hubo error.

ifx_create_char (PHP 3>= 3.0.6, PHP 4)

Crea un objeto char

```
int ifx_create_char (string param)
```

Crea un objeto char. *param* será el contenido del char.

ifx_free_char (PHP 3>= 3.0.6, PHP 4)

Elimina un objeto char

```
int ifx_free_char (int bid)
```

Borra el objeto char representado por el identificador del char *bid*. Devuelve FALSE si se produjo un error, en otro caso TRUE.

ifx_update_char (PHP 3>= 3.0.6, PHP 4)

Actualiza el contenido de un objeto char

```
int ifx_update_char (int bid, string content)
```

Actualiza el contenido de un objeto char representado por su identificador *bid*. *content* es una cadena con la información nueva. Devuelve FALSE si se produjo un error, en otro caso TRUE.

ifx_get_char (PHP 3>= 3.0.6, PHP 4)

Obtiene el contenido de un objeto char

```
int ifx_get_char (int bid)
```

Devuelve el contenido de un objeto char representado por su identificador *bid*.

ifx_create_blob (PHP 3>= 3.0.4, PHP 4)

Crea un objeto blob

```
int ifx_create_blob (int type, int mode, string param)
```

Crea un objeto blob.

type (tipo): 1 = TEXT, 0 = BYTE

mode (modo): 0 = el contenido del objeto blob es conservado en memoria, 1 = el contenido del objeto blob es mantenido en un archivo.

param: si mode = 0: apunta al contenido en memoria, si mode = 1: contiene el nombre del fichero.

Devuelve FALSE si hubo error, en otro caso el identificador del nuevo objeto blob.

ifx_copy_blob (PHP 3>= 3.0.4, PHP 4)

Duplica el objeto blob dado

```
int ifx_copy_blob (int bid)
```

Duplica el objeto blob dado. *bid* es el identificador del objeto blob a copiar.

Devuelve FALSE si hubo error, en otro caso el identificador del nuevo objeto blob.

ifx_free_blob (PHP 3>= 3.0.4, PHP 4)

Borra el objeto blob

```
int ifx_free_blob (int bid)
```

Elimina el objeto blob representado por el identificador *bid*. Devuelve FALSE si se produjo error, en otro caso TRUE.

ifx_get_blob (PHP 3>= 3.0.4, PHP 4)

Obtiene el contenido de un objeto blob

```
int ifx_get_blob (int bid)
```

Devuelve el contenido de un objeto blob representado por su identificador *bid*.

ifx_update_blob (PHP 3>= 3.0.4, PHP 4)

Actualiza el contenido de un objeto blob

```
ifx_update_blob (int bid, string content)
```

Actualiza el contenido de un objeto blob representado por su identificador *bid*. *content* es una cadena con el nuevo contenido. Devuelve FALSE si hubo error, en otro caso TRUE.

ifx_blobinfile_mode (PHP 3>= 3.0.4, PHP 4)

Define el modo por defecto para los blob en todas las consultas de selección

```
void ifx_blobinfile_mode (int mode)
```

Define el modo por defecto para los blob en todas las consultas de selección. El modo (mode) "0" quiere decir que guarda en memoria los blobs de tipo BYTE y modo "1" significa guardarlos en un archivo.

ifx_textasvarchar (PHP 3>= 3.0.4, PHP 4)

Define el modo por defecto para los campos de tipo text

```
void ifx_textasvarchar (int mode)
```

Define el modo por defecto para los campos de tipo text en todas las consultas de selección. Modo (mode) "0" devolverá un identificador de blob y "1" dará el contenido en un campo de tipo varchar.

ifx_byteasvarchar (PHP 3>= 3.0.4, PHP 4)

Define el modo por defecto para los campos de tipo byte

```
void ifx_byteasvarchar (int mode)
```

Define el modo por defecto para los campos de tipo byte en todas las consultas de selección. Modo (mode) "0" devolverá un identificador de blob y "1" dará el contenido en un campo de tipo varchar.

ifx_nullformat (PHP 3>= 3.0.4, PHP 4)

Define el valor por defecto cuando se leen valores nulos

```
void ifx_nullformat (int mode)
```

Define el valor por defecto cuando se leen valores nulos. Modo (mode) "0" devuelve "", y modo "1" devuelve "NULL".

ifxus_create_slob (PHP 3>= 3.0.4, PHP 4)

Crea un objeto slob y lo abre

```
int ifxus_create_slob (int mode)
```

Crea un objeto slob y lo abre. Modos: 1 = LO_RDONLY, 2 = LO_WRONLY, 4 = LO_APPEND, 8 = LO_RDWR, 16 = LO_BUFFER, 32 = LO_NOBUFFER o una combinación de ellos. También puedes usar nombres de constantes IFX_LO_RDONLY, IFX_LO_WRONLY, etc. Devuelve FALSE si hubo error, en otro caso el identificador del nuevo objeto slob.

ifx_free_slob (unknown)

Elimina un objeto slob

```
int ifxus_free_slob (int bid)
```

Borra un objeto slob. *bid* es el identificador del objeto slob. Devuelve FALSE si hubo error, TRUE en otro caso.

ifxus_close_slob (PHP 3>= 3.0.4, PHP 4)

Cierra un objeto slob

```
int ifxus_close_slob (int bid)
```

Cierra un objeto slob representado por su identificador de slob *bid*. Devuelve FALSE si hubo error, TRUE en otro caso.

ifxus_open_slob (PHP 3>= 3.0.4, PHP 4)

Abre un objeto slob

```
int ifxus_open_slob (long bid, int mode)
```

Abre un objeto slob. *bid* será un identificador de slob que válido. Modos: 1 = LO_RDONLY, 2 = LO_WRONLY, 4 = LO_APPEND, 8 = LO_RDWR, 16 = LO_BUFFER, 32 = LO_NOBUFFER o una combinación de ellos. Devuelve FALSE si hubo error, en otro caso el identificador del nuevo objeto slob.

ifxus_tell_slob (PHP 3>= 3.0.4, PHP 4)

Devuelve el fichero actual o la posición en memoria

```
int ifxus_tell_slob (long bid)
```

Devuelve el fichero actual o la posición en memoria de un objeto slob abierto, *bid* será un identificador de slob válido. Si hubo error entonces da FALSE.

ifxus_seek_slob (PHP 3>= 3.0.4, PHP 4)

Define el fichero o posición en memoria

```
int ifxus_seek_blob (long bid, int mode, long offset)
```

Define el fichero o posición en memoria de un objeto slob abierto, *bid* será un identificador de slob válido. Modos (mode): 0 = LO_SEEK_SET, 1 = LO_SEEK_CUR, 2 = LO_SEEK_END y *offset* es el desplazamiento en bytes. Si hubo error entonces da FALSE.

ifxus_read_slob (PHP 3>= 3.0.4, PHP 4)

Lee un número de bytes (nbytes) de un objeto slob

```
int ifxus_read_slob (long bid, long nbytes)
```

Lee un número de bytes (nbytes) de un objeto slob. *bid* es un identificador de slob válido y *nbytes* es el número de bytes a leer. Devuelve FALSE si hubo error, sino la cadena.

ifxus_write_slob (PHP 3>= 3.0.4, PHP 4)

Escribe una cadena en un objeto slob

```
int ifxus_write_slob (long bid, string content)
```

Escribe una cadena en un objeto slob. *bid* es un identificador de slob válido y *content* el contenido a escribir. Devuelve FALSE si hubo error, sino el número de bytes escritos.

XXXIV. Funciones InterBase

ibase_connect (PHP 3>= 3.0.6, PHP 4)

```
ibase_connect ()
```

ibase_pconnect (PHP 3>= 3.0.6, PHP 4)

```
ibase_pconnect ()
```

ibase_close (PHP 3>= 3.0.6, PHP 4)

```
ibase_close ()
```

ibase_query (PHP 3>= 3.0.6, PHP 4)

```
ibase_query ()
```

ibase_fetch_row (PHP 3>= 3.0.6, PHP 4)

```
ibase_fetch_row ()
```

ibase_free_result (PHP 3>= 3.0.6, PHP 4)

```
ibase_free_result ()
```

ibase_prepare (PHP 3>= 3.0.6, PHP 4)

```
ibase_prepare ()
```

ibase_bind (3.0.6 - 3.0.7 only, PHP 4 <= 4.0b4)

```
ibase_bind ()
```

ibase_execute (PHP 3>= 3.0.6, PHP 4)

```
ibase_execute ( )
```

ibase_free_query (PHP 3>= 3.0.6, PHP 4)

```
ibase_free_query ( )
```

ibase_timefmt (PHP 3>= 3.0.6, PHP 4)

```
ibase_timefmt ( )
```

XXXV. Ingres II functions

These functions allow you to access Ingres II database servers.

In order to have these functions available, you must compile php with Ingres support by using the `-with-ingres` option. You need the Open API library and header files included with Ingres II. If the `II_SYSTEM` environment variable isn't correctly set you may have to use `-with-ingres=DIR` to specify your Ingres installation directory.

When using this extension with Apache, if Apache does not start and complains with "PHP Fatal error: Unable to start `ingres_ii` module in Unknown on line 0" then make sure the environment variable `II_SYSTEM` is correctly set. Adding "`export II_SYSTEM="/home/ingres/II"`" in the script that starts Apache, just before launching `httpd`, should be fine.

Nota: If you already used PHP extensions to access other database servers, note that Ingres doesn't allow concurrent queries and/or transaction over one connection, thus you won't find any result or transaction handle in this extension. The result of a query must be treated before sending another query, and a transaction must be committed or rolled back before opening another transaction (which is automatically done when sending the first query).

ingres_connect (PHP 4 >= 4.0.2)

Open a connection to an Ingres II database.

```
resource ingres_connect ([string database [, string username [, string password]]])
```

Returns a Ingres II link resource on success, or false on failure.

ingres_connect() opens a connection with the Ingres database designated by *database*, which follows the syntax [*node_id*:::]*dbname[/svr_class]*.

If some parameters are missing, **ingres_connect()** uses the values in *php.ini* for *ingres.default_database*, *ingres.default_user* and *ingres.default_password*.

The connection is closed when the script ends or when **ingres_close()** is called on this link.

All the other *ingres* functions use the last opened link as a default, so you need to store the returned value only if you use more than one link at a time.

Ejemplo 1. **ingres_connect()** example

```
<?php
    $link = ingres_connect ("mydb", "user", "pass")
        or die ("Could not connect");
    print ("Connected successfully");
    ingres_close ($link);
?>
```

Ejemplo 2. **ingres_connect()** example using default link

```
<?php
    ingres_connect ("mydb", "user", "pass")
        or die ("Could not connect");
    print ("Connected successfully");
    ingres_close ();
?>
```

See also **ingres_pconnect()**, and **ingres_close()**.

ingres_pconnect (PHP 4 >= 4.0.2)

Open a persistent connection to an Ingres II database.

```
resource ingres_pconnect ([string database [, string username [, string password]]])
```

Returns a Ingres II link resource on success, or false on failure.

See **ingres_connect()** for parameters details and examples. There are only 2 differences between **ingres_pconnect()** and **ingres_connect()** : First, when connecting, the function will first try to find a (persistent) link that's already opened with the same parameters. If one is found, an identifier for it will be returned instead of opening a new connection. Second, the connection to the Ingres server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (**ingres_close()** will not close links established by **ingres_pconnect()**). This type of link is therefore called 'persistent'.

See also **ingres_connect()**, and **ingres_close()**.

ingres_close (PHP 4 >= 4.0.2)

Close an Ingres II database connection

```
bool ingres_close ([resource link])
```

Returns true on success, or false on failure.

ingres_close() closes the connection to the Ingres server that's associated with the specified link. If the *link* parameter isn't specified, the last opened link is used.

ingres_close() isn't usually necessary, as it won't close persistent connections and all non-persistent connections are automatically closed at the end of the script.

See also **ingres_connect()**, and **ingres_pconnect()**.

ingres_query (PHP 4 >= 4.0.2)

Send a SQL query to Ingres II

```
bool ingres_query (string query [, resource link])
```

Returns true on success, or false on failure.

ingres_query() sends the given *query* to the Ingres server. This query must be a valid SQL query (see the Ingres SQL reference guide)

The query becomes part of the currently open transaction. If there is no open transaction, **ingres_query()** opens a new transaction. To close the transaction, you can either call **ingres_commit()** to commit the changes made to the database or **ingres_rollback()** to cancel these changes. When the script ends, any open transaction is rolled back (by calling **ingres_rollback()**). You can also use **ingres_autocommit()** before opening a new transaction to have every SQL query immediately committed.

Some types of SQL queries can't be sent with this function :

- close (see **ingres_close()**).
- commit (see **ingres_commit()**).
- connect (see **ingres_connect()**).
- disconnect (see **ingres_close()**).
- get dbevent
- prepare to commit
- rollback (see **ingres_rollback()**).
- savepoint
- set autocommit (see **ingres_autocommit()**).
- all cursor related queries are unsupported

Ejemplo 1. **ingres_query()** example

```
<?php
    ingres_connect ($database, $user, $password);
```

```
ingres_query ("select * from table");
while ($row = ingres_fetch_row()) {
    echo $row[1];
    echo $row[2];
}
?>
```

See also **ingres_fetch_array()**, **ingres_fetch_object()**, **ingres_fetch_row()**, **ingres_commit()**, **ingres_rollback()** and **ingres_autocommit()**.

ingres_num_rows (PHP 4 >= 4.0.2)

Get the number of rows affected or returned by the last query

```
int ingres_num_rows ([resource link])
```

For delete, insert or update queries, **ingres_num_rows()** returns the number of rows affected by the query. For other queries, **ingres_num_rows()** returns the number of rows in the query's result.

Nota: This function is mainly meant to get the number of rows modified in the database. If this function is called before using **ingres_fetch_array()**, **ingres_fetch_object()** or **ingres_fetch_row()** the server will delete the result's data and the script won't be able to get them.

You should instead retrieve the result's data using one of these fetch functions in a loop until it returns false, indicating that no more results are available.

See also **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()** and **ingres_fetch_row()**.

ingres_num_fields (PHP 4 >= 4.0.2)

Get the number of fields returned by the last query

```
int ingres_num_fields ([resource link])
```

ingres_num_fields() returns the number of fields in the results returned by the Ingres server after a call to **ingres_query()**

See also **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()** and **ingres_fetch_row()**.

ingres_field_name (PHP 4 >= 4.0.2)

Get the name of a field in a query result.

```
string ingres_field_name (int index [, resource link])
```

ingres_field_name() returns the name of a field in a query result, or false on failure.

index is the number of the field and must be between 1 and the value given by **ingres_num_fields()**.

See also **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()** and **ingres_fetch_row()**.

ingres_field_type (PHP 4 >= 4.0.2)

Get the type of a field in a query result.

```
string ingres_field_type (int index [, resource link])
```

ingres_field_type() returns the type of a field in a query result, or false on failure. Examples of types returned are "IIAPI_BYTE_TYPE", "IIAPI_CHA_TYPE", "IIAPI_DTE_TYPE", "IIAPI_FLT_TYPE", "IIAPI_INT_TYPE", "IIAPI_VCH_TYPE". Some of these types can map to more than one SQL type depending on the length of the field (see **ingres_field_length()**). For example "IIAPI_FLT_TYPE" can be a float4 or a float8. For detailed information, see the Ingres/OpenAPI User Guide - Appendix C.

index is the number of the field and must be between 1 and the value given by **ingres_num_fields()**.

See also **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()** and **ingres_fetch_row()**.

ingres_field_nullable (PHP 4 >= 4.0.2)

Test if a field is nullable.

```
bool ingres_field_nullable (int index [, resource link])
```

ingres_field_nullable() returns true if the field can be set to the NULL value and false if it can't.

index is the number of the field and must be between 1 and the value given by **ingres_num_fields()**.

See also **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()** and **ingres_fetch_row()**.

ingres_field_length (PHP 4 >= 4.0.2)

Get the length of a field.

```
int ingres_field_length (int index [, resource link])
```

ingres_field_length() returns the length of a field. This is the number of bytes used by the server to store the field. For detailed information, see the Ingres/OpenAPI User Guide - Appendix C.

index is the number of the field and must be between 1 and the value given by **ingres_num_fields()**.

See also **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()** and **ingres_fetch_row()**.

ingres_field_precision (PHP 4 >= 4.0.2)

Get the precision of a field.

```
int ingres_field_precision (int index [, resource link])
```

ingres_field_precision() returns the precision of a field. This value is used only for decimal, float and money SQL data types. For detailed information, see the Ingres/OpenAPI User Guide - Appendix C.

index is the number of the field and must be between 1 and the value given by **ingres_num_fields()**.

See also **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()** and **ingres_fetch_row()**.

ingres_field_scale (PHP 4 >= 4.0.2)

Get the scale of a field.

```
int ingres_field_scale (int index [, resource link])
```

ingres_field_scale() returns the scale of a field. This value is used only for the decimal SQL data type. For detailed information, see the Ingres/OpenAPI User Guide - Appendix C.

index is the number of the field and must be between 1 and the value given by **ingres_num_fields()**.

See also **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()** and **ingres_fetch_row()**.

ingres_fetch_array (PHP 4 >= 4.0.2)

Fetch a row of result into an array.

```
array ingres_fetch_array ([int result_type [, resource link]])
```

ingres_fetch_array() Returns an array that corresponds to the fetched row, or false if there are no more rows.

This function is an extended version of **ingres_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 use the numeric index of the column or make an alias for the column.

```
ingres_query(select t1.f1 as foo t2.f1 as bar from t1, t2);
$result = ingres_fetch_array();
$foo = $result["foo"];
$bar = $result["bar"];
```

result_type can be II_NUM for enumerated array, II_ASSOC for associative array, or II_BOTH (default).

Speed-wise, the function is identical to **ingres_fetch_object()**, and almost as quick as **ingres_fetch_row()** (the difference is insignificant).

Ejemplo 1. **ingres_fetch_array()** example

```
<?php
ingres_connect ($database, $user, $password);

ingres_query ("select * from table");
while ($row = ingres_fetch_array()) {
    echo $row["user_id"]; # using associative array
    echo $row["fullname"];
    echo $row[1];          # using enumerated array
    echo $row[2];
```

```
}
```

```
?>
```

See also **ingres_query()**, **ingres_num_fields()**, **ingres_field_name()**, **ingres_fetch_object()** and **ingres_fetch_row()**.

ingres_fetch_row (PHP 4 >= 4.0.2)

Fetch a row of result into an enumerated array.

```
arrayingres_fetch_row ([resource link])
```

ingres_fetch_row() returns an array that corresponds to the fetched row, or false if there are no more rows. Each result column is stored in an array offset, starting at offset 1.

Subsequent call to **ingres_fetch_row()** would return the next row in the result set, or false if there are no more rows.

Ejemplo 1. **ingres_fetch_row()** example

```
<?php
ingres_connect ($database, $user, $password);

ingres_query ("select * from table");
while ($row = ingres_fetch_row()) {
    echo $row[1];
    echo $row[2];
}
?>
```

See also **ingres_num_fields()**, **ingres_query()**, **ingres_fetch_array()** and **ingres_fetch_object()**.

ingres_fetch_object (PHP 4 >= 4.0.2)

Fetch a row of result into an object.

```
objectingres_fetch_object ([int result_type [, resource link]])
```

ingres_fetch_object() Returns an object that corresponds to the fetched row, or false if there are no more rows.

This function is similar to **ingres_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: **II_ASSOC**, **II_NUM**, and **II_BOTH**.

Speed-wise, the function is identical to **ingres_fetch_array()**, and almost as quick as **ingres_fetch_row()** (the difference is insignificant).

Ejemplo 1. **ingres_fetch_object()** example

```
<?php
```

```

ingres_connect ($database, $user, $password);
ingres_query ("select * from table");
while ($row = ingres_fetch_object()) {
    echo $row->user_id;
    echo $row->fullname;
}
?>

```

See also [ingres_query\(\)](#), [ingres_num_fields\(\)](#), [ingres_field_name\(\)](#), [ingres_fetch_array\(\)](#) and [ingres_fetch_row\(\)](#).

ingres_rollback (PHP 4 >= 4.0.2)

Roll back a transaction.

```
bool ingres_rollback ([resource link])
```

ingres_rollback() rolls back the currently open transaction, actually canceling all changes made to the database during the transaction.

This closes the transaction. A new one can be open by sending a query with [ingres_query\(\)](#).

See also [ingres_query\(\)](#), [ingres_commit\(\)](#) and [ingres_autocommit\(\)](#).

ingres_commit (PHP 4 >= 4.0.2)

Commit a transaction.

```
bool ingres_commit ([resource link])
```

ingres_commit() commits the currently open transaction, making all changes made to the database permanent.

This closes the transaction. A new one can be open by sending a query with [ingres_query\(\)](#).

You can also have the server commit automatically after every query by calling [ingres_autocommit\(\)](#) before opening the transaction.

See also [ingres_query\(\)](#), [ingres_rollback\(\)](#) and [ingres_autocommit\(\)](#).

ingres_autocommit (PHP 4 >= 4.0.2)

Switch autocommit on or off.

```
bool ingres_autocommit ([resource link])
```

ingres_autocommit() is called before opening a transaction (before the first call to [ingres_query\(\)](#) or just after a call to [ingres_rollback\(\)](#) or [ingres_autocommit\(\)](#)) to switch the "autocommit" mode of the server on or off (when the script begins the autocommit mode is off).

When the autocommit mode is on, every query is automatically committed by the server, as if [ingres_commit\(\)](#) was called after every call to [ingres_query\(\)](#).

See also [ingres_query\(\)](#), [ingres_rollback\(\)](#) and [ingres_commit\(\)](#).

XXXVI. Funciones LDAP

Introducción a LDAP

LDAP es el protocolo de acceso a directorios ligero (Lightweight Directory Access Protocol), un protocolo usado para acceder a "Servidores de Directorio". El directorio es una clase especial de base de datos que contiene información estructurada en forma de árbol.

El concepto es similar a la estructura de directorios de los discos duros, pero en este caso, el directorio raíz es "El Mundo" y los subdirectorios de primer nivel son los "países". Niveles inferiores de la estructura de directorio contienen entradas para compañías, organizaciones o lugares, y en niveles aún inferiores se encuentran las entradas para la gente, y quizás de equipos informáticos y documentos.

Para referirse a un fichero en un subdirectorio del disco duro se usa algo como

/usr/local/misapps/docs

Las barras marcan cada división en la referencia al fichero, y la secuencia es leída de izquierda a derecha.

El equivalente a la referencia a un fichero en LDAP es el "distinguished name"(nombre distinguido), abreviado como "dn". Un ejemplo de dn podría ser.

cn=Pedro Pérez,ou=Contabilidad,o=Mi Compañía,c=ES

Las comas marcan cada división en la referencia, y la secuencia se lee de derecha a izquierda. Este dn se leería como ..

country = ES
organization = Mi Compañía
organizationalUnit = Contabilidad
commonName = Pedro Pérez

De la misma manera que no hay reglas estrictas sobre como organizar la estructura de directorios de un disco duro, un administrador de un servidor de directorio puede establecer cualquier estructura que sea útil para sus propósitos. Sin embargo hay algunos acuerdos tácitos que siempre deben seguirse. El mensaje es que no se puede escribir código para acceder un directorio si no se conoce algo de su estructura, igual que no se puede usar una base de datos sin algún conocimiento sobre lo que está disponible en ella.

Ejemplo de código completo

Recuperar información para todas las entradas donde el apellido empiece por "P" de un servidor de directorio, mostrando un extracto con el nombre y dirección de correo electrónico.

Ejemplo 1. ejemplo de búsqueda LDAP

```
<?php
// La secuencia básica para trabajar con LDAP es conectar, autenticarse,
// buscar, interpretar el resultado de la búsqueda y cerrar la conexión.

echo "<h3>Prueba de consulta LDAP</h3>";
echo "Conectando ...";
$ds=ldap_connect("localhost"); // Debe ser un servidor LDAP válido!
echo "El resultado de la conexión es ".$ds."<p>";

if ($ds) {
    echo "Autentificándose ...";
    $r=ldap_bind($ds);      // Autenticación anónima, típicamente con
                           // acceso de lectura
```

```

echo "El resultado de la autentificación es ".$r."<p>";

echo "Buscando (sn=P*) ...";
// Busqueda de entradas por apellidos
$sr=ldap_search($ds,"o=Mi Compañía, c=ES", "sn=P*");
echo "El resultado de la búsqueda es ".$sr."<p>";

echo "El número de entradas devueltas es ".ldap_count_entries($ds,$sr). "<p>";

echo "Recuperando entradas ...<p>";
$info = ldap_get_entries($ds, $sr);
echo "Devueltos datos de ".$info["count"]." entradas:<p>";

for ($i=0; $i<$info["count"]; $i++) {
    echo "dn es: ". $info[$i]["dn"] . "<br>";
    echo "La primera entrada cn es: ". $info[$i]["cn"][0] . "<br>";
    echo "La primera entrada email es: ". $info[$i]["mail"][0] . "<p>";
}

echo "Cerrando conexión";
ldap_close($ds);

} else {
    echo "<h4>Ha sido imposible conectar al servidor LDAP</h4>";
}
?>

```

Usando las llamadas LDAP de PHP

Es necesario conseguir y compilar la librerías cliente de LDAP ya sea del paquete ldap-3.3 de la Universidad de Michigan o del Netscape Directory SDK. También es necesario recompilar PHP con soporte LDAP activado para que la funciones LDAP de PHP funcionen.

Antes de usarse las llamadas LDAP se debe saber ..

- El nombre o dirección del servidor de directorio que se va a usar
- El "dn base" del servidor (la parte del directorio global contenida en ese servidor, que puede ser por ejemplo "o=Mi Compañía,c=ES")
- Si es necesaria contraseña para acceder al servidor (muchos servidores ofrecen acceso de lectura para usuarios anónimos pero requieren un password para cualquier otro acceso)

La secuencia típica de llamadas LDAP suele implementarse en aplicaciones que siguen el siguiente patrón:

```

ldap_connect() // establecer la conexión con el servidor
|
ldap_bind() // login anónimo o autenticado
|
Hacer búsquedas o actualizaciones en el directorio
y mostrar los resultados
|
ldap_close() // Cerrar la conexión

```

Más información

Mucha información acerca de LDAP puede ser consultada en

- Netscape (<http://developer.netscape.com/tech/directory/>)
- Universidad de Michigan (<http://www.umich.edu/~dirsvcs/ldap/index.html>)
- Proyecto OpenLDAP (<http://www.openldap.org/>)
- LDAP World (<http://elvira.innosoft.com/ldapworld>)

El SDK de Netscape contiene una Guía de Programación muy útil en formato html.

ldap_add (PHP 3, PHP 4)

Añade entradas a un directorio LDAP

```
int ldap_add (int identificador_deConexion, string dn, array entrada)
```

Devuelve verdadero cuando finaliza correctamente y falso se produce un error.

La función **ldap_add()** se usa para añadir entradas o registros a un directorio LDAP. El DN ("distinguished name", nombre distingible, la referencia de cualquier entrada LDAP) es especificado por *dn*. El array *entrada* especifica la información que quiere añadirse. Los valores del array son indexados por sus propios atributos. En caso de valores múltiples para un mismo atributo, son indexados usando enteros empezando con 0.

```
entry["atributo1"] = valor
entry["atributo2"][0] = valor1
entry["atributo2"][1] = valor2
```

Ejemplo 1. Ejemplo completo con login atentificado

```
<?php
$ds=ldap_connect("localhost"); // Asumimos que el servidor LDAP está en el
                                // servidor local

if ($ds) {
    // autentificarse con el dn apropiado para tener permisos de modificación
    $r=ldap_bind($ds, "cn=root, o=Mi Compañía, c=ES", "secreto");

    // prepare data
    $info[ "cn" ]="Pedro Pérez";
    $info[ "sn" ]="Pedro";
    $info[ "mail" ]="pedro.p@algun.sitio";
    $info[ "objectclass" ]="persona";

    // add data to directory
    $r=ldap_add($ds, "cn=Pedro Pérez, o=Mi Compañía, c=ES", $info);

    ldap_close($ds);
} else {
    echo "Ha sido imposible conectar al servidor LDAP";
}
?>
```

ldap_mod_add (PHP 3>= 3.0.8, PHP 4)

Añade valores de atributos

```
int ldap_mod_add (int identificador_deConexion, string dn, array entrada)
```

Devuelve verdadero cuando finaliza correctamente y falso se produce un error.

Esta función añadir uno o varios atributos al *dn* especificado. Realiza la modificación al nivel de atributos, en vez de hacerlo al nivel de objetos. Las modificaciones a nivel de objeto son proporcionadas por la función **ldap_add()**.

ldap_mod_del (PHP 3>= 3.0.8, PHP 4)

Borra valores de atributos

```
int ldap_mod_del (int identificador_deConexion, string dn, array entrada)
```

returns true on success and false on error.

Esta función elimina atributos del dn especificado. Realiza la modificación a nivel de atributos, en vez de hacerlo a nivel de objetos. Las modificaciones a nivel de objeto son proporcionadas por la función **ldap_del()**.

ldap_mod_replace (PHP 3>= 3.0.8, PHP 4)

Reemplaza valores de atributos

```
int ldap_mod_replace (int identificador_deConexion, string dn, array entrada)
```

Devuelve verdadero cuando finaliza correctamente y falso se produce un error.

Esta función reemplaza atributos del dn especificado. Realiza la modificación a nivel de atributos, en vez de hacerlo a nivel de objetos. Las modificaciones a nivel de objeto son proporcionadas por la función **ldap_modify()**.

ldap_bind (PHP 3, PHP 4)

Autentifica en un directorio LDAP

```
int ldap_bind (int identificador_deConexion [, string rdn_de_usuario [, string contraseña]])
```

Se conecta a un directorio LDAP con un RDN y su contraseña. Devuelve verdadero cuando finaliza correctamente y falso se produce un error.

ldap_bind() se conecta al directorio con un determinado usuario. *rdn_de_usuario* y *contraseña* son opcionales. Si no son especificados, se intenta el acceso anónimo.

ldap_close (PHP 3, PHP 4)

Cierra una conexión a un servidor LDAP

```
int ldap_close (int identificador_deConexion)
```

Devuelve verdadero cuando finaliza correctamente y falso se produce un error.

ldap_close() cierra la conexión con el servidor LDAP asociada con el *identificador_deConexion* especificado.

Esta llamada es idéntica internamente a **ldap_unbind()**. La API LDAP usa la llamada **ldap_unbind()**, y por lo tanto quizás deba usar esta llamada en lugar de **ldap_close()**.

ldap_connect (PHP 3, PHP 4)

Conecta con un servidor LDAP

```
int ldap_connect ([string nombre_host [, int puerto]])
```

Devuelve un identificador de conexión positivo en caso de éxito, ó falso si ocurre algún error.

ldap_connect() establece una conexión con el servidor LDAP especificado en *nombre_host* y *puerto*. Ambos argumentos son opcionales. Si no se especifican, el identificador de la conexión LDAP actualmente abierta es devuelto. Si sólo es especificado *nombre_host* el puerto tomado por defecto es el 389.

ldap_count_entries (PHP 3, PHP 4)

Cuenta el número de entradas de una búsqueda

```
int ldap_count_entries (int identificador_deConexion, int identificador_deResultado)
```

Devuelve el número de entradas del resultado o falso si ha ocurrido algún error.

ldap_count_entries() devuelce el número de entradas almacenadas en el resultado de operaciones de búsqueda previas. *identificador_deResultado* identifica el resultado ldap interno al que hacemos referencia.

ldap_delete (PHP 3, PHP 4)

Borra una entrada de un directorio

```
int ldap_delete (int identificador_deConexion, string dn)
```

Devuelve verdadero cuando finaliza correctamente y falso se produce un error.

La función **ldap_delete()** borra la entrada particular dn del directorio LDAP.

ldap_dn2ufn (PHP 3, PHP 4)

Convierte un dn al formato User Friendly Naming

```
string ldap_dn2ufn (string dn)
```

La función **ldap_dn2ufn()** es usada para convertir un DN en un formato más amigable para el usuario.

ldap_explode_dn (PHP 3, PHP 4)

Divide un DN en las partes que le componen

```
array ldap_explode_dn (string dn, int con_atributos)
```

La función **ldap_explode_dn()** es usada para dividir un DN devuelto por **ldap_get_dn()** en las partes que le componen. Cada parte es conocida como Relative Distinguished Name (Nombre Relativo Distinguible) abreviado como RDN. **ldap_explode_dn()** devuelve un array con todos esos componentes. *con_atributos* sirve para especificar si los RDN se devuelven sólo como valores o con sus atributos también (es decir, en un formato atributo=valor). Hay que poner *with_attrib* a 0 para obtener también los atributos y a 1 para obtener sólo los valores.

ldap_first_attribute (PHP 3, PHP 4)

Devuelve el primer atributo

```
string ldap_first_attribute (int identificador_deConexion, int identificador_deEntrada_en_resultado, int identificador_ber)
```

Devuelve el primer atributo en la entrada o falso si ocurre algún error.

De manera similar a leer entradas, los atributos también son leídos de uno en uno de una entrada en particular del directorio. **ldap_first_attribute()** devuelve el primer atributo en la entrada a la que apunta el *identificador_de_entrada_en_resultado*. El resto de los atributos son obtenidos llamando a la función **ldap_next_attribute()** sucesivamente. El parámetro *identificador_ber* es el identificador del puntero interno a memoria. Es pasado por referencia. El mismo *identificador_ber* es pasado a la función **ldap_next_attribute()** que modifica dicho puntero.

Ver también **ldap_get_attributes()**

ldap_first_entry (PHP 3, PHP 4)

Devuelve el identificador del primer resultado

```
int ldap_first_entry (int identificador_deConexion, int identificador_deResultado)
```

Devuelve el identificador de la primera entrada del resultado ó falso en caso de error.

Las entradas en un resultado LDAP son leídas secuencialmente usando las funciones **ldap_first_entry()** y **ldap_next_entry()**. **ldap_first_entry()** devuelve el identificador de la primera entrada del resultado. Este identificador es entonces suministrado a la rutina **lap_next_entry()** para obtener sucesivas entradas del resultado.

Ver también **ldap_get_entries()**.

ldap_free_result (PHP 3, PHP 4)

Libera la memoria que almacena los resultados

```
int ldap_free_result (int identificador_deResultado)
```

Devuelve verdadero cuando finaliza correctamente y falso se produce un error.

ldap_free_result() libera la memoria reservada internamente para almacenar el resultado de búsquedas LDAP asociada al identificador *identificador_de_resultado*. Toda la memoria de resultados es automáticamente liberada al finalizarse la ejecución de un script.

Normalmente la memoria reservada para resultados ldap se libera al final del script. En caso de que el script realice sucesivas búsquedas que devuelvan conjuntos de resultados grandes, puede utilizarse **ldap_free_result()** para mantener bajo el uso de memoria del script durante su ejecución.

ldap_get_attributes (PHP 3, PHP 4)

Obtiene los atributos de una entrada de un resultado de búsqueda

```
array ldap_get_attributes (int identificador_deConexion, int identificador_deEntrada_deResultado)
```

Devuelve una completa información de la entrada en un array multidimensional o falso en caso de error.

La función **ldap_get_attributes()** es usada para simplificar el leer atributos y valores de una entrada de un resultado de búsqueda. El valor de retorno es un array multidimensional de atributos y sus valores.

Teniendo localizado una entrada específica en el directorio se puede conseguir la información que contiene dicha entrada usando esta llamada. Puede usar esta función para aplicaciones que naveguen por las entradas del directorio y/o cuando no se conoce la estructura de las entradas del directorio. En otras aplicaciones se busca un atributo específico, como la dirección de email o los apellidos y no importa el resto de información contenida..

`valor_devuelto["count"]` = número de atributos en la entrada

`valor_devuelto[0]` = primer atributo

`valor_devuelto[n]` = enésimo atributo

`valor_devuelto["atributo"]["count"]` = número de vaslores del atributo

`valor_devuelto["atributo"][0]` = primer valor del atributo

`valor_devuelto["atributo"][i]` = iésimo valor del atributo

Ejemplo 1. Mostrar la lista de atributos contenida en una entrada específica de un directorio

```
// $ds es un identificador de conexión al directorio
// $sr es un resultado de búsqueda válido de una llamada
// anterior a una de las funciones de búsqueda en directorios
// ldap.

$entrada = ldap_first_entry($ds, $sr);

$atributos = ldap_get_attributes($ds, $entrada);

echo $atributos["count"]." atributos contenidos en esta entrada:<p>";

for ($i=0; $i<$atributos["count"]; $i++)
    echo $atributos[$i]."<br>";
```

Ver también **ldap_first_attribute()** y **ldap_next_attribute()**

ldap_get_dn (PHP 3, PHP 4)

Obtiene el DN de una entrada de un resultado

```
string ldap_get_dn (int identificador_deConexion, int identificador_deEntrada_deResultado)
```

Devuelve el DN de la entrada del resultado o falso en caso de error.

La función **ldap_get_dn()** se utiliza para obtener el DN de una entrada de un resultado de búsqueda.

ldap_get_entries (PHP 3, PHP 4)

Obtiene todas las entradas de un resultado

```
array ldap_get_entries ( int identificador_deConexion, int identificador_deResultado )
```

Devuelve una completa información de un resultado de búsqueda en un array multidimensional o falso en caso de error.

La función **ldap_get_entries()** es usada para simplificar el leer múltiples entradas de un resultado y después leer sus atributos y múltiples valores. Toda la información es devuelta por una llamada a una función en forma de array multidimensional. La estructura del array es como se muestra más abajo.

Los índices de atributos son convertidos a minúsculas. (Los atributos de servidores de directorios son indiferentes a las mayúsculas/minúsculas, pero no cuando son usados como índices de arrays)

`valor_devuelto["count"]` = número de entradas del resultado

`valor_devuelto[0]` : contiene los detalles de la primera entrada

`valor_devuelto[i]["dn"]` = DN de la entrada iésima del resultado

`valor_devuelto[i]["count"]` = número de atributos de la entrada iésima

`valor_devuelto[i][j]` = jésimo atributo de la iésima entrada del resultado

`valor_devuelto[i]["atributo"]["count"]` = número de valores para "atributo"
en la entrada iésima

`valor_devuelto[i]["atributo"][j]` = jésimo valor de "atributo" en la entrada
iésima

Ver también **ldap_first_entry()** y **ldap_next_entry()**

ldap_get_values (PHP 3, PHP 4)

Obtiene todos los valores de un atributo de una entrada

```
array ldap_get_values ( int identificador_deConexion, int identificador_deEntrada_deResultado, string atributo )
```

Devuelve un array de valores del atributo o falso en caso de error.

La función **ldap_get_values()** se utiliza para obtener todos los valores de un atributo de una entrada. La entrada del resultado es especificada por el *identificador_de_entrada_de_resultado*. El número de valores se almacena en el índice "count" del array devuelto. Los valores individuales se almacenan con índices enteros en el array. El primer índice es 0.

Esta llamada necesita un *identificador_de_entrada_de_resultado*, por lo que necesita ser precedida por una de las llamadas de búsqueda ldap y una llamada para obtener una entrada en particular del resultado.

La aplicación debe ser o bien programada específicamente para buscar ciertos atributos (como apellidos o email) o bien utilizar la función **ldap_get_attributes()** para averiguar qué atributos existen para una entrada dada, antes de llamar a **ldap_get_values()**.

LDAP permite más de un valor para cada atributo, por lo que se puede, por ejemplo, almacenar varias direcciones de email para una persona en el directorio y nombrar a ese atributo como "email"

`valor_devuelto["count"]` = número de valores del atributo

valor_devuelto[0] = primer valor del atributo
 valor_devuelto[i] = iésimo valor del atributo

Ejemplo 1. Listar todos los valores del atributo "email" de una entrada de un directorio

```
// $ds es un identificador de conexión al directorio

// $sr es un resultado de búsqueda válido de una llamada
// anterior a una de las funciones de búsqueda en directorios
// ldap.

// $entrada es un identificador de entrada válido de una llamada
// anterior a una de las funciones que devuelven una entrada de
// directorio

$valores = ldap_get_values($ds, $entrada, "email");

echo $valores["count"]." direcciones de email para esta entrada.<p>";

for ($i=0; $i < $valores["count"]; $i++)
  echo $valores[$i]."<br>";
```

ldap_get_values_len (PHP 3>= 3.0.13, PHP 4 >= 4.0RC2)

Obtiene todos los valores binarios de un atributo de una entrada

```
array ldap_get_values_len ( int identificador_deConexion, int
identificador_deEntrada_deResultado, string atributo )
```

Devuelve un array de valores del atributo o falso en caso de error.

La función **ldap_get_values_len()** se utiliza para obtener todos los valores de un atributo de una entrada de un resultado de búsqueda. La entrada es especificada por el *identificador_de_entrada_de_resultado*. El número de valores se almacena en el índice "count" del array devuelto. Los valores individuales se almacenan con índices enteros en el array. El primer índice es 0.

Esta función se utiliza exactamente como **ldap_get_values()** salvo que permite manejar datos binarios y no cadenas de caracteres.

ldap_list (PHP 3, PHP 4)

Búsqueda Single-level (Nivel Único)

```
int ldap_list ( int identificador_deConexion, string dn_base, string filtro [, array
atributos] )
```

Devuelve un identificador de resultado de búsqueda o falso en caso de error.

ldap_list() realiza la búsqueda según el filtro especificado en el directorio con el alcance LDAP_SCOPE_ONELEVEL.

LDAP_SCOPE_ONELEVEL significa que la búsqueda solo devuelve información que se encuentre en el nivel inmediatamente inferior al *dn_base* especificado en la llamada a la función. (Equivalente a ejecutar "ls" en un unix y obtener un listado de ficheros y carpetas en el directorio de trabajo actual.)

Esta llamada toma un cuarto parámetro opcional, que es un array de los atributos requeridos. Consulte las notas de la función **ldap_search()**.

Ejemplo 1. Produce una lista de todas las unidades organizativas de una compañía

```
// $ds es un identificador de conexión válido.

$dnbase = "o=Mi Compañía, c=ES";
$solonecesito = array("ou");

$sr=ldap_list($ds, $dnbase, "ou=*", $solonecesito);

$info = ldap_get_entries($ds, $sr);

for ($i=0; $i<$info["count"]; $i++)
    echo $info[$i]["ou"][0] ;
```

ldap_modify (PHP 3, PHP 4)

Modifica una entrada LDAP

```
int ldap_modify (int identificador_deConexion, string dn, array entrada)
```

Devuelve verdadero cuando finaliza correctamente y falso se produce un error.

La función **ldap_modify()** se utiliza para modificar entradas existentes en un directorio LDAP. La estructura de la entrada es igual a la de **ldap_add()**.

ldap_next_attribute (PHP 3, PHP 4)

Obtiene el siguiente atributo de una entrada

```
string ldap_next_attribute (int identificador_deConexion, int identificador_deEntrada_deResultado, int identificador_ber)
```

Devuelve el siguiente atributo de una entrada o falso en caso de error.

ldap_next_attribute() es llamado para recuperar los atributos de una entrada. El estado interno del puntero es mantenido por el *identificador_ber*, que es pasado por referencia a la función. La primera llamada a **ldap_next_attribute()** es realizada con el *identificador_deEntrada_deResultado* devuelto por la función **ldap_first_attribute()**.

Ver también **ldap_get_attributes()**

ldap_next_entry (PHP 3, PHP 4)

Obtiene la siguiente entrada de un resultado

```
int ldap_next_entry (int identificador_deConexion, int identificador_deEntrada_deResultado)
```

Devuelve el identificador de la siguiente entrada del resultado. Las entradas deben haber sido leidas al principio con **ldap_first_entry()**. Si no hay más entradas en el resultado devuelve falso.

La función **ldap_next_entry()** se utiliza para obtener las entradas almacenadas en un resultado. Llamadas sucesivas a la función **ldap_next_entry()** devuelven las entradas una a una hasta que ya no quedan más entradas. La primera llamada a **ldap_next_entry()** se realiza después de llamar a **ldap_first_entry()**.

Ver también **ldap_get_entries()**

ldap_read (PHP 3, PHP 4)

Lee una entrada

```
int ldap_read (int identificador_de_conexión, string dn_base, string filtro [, array atributos])
```

Devuelve un identificador de resultado de búsqueda o falso en caso de error.

ldap_read() realiza la búsqueda según el filtro especificado con alcance LDAP_SCOPE_BASE, por lo que es equivalente a leer cualquier entrada del directorio.

No se permiten filtros vacíos. Si se pretende obtener absolutamente toda la información, se debe usar un filtro del tipo "objectClass=*". Si conoce que tipos de entradas son usadas en el servidor de directorio es conveniente usar el filtro apropiado, como por ejemplo "objectClass=inetOrgPerson".

Esta llamada toma un cuarto parámetro opcional que es un array de los atributos requeridos. Consulte las notas de la función **ldap_search()**.

ldap_search (PHP 3, PHP 4)

Busca en un arbol LDAP

```
int ldap_search (int identificador_deConexion, string dn_base, string filtro [, array atributos])
```

Devuelve un identificador de resultado de búsqueda o falso en caso de error.

ldap_search() realiza la búsqueda según el filtro especificado con alcance LDAP_SCOPE_SUBTREE. Esto es equivalente a buscar en el directorio entero. *dn_base* especifica el DN base para el directorio.

Existe un cuarto parámetro opcional que puede ser añadido para restringir los atributos y valores devueltos por el servidor a sólo los requeridos. Es mucho más eficiente que la acción por defecto (que devolverá todos los atributos y sus valores asociados). El uso del cuarto parámetro debe ser por tanto considerado una práctica recomendable.

El cuerto parámetro es un array estándar de PHP con los atributos requeridos, por ejemplo array("email","sn","cn"). Nota: "dn"siempre es devuelto independientemente de que tipos de atributos sean solicitados.

También es necesario resaltar que algunos servidores de directorio están configurados para devolver un cierto número de entradas como máximo. Si esto ocurre, el servidor indicará que solo devuelve un conjunto de resultados parcial.

El filtro de búsqueda puede ser simple o avanzado, usando operadores booleanos en el formato descrito en la documentación sobre LDAP (Consulte el Netscape Directory SDK (<http://developer.netscape.com/tech/directory/>) para obtener completa información sobre filtros).

El ejemplo de abajo recupera la unidad organizativa (ou), apellidos nombre común y dirección de email para todas las personas de "Mi Compañía" donde los apellidos o el nombre común contiene la subcadena \$persona. Este ejemplo usa un filtro booleano para indicar al servidor que busque la información en más de un atributo.

Ejemplo 1. Búsqueda LDAP

```
// $ds es un identificador de conexión válido
// $persona es todo o parte del nombre de una persona, por ejemplo "Pe"
$dn = "o=Mi Compañía, c=ES";
$filter="(|(sn=$persona*)(givenname=$persona*))";
$solonecesito = array( "ou", "sn", "givenname", "mail");

$sr=ldap_search($ds, $dn, $filter, $solonecesito);

$info = ldap_get_entries($ds, $sr);

print $info["count"]." entradas devueltas<p>";
```

ldap_unbind (PHP 3, PHP 4)

Hace logout de un directorio LDAP

```
int ldap_unbind (int identificador_deConexion)
```

Devuelve verdadero cuando finaliza correctamente y falso se produce un error.

La función **ldap_unbind()** hace logout, desautentifica de un directorio LDAP.

ldap_err2str (PHP 3>= 3.0.13, PHP 4 >= 4.0RC2)

Convierte un código numérico de error LDAP en un mensaje.

```
string ldap_err2str (int numerr)
```

Devuelve una cadena con el mensaje de error.

Esta función devuelve una cadena con el mensaje de error explicativo del código numérico de error numerr. Aunque los códigos de error LDAP están estandarizados, diferentes librerías devuelven mensajes textuales de error diferentes o incluso localizados. Nunca se debe comprobar la existencia de un error específico por el mensaje textual, sino por el código numérico.

Var también **ldap_errno()** y **ldap_error()**.

Ejemplo 1. Enumerando todos los mensajes de error LDAP

```
<?php
for($i=0; $i<100; $i++) {
    printf("Error $i: %s<br>\n", ldap_str2err($i));
}
?>
```

ldap_errno (PHP 3>= 3.0.12, PHP 4 >= 4.0RC2)

Devuelve el código numérico de error para el último comando LDAP.

```
int ldap_errno (int identificador_de_conexión)
```

Devuelve el código de error del último comando LDAP para la conexión especificada.

Esta función devuelve el código numérico de error, que está estandarizado, producido por el último comando LDAP y en la conexión especificada. Este número puede ser convertido en un mensaje textual de error usando **ldap_err2str()**.

A menos que decremente el nivel de alerta en su fichero `php3.ini` (ó `php.ini`) o anteponga a los comandos LDAP en símbolo @ (arroba) para suprimir las alerts y warnings, los errores producidos serán mostrados automáticamente en el código HTML generado.

Ejemplo 1. Generando y capturando un error

```
<?php
// Este ejemplo contiene un error, que será capturado.
$ld = ldap_connect("localhost");
$bind = ldap_bind($ld);
// error de sintaxis en la expresión del filtro (codigo
// de error 87). Debería ser "objectclass=*".
$res = @ldap_search($ld, "o=Mi Compañía, c=ES", "objectclass");
if (!$res) {
    printf("LDAP-Código Error: %s<br>\n", ldap_errno($ld));
    printf("LDAP-Mensaje Error: %s<br>\n", ldap_error($ld));
    die("Argh!<br>\n");
}
$info = ldap_get_entries($ld, $res);
printf("%d entradas encontradas.<br>\n", $info["count"]);
?>
```

Ver también **ldap_err2str()** y **ldap_error()**.

ldap_error (PHP 3>= 3.0.12, PHP 4 >= 4.0RC2)

Devuelve el mensaje de error del último comando LDAP

```
string ldap_error (int identificador_de_conexión)
```

Devuelve una cadena con el mensaje de error.

Esta función devuelve una cadena con el mensaje de error explicativo del error generado por el último comando LDAP en la conexión especificada. Aunque los códigos de error LDAP están estandarizados, diferentes librerías devuelven mensajes textuales de error diferentes o incluso localizados. Nunca se debe comprobar la existencia de un error específico por el mensaje textual, sino por el código numérico.

A menos que decremente el nivel de alerta en su fichero `php3.ini` (ó `php.ini`) o anteponga a los comandos LDAP en símbolo @ (arroba) para suprimir las alerts y warnings, los errores producidos serán mostrados automáticamente en el código HTML generado.

Ver también **ldap_err2str()** y **ldap_errno()**.

XXXVII. Funciones de Correo

The **mail()** Funciones que permiten enviar correo.

mail (PHP 3, PHP 4)

Envía correo

```
bool mail (string para, string sobre, string mensaje [, string cabeceras_adicionales])
```

Mail() envía automáticamente el mensaje especificado en *mensaje* al destinatario especificado en *para*. Para especificar múltiples destinatarios se puede hacer incluyendo una coma entre las direcciones en *para*.

Ejemplo 1. Enviando correo.

```
mail("pepito@loquesea.es", "Sobre este tema", "Linea 1\nLinea 2\nLinea 3");
```

Si se añadiera una cadena como cuarto argumento, esta cadena sería enviada al final de la cabecera. Esto se usa normalmente para enviar cabeceras extra en los mensajes. Si se desea enviar varias cabeceras extra el mecanismo será el mismo separándolas una linea.

Ejemplo 2. Enviando correo con varias cabeceras.

```
mail("pepito@loquesea.es", "El tema", $message,  
"From: webmaster@$SERVER_NAME\nReply-To: webmaster@$SERVER_NAME\nX-  
Mailer: PHP/" . phpversion());
```


XXXVIII. Funciones matemáticas

Introducción

Estas funciones matemáticas solo manejan valores dentro de los rangos de los tipos long y double de su ordenador. Si necesita manejar números mayores, pege un vistazo a [funciones matemáticas de precisión arbitraria](#).

Constantes matemáticas

Los siguientes valores están definidos como constantes en PHP por la extensión matemática:

Tabla 1. Constantes matemáticas

Constante	Valor	Descripción
M_PI	3.14159265358979323846	El valor de π (pi)

Abs (PHP 3, PHP 4)

Valor absoluto

```
mixed abs (mixed number)
```

Devuelve el valor absoluto de un número. Si el número del argumento es decimal, el tipo de retorno también es decimal, de otra manera devuelve un entero.

Acos (PHP 3, PHP 4)

Arco coseno

```
float acos (float arg)
```

Devuelve el arco coseno de arg en radianes.

Vea también **asin()** y **atan()**.

Asin (PHP 3, PHP 4)

Arco seno

```
float asin (float arg)
```

Devuelve el arco seno de arg en radianes.

Vea también **acos()** y **atan()**.

Atan (PHP 3, PHP 4)

Arco tangente

```
float atan (float arg)
```

Devuelve la arco tangente de arg en radianes.

Vea también **acos()** y **atan()**.

Atan2 (PHP 3>= 3.0.5, PHP 4)

Arco tangente de dos variables

```
float atan2 (float y, float x)
```

Esta función calcula la arco tangente de las dos variables x e y. Es similar a el cálculo de la arco tangente de y / x, excepto que los signos de ambos argumentos son usados para determinar el cuadrante del resultado

La función devuelve el resultado en radianes, el cual está entre -PI y PI (inclusive).

Vea también **acos()** y **atan()**.

base_convert (PHP 3>= 3.0.6, PHP 4)

Convierte un número entre bases arbitrarias

```
string base_convert (string number, int frombase, int tobase)
```

Devuelve una cadena conteniendo el *number* representado en base *tobase*. La base en la cual *number* es dado viene especificada en *frombase*. Tanto *frombase* y *tobase* tienen que estar entre 2 y 36, inclusive. Los dígitos en números con una base mayor que 10 serán representados con las letras a-z, a vale 10, b vale 11 y z vale 36.

Ejemplo 1. base_convert()

```
$binary = base_convert($hexadecimal, 16, 2);
```

BinDec (PHP 3, PHP 4)

binario a decimal

```
int bindec (string binary_string)
```

Devuelve el equivalente decimal del número binario representado por el argumento *binary_string*.

BinDec convierte un número binario en un número decimal. El mayor número que puede ser convertido son 31 bits de unos o 2147483647 en decimal.

Vea también la función **decbin()** .

Ceil (PHP 3, PHP 4)

Redondea fracciones hacia arriba

```
int ceil (float number)
```

Devuelve el valor entero superior más cercano a *number*. El uso de **ceil()** con enteros es una perdida de tiempo absoluta.

NOTA: PHP/FI 2's **ceil()** devuelve un float. Use: \$new = (double)ceil(\$number); para tener el comportamiento antiguo.

Vea también **floor()** y **round()**.

Cos (PHP 3, PHP 4)

coseno

```
float cos (float arg)
```

Devuelve el coseno de arg en radianes.

Vea también **sin()** y **tan()**.

DecBin (PHP 3, PHP 4)

decimal a binario

```
string decbin (int number)
```

Devuelve una cadena conteniendo la representación en binario de el número dado en el argumento. El número mayor que puede ser convertido es 2147483647 en decimal que resulta una cadena de 31 unos.

Vea también la función **bindec()**.

DecHex (PHP 3, PHP 4)

decimal a hexadecimal

```
string dechex (int number)
```

Devuelve una cadena conteniendo la representación hexadecimal del número dado en el argumento. El mayor número que puede ser convertido es 2147483647 en decimal resultando "7fffffff".

Vea también la función **hexdec()**.

DecOct (PHP 3, PHP 4)

decimal a octal

```
string decoct (int number)
```

Devuelve una cadena conteniendo la representación octal del número dado en el argumento. Returns a string containing an octal representation of the given number argument. El mayor número que puede ser convertido es 2147483647 en decimal resultando "17777777777". Vea también **octdec()**.

Exp (PHP 3, PHP 4)

e elevado a...

```
float exp (float arg)
```

Devuelve el número e elevado a la potencia de *arg*.

Vea también **pow()**.

Floor (PHP 3, PHP 4)

redondea fracciones hacia abajo

```
int floor (float number)
```

Devuelve el valor entero inferior más cercano a *number*. El uso de **floor()** con enteros es una perdida de tiempo absoluta.

NOTA: PHP/FI 2's **floor()** devuelve un float. Use: `$new = (double)floor($number);` para tener el comportamiento antiguo.

Vea también **ceil()** y **round()**.

getrandmax (PHP 3, PHP 4)

Muestra el mayor valor aleatorio posible

```
int getrandmax (void)
```

Devuelve el valor máximo que puede devolver una llamada a **rand()**.

Vea también **rand()**, **srand()**, **mt_rand()**, **mt_srand()** y **mt_getrandmax()**.

HexDec (PHP 3, PHP 4)

hexadecimal a decimal

```
int hexdec (string hex_string)
```

Devuelve el equivalente decimal de el número hexadecimal representado por el argumento *hex_string*. HexDec convierte una cadena hexadecimal en un número decimal. El número mayor que puede ser convertido es 7fffffff o 2147483647 en decimal.

Vea también la función **dechex()** .

Log (PHP 3, PHP 4)

Logaritmo natural

```
float log (float arg)
```

Devuelve el logaritmo de *arg*.

Log10 (PHP 3, PHP 4)

Logaritmo en base-10

```
float log10 (float arg)
```

Devuelve el logaritmo en base-10 de arg.

max (PHP 3, PHP 4)

encuentra el valor mayor

```
mixed max (mixed arg1, mixed arg2, mixed argn)
```

max() devuelve el número mayor de los valores de los parámetros.

Si el primer parámetro es un array, **max()** devuelve el mayor valor en ese array. Si el primer parámetro es un entero, string o double, necesita al menos dos parámetros y **max()** devuelve el mayor número de estos valores. Puede comparar un número ilimitado de valores.

Si uno o más de los valores es un double, todos los valores serán tratados como doubles, y devolverá un double. Si ninguno de los valores es un double, todos ellos serán tratados como enteros, y se devolverá un entero.

min (PHP 3, PHP 4)

encuentra el valor menor

```
mixed min (mixed arg1, mixed arg2, mixed argn)
```

min() returns the numerically lowest of the parameter values.

Si el primer parámetro es un array, **min()** devuelve el menor valor en ese array. Si el primer parámetro es un entero, string o double, necesita al menos dos parámetros y **min()** devuelve el número menor de estos valores. Puede comparar un número ilimitado de valores.

Si uno o más de los valores es un double, todos los valores serán tratados como doubles, y devolverá un double. Si ninguno de los valores es un double, todos ellos serán tratados como enteros, y se devolverá un entero.

mt_rand (PHP 3>= 3.0.6, PHP 4)

genera un valor aleatorio mejorado

```
int mt_rand ([int min [, int max]])
```

Muchos generadores de números aleatorios de libcs antiguas tienen características dudosas o desconocidas y son lentas. Por defecto, PHP usa el generador de números aleatorios de libc con la función **rand()**. La función **mt_rand()** es un reemplazo para esta. Usa un generador de números aleatorios con características conocidas, el Tornado de Mersenne, que es capaz de producir números aleatorios que incluso se pueden emplear para propósitos criptográficos y es cuatro veces más rápido que la media de los que provee libc. La página principal del Tornado de Mersenne puede encontrarse en <http://www.math.keio.ac.jp/~matumoto/emt.html>, y una versión optimizada del código del TM esta disponible en <http://www.scp.syr.edu/~marc/hawk/twister.html>.

Si es llamada sin los parámetros opcionales min y max **mt_rand()** devuelve un valor pseudo-aleatorio entre 0 y RAND_MAX. Si quiere un número aleatorio entre 5 y 15 (inclusive), use **mt_rand(5,15)**.

Recuerde introducir la semilla del generador de números aleatorios antes de usar la instrucción con **mt_srand()**.

Vea también **mt_srand()**, **mt_getrandmax()**, **srand()**, **rand()** y **getrandmax()**.

mt_srand (PHP 3>= 3.0.6, PHP 4)

Introduce la semilla del generador de números aleatorios mejorado

```
void mt_srand (int seed)
```

Introduce la semilla *seed* en el generador de números aleatorios mejorado.

```
// seed son los microsegundos desde el último segundo "entero"  
mt_srand((double)microtime()*1000000);  
$randval = mt_rand();
```

Vea también **mt_rand()**, **mt_getrandmax()**, **srand()**, **rand()** y **getrandmax()**.

mt_getrandmax (PHP 3>= 3.0.6, PHP 4)

muestra el mayor valor aleatorio posible

```
int mt_getrandmax (void)
```

Devuelve el valor máximo que se puede obtener con una llamada a **mt_rand()**.

Vea también **mt_rand()**, **mt_srand()**, **rand()**, **srand()** y **getrandmax()**.

number_format (PHP 3, PHP 4)

formatea un número en grupos de miles

```
string number_format (float number, int decimals, string dec_point, string thousands_sep)
```

number_format() devuelve la versión formateada de *number*. Esta función acepta tanto uno, como dos o cuatro parámetros (tres no):

Si sólo se da un parámetro, *number* será formateado sin decimales, pero con una coma (",") entre cada grupo de miles.

Si se dan dos parámetros, *number* será formateado con *decimals* decimales con un punto (".") al principio, y una coma (",") entre cada grupo de miles.

Si se dan cuatro parámetros, *number* será formateado con *decimals* decimales, *dec_point* en vez del punto (".") antes de los decimales y *thousands_sep* en vez de la coma (",") entre cada grupo de miles.

OctDec (PHP 3, PHP 4)

octal a decimal

```
int octdec (string octal_string)
```

Devuelve el equivalente decimal del número octal representado por el argumento *octal_string*. OctDec convierte una cadena octal en un número decimal. El mayor número que puede ser convertido es 17777777777 o 2147483647 en decimal.

Vea también **decoc()**.

pi (PHP 3, PHP 4)

devuelve el valor de pi

```
double pi (void)
```

Devuelve una aproximación de pi.

pow (PHP 3, PHP 4)

expresión exponencial

```
float pow (float base, float exp)
```

Devuelve base elevado a la potencia de exp.

Vea también **exp()**.

rand (PHP 3, PHP 4)

genera un valor aleatorio

```
int rand ([int min [, int max]])
```

Si es llamada sin los argumentos opcionales min y max, rand() devuelve un valor pseudo-aleatorio entre 0 y RAND_MAX.
Si quiere un número aleatorio entre 5 y 15 (inclusive), por ejemplo, use rand(5,15).

Recuerde introducir la semilla del generador de números aleatorios antes de usar **srand()**.

Vea también **srand()**, **getrandmax()**, **mt_rand()**, **mt_srand()** y **mt_getrandmax()**.

round (PHP 3, PHP 4)

Redondea un float.

```
double round (double val)
```

Devuelve el valor redondeado de *val*.

```
$foo = round( 3.4 ); // $foo == 3.0
$foo = round( 3.5 ); // $foo == 4.0
$foo = round( 3.6 ); // $foo == 4.0
```

Vea también **ceil()** y **floor()**.

Sin (PHP 3, PHP 4)

seno

```
float sin (float arg)
```

Devuelve el seno de arg en radianes.

Vea también **cos()** y **tan()**.

Sqrt (PHP 3, PHP 4)

Raíz cuadrada

```
float sqrt (float arg)
```

Devuelve la raíz cuadrada de arg.

srand (PHP 3, PHP 4)

introduce la semilla del generador de números aleatorios

```
void srand (int seed)
```

Inicializa el generador de números aleatorios con *seed*.

```
// seed son los microsegundos desde el último segundo "entero"  
$rand((double)microtime()*1000000);  
$randval = rand();
```

Vea también **rand()**, **getrandmax()**, **mt_rand()**, **mt_srand()** y **mt_getrandmax()**.

Tan (PHP 3, PHP 4)

tangente

```
float tan (float arg)
```

Devuelve la tangente de arg en radianes.

Vea también **sin()** y **cos()**.

XXXIX. MCAL functions

MCAL stands for Modular Calendar Access Library.

Libmcal is a C library for accessing calendars. It's written to be very modular, with plugable drivers. MCAL is the calendar equivalent of the IMAP module for mailboxes.

With mcal support, a calendar stream can be opened much like the mailbox stream with the IMAP support. Calendars can be local file stores, remote ICAP servers, or other formats that are supported by the mcal library.

Calendar events can be pulled up, queried, and stored. There is also support for calendar triggers (alarms) and reoccurring events.

With libmcal, central calendar servers can be accessed and used, removing the need for any specific database or local file programming.

To get these functions to work, you have to compile PHP with `-with-mcal`. That requires the mcal library to be installed. Grab the latest version from <http://mcal.chek.com/> and compile and install it.

The following constants are defined when using the MCAL module: MCAL_SUNDAY, MCAL_MONDAY, MCAL_TUESDAY, MCAL_WEDNESDAY, MCAL_THURSDAY, MCAL_FRIDAY, MCAL_SATURDAY, MCAL_RECUR_NONE, MCAL_RECUR_DAILY, MCAL_RECUR_WEEKLY, MCAL_RECUR_MONTHLY_MDAY, MCAL_RECUR_MONTHLY_WDAY, MCAL_RECUR_YEARLY, MCAL_JANUARY, MCAL_FEBRUARY, MCAL_MARCH, MCAL_APRL, MCAL_MAY, MCAL_JUNE, MCAL_JULY, MCAL_AUGUST, MCAL_SEPTEMBER, MCAL_OCTOBER, MCAL_NOVEMBER, and MCAL_DECEMBER. Most of the functions use an internal event structure that is unique for each stream. This alleviates the need to pass around large objects between functions. There are convenience functions for setting, initializing, and retrieving the event structure values.

mcal_open (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Opens up an MCAL connection

```
int mcal_open (string calendar, string username, string password, string options)
```

Returns an MCAL stream on success, false on error.

mcal_open() opens up an MCAL connection to the specified *calendar* store. If the optional *options* is specified, passes the *options* to that mailbox also. The streams internal event structure is also initialized upon connection.

mcal_close (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Close an MCAL stream

```
int mcal_close (int mcal_stream, int flags)
```

Closes the given mcal stream.

mcal_fetch_event (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Fetches an event from the calendar stream

```
object mcal_fetch_event (int mcal_stream, int event_id [, int options])
```

mcal_fetch_event() fetches an event from the calendar stream specified by *id*.

Returns an event object consisting of:

- int id - ID of that event.
- int public - TRUE if the event is public, FALSE if it is private.
- string category - Category string of the event.
- string title - Title string of the event.
- string description - Description string of the event.
- int alarm - number of minutes before the event to send an alarm/reminder.
- object start - Object containing a datetime entry.
- object end - Object containing a datetime entry.
- int recur_type - recurrence type
- int recur_interval - recurrence interval
- datetime recur_enddate - recurrence end date
- int recur_data - recurrence data

All datetime entries consist of an object that contains:

- int year - year
- int month - month

- int mday - day of month
- int hour - hour
- int min - minutes
- int sec - seconds
- int alarm - minutes before event to send an alarm

mcal_list_events (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Return a list of events between two given datetimes

```
array mcal_list_events (int mcal_stream [, int begin_year [, int begin_month [, int begin_day [, int end_year [, int end_month [, int end_day]]]]]])
```

Returns an array of event ID's that are between the start and end dates, or if just a stream is given, uses the start and end dates in the global event structure.

mcal_list_events() function takes in an optional beginning date and an end date for a calendar stream. An array of event id's that are between the given dates or the internal event dates are returned.

mcal_append_event (PHP 4 >= 4.0RC1)

Store a new event into an MCAL calendar

```
int mcal_append_event (int mcal_stream)
```

mcal_append_event() Stores the global event into an MCAL calendar for the given stream.

Returns the uid of the newly inserted event.

mcal_store_event (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Modify an existing event in an MCAL calendar

```
int mcal_store_event (int mcal_stream)
```

mcal_store_event() Stores the modifications to the current global event for the given stream.

Returns true on success and false on error.

mcal_delete_event (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Delete an event from an MCAL calendar

```
int mcal_delete_event (int uid)
```

mcal_delete_event() deletes the calendar event specified by the uid.

Returns true.

mcal_snooze (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Turn off an alarm for an event

```
int mcal_snooze (int uid)
```

mcal_snooze() turns off an alarm for a calendar event specified by the uid.

Returns true.

mcal_list_alarms (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Return a list of events that has an alarm triggered at the given datetime

```
array mcal_list_events (int mcal_stream [, int begin_year [, int begin_month [, int begin_day [, int end_year [, int end_month [, int end_day]]]]]])
```

Returns an array of event ID's that has an alarm going off between the start and end dates, or if just a stream is given, uses the start and end dates in the global event structure.

mcal_list_events() function takes in an optional beginning date and an end date for a calendar stream. An array of event id's that are between the given dates or the internal event dates are returned.

mcal_event_init (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Initializes a streams global event structure

```
int mcal_event_init (int stream)
```

mcal_event_init() initializes a streams global event structure. this effectively sets all elements of the structure to 0, or the default settings.

Returns true.

mcal_event_set_category (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Sets the category of the streams global event structure

```
int mcal_event_set_category (int stream, string category)
```

mcal_event_set_category() sets the streams global event structure's category to the given string.

Returns true.

mcal_event_set_title (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Sets the title of the streams global event structure

```
int mcal_event_set_title (int stream, string title)
```

mcal_event_set_title() sets the streams global event structure's title to the given string.

Returns true.

mcal_event_set_description (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Sets the description of the streams global event structure

```
int mcal_event_set_description (int stream, string description)
```

mcal_event_set_description() sets the streams global event structure's description to the given string.

Returns true.

mcal_event_set_start (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Sets the start date and time of the streams global event structure

```
int mcal_event_set_start (int stream, int year, int month [, int day [, int hour [, int min [, int sec]]]])
```

mcal_event_set_start() sets the streams global event structure's start date and time to the given values.

Returns true.

mcal_event_set_end (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Sets the end date and time of the streams global event structure

```
int mcal_event_set_end (int stream, int year, int month [, int day [, int hour [, int min [, int sec]]]])
```

mcal_event_set_end() sets the streams global event structure's end date and time to the given values.

Returns true.

mcal_event_set_alarm (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Sets the alarm of the streams global event structure

```
int mcal_event_set_alarm (int stream, int alarm)
```

mcal_event_set_alarm() sets the streams global event structure's alarm to the given minutes before the event.

Returns true.

mcal_event_set_class (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Sets the class of the streams global event structure

```
int mcal_event_set_class (int stream, int class)
```

mcal_event_set_class() sets the streams global event structure's class to the given value. The class is either 1 for public, or 0 for private.

Returns true.

mcal_is_leap_year (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Returns if the given year is a leap year or not

```
int mcal_is_leap_year (int year)
```

mcal_is_leap_year() returns 1 if the given year is a leap year, 1 if not.

mcal_days_in_month (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Returns the number of days in the given month

```
int mcal_days_in_month (int month, int leap year)
```

mcal_days_in_month() Returns the number of days in the given month, taking into account if the given year is a leap year or not.

mcal_date_valid (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Returns true if the given year, month, day is a valid date

```
int mcal_date_valid (int year, int month, int day)
```

mcal_date_valid() Returns true if the given year, month and day is a valid date, false if not.

mcal_time_valid (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Returns true if the given year, month, day is a valid time

```
int mcal_time_valid (int hour, int minutes, int seconds)
```

mcal_time_valid() Returns true if the given hour, minutes and seconds is a valid time, false if not.

mcal_day_of_week (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Returns the day of the week of the given date

```
int mcal_ (int year, int month, int day)
```

mcal_day_of_week() returns the day of the week of the given date

mcal_day_of_year (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Returns the day of the year of the given date

```
int mcal_ (int year, int month, int day)
```

mcal_day_of_year() returns the day of the year of the given date

mcal_date_compare (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Compares two dates

```
int mcal_date_compare (int a_year, int a_month, int a_day, int b_year, int b_month, int b_day)
```

mcal_date_compare() Compares the two given dates, returns <0, 0, >0 if a<b, a==b, a>b respectively

mcal_next_recurrence (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Returns the next recurrence of the event

```
int mcal_next_recurrence (int stream, int weekstart, array next)
```

mcal_next_recurrence() returns an object filled with the next date the event occurs, on or after the supplied date. Returns empty date field if event does not occur or something is invalid. Uses weekstart to determine what day is considered the beginning of the week.

mcal_event_set_recur_none (PHP 3>= 3.0.15, PHP 4 >= 4.0RC1)

Sets the recurrence of the streams global event structure

```
int mcal_event_set_recur_none (int stream)
```

mcal_event_set_recur_none() sets the streams global event structure to not recur (event->recur_type is set to MCAL_RECUR_NONE).

mcal_event_set_recur_daily (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Sets the recurrence of the streams global event structure

```
int mcal_event_set_recur_daily (int stream, int year, int month, int day, int interval)
```

mcal_event_set_recur_daily() sets the streams global event structure's recurrence to the given value to be reoccurring on a daily basis, ending at the given date.

mcal_event_set_recur_weekly (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Sets the recurrence of the streams global event structure

```
int mcal_event_set_recur_weekly (int stream, int year, int month, int day, int interval,
int weekdays)
```

mcal_event_set_recur_weekly() sets the streams global event structure's recurrence to the given value to be reoccurring on a weekly basis, ending at the given date.

mcal_event_set_recur_monthly_mday (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Sets the recurrence of the streams global event structure

```
int mcal_event_set_recur_monthly_mday (int stream, int year, int month, int day, int
interval)
```

mcal_event_set_recur_monthly_mday() sets the streams global event structure's recurrence to the given value to be reoccurring on a monthly by month day basis, ending at the given date.

mcal_event_set_recur_monthly_wday (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Sets the recurrence of the streams global event structure

```
int mcal_event_set_recur_monthly_wday (int stream, int year, int month, int day, int
interval)
```

mcal_event_set_recur_monthly_wday() sets the streams global event structure's recurrence to the given value to be reoccurring on a monthly by week basis, ending at the given date.

mcal_event_set_recur_yearly (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Sets the recurrence of the streams global event structure

```
int mcal_event_set_recur_yearly (int stream, int year, int month, int day, int interval)
```

mcal_event_set_recur_yearly() sets the streams global event structure's recurrence to the given value to be reoccurring on a yearly basis, ending at the given date .

mcal_fetch_current_stream_event (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Returns an object containing the current streams event structure

```
int mcal_fetch_current_stream_event (int stream)
```

mcal_event_fetch_current_stream_event() returns the current stream's event structure as an object containing:

- int id - ID of that event.
- int public - TRUE if the event is public, FALSE if it is private.
- string category - Category string of the event.
- string title - Title string of the event.
- string description - Description string of the event.
- int alarm - number of minutes before the event to send an alarm/reminder.
- object start - Object containing a datetime entry.
- object end - Object containing a datetime entry.
- int recur_type - recurrence type
- int recur_interval - recurrence interval
- datetime recur_enddate - recurrence end date
- int recur_data - recurrence data

All datetime entries consist of an object that contains:

- int year - year
- int month - month
- int mday - day of month
- int hour - hour
- int min - minutes
- int sec - seconds
- int alarm - minutes before event to send an alarm

mcal_event_add_attribute (PHP 3>= 3.0.15, PHP 4 >= 4.0RC1)

Adds an attribute and a value to the streams global event structure

```
void mcal_event_add_attribute (int stream, string attribute, string value)
```

mcal_event_add_attribute() adds an attribute to the stream's global event structure with the value given by "value".

XL. Funciones Criptográficas

Estas funciones trabajan usando mcrypt (<ftp://argeas.cs-net.gr/pub/unix/mcrypt/>).

Esta es una interfaz a la librería mcrypt, que soporta una gran variedad de algoritmos de bloque como DES, TripleDES, Blowfish (por defecto), 3-WAY, SAFER-SK64, SAFER-SK128, TWOFISH, TEA, RC2 y GOST en los modos de cifrado CBC, OFB, CFB y ECB. Adicionalmente, soporta RC6 e IDEA que se consideran "no-libres".

Para usarlos, descarga libmcrypt-x.x.tar.gz de aquí (<ftp://argeas.cs-net.gr/pub/unix/mcrypt/>) y sigue las instrucciones de instalación incluidas. Necesitas compilar PHP con el parámetro `-with-mcrypt` para activar esta extensión.

mcrypt puede usarse para encriptar y desencriptar usando los cifrados mencionados arriba. Los cuatro comandos importantes de mcrypt (`mcrypt_cfb()`, `mcrypt_cbc()`, `mcrypt_ecb()`, y `mcrypt_ofb()`) pueden operar en ambos modos que se llaman MCRYPT_ENCRYPT y MCRYPT_DECRYPT, respectivamente.

Ejemplo 1. Encripta un valor de entrada con TripleDES en modo ECB

```
<?php
$key = "esta es una clave muy secreta";
$input = "Nos vemos a las 9 en punto en el lugar secreto.";

$encrypted_data = mcrypt_ecb(MCRYPT_TripleDES, $key, $input, MCRYPT_ENCRYPT);
?>
```

Este ejemplo devolverá los datos encriptados como una cadena en `$encrypted_data`.

mcrypt puede operar en cuatro modos de cifrado (CBC, OFB, CFB y ECB). Perfilaremos el uso normal de cada uno de estos modos. Para una mejor referencia y una discusión más completa ver Applied Cryptography by Schneier (ISBN 0-471-11709-9).

- ECB (electronic codebook o libro de códigos electrónico) va bien para datos aleatorios, tales como encriptar otras claves. Puesto que los datos son cortos y aleatorios, las desventajas de ECB tienen un efecto negativo favorable.
- CBC (cipher block chaining o cifrado en bloque encadenado) es especialmente útil para encriptar ficheros, donde incrementa significativamente la seguridad por encima de ECB.
- CFB (cipher feedback o cifrado realimentado) es el mejor modo de encriptar flujos de bytes donde cada byte debe ser encriptado.
- OFB (output feedback o salida realimentada) es comparable al CFB, pero puede usarse en aplicaciones donde la propagación de errores no puede tolerarse.

Actualmente PHP no soporta el encriptado/desencriptado de flujos de bits. Por ahora, sólo soporta el manejo de cadenas.

Para una lista completa de los cifrados soportados, ver las definiciones al final de `mcrypt.h`. La regla general es que se puede acceder al cifrado desde PHP con `MCRYPT_nombredelcifrado`.

Aquí hay una pequeña lista de los cifrados que están soportados actualmente por la extensión mcrypt. Si un cifrado no está listado aquí, pero está listado por mcrypt como soportado, puedes asumir con seguridad que ésta documentación está caduca.

- MCRYPT_BLOWFISH
- MCRYPT_DES
- MCRYPT_TripleDES
- MCRYPT_ThreeWAY
- MCRYPT_GOST
- MCRYPT_CRYPT

- MCRYPT DES_COMPAT
- MCRYPT_SAFER64
- MCRYPT_SAFER128
- MCRYPT_CAST128
- MCRYPT_TEAN
- MCRYPT_RC2
- MCRYPT_TWOFISH (para las antiguas versiones mcrypt 2.x)
- MCRYPT_TWOFISH128 (TWOFISHxxx está disponible en las versiones más nuevas 2.x)
- MCRYPT_TWOFISH192
- MCRYPT_TWOFISH256
- MCRYPT_RC6
- MCRYPT_IDEA

Debes (en los modos CFB y OFB) o puedes (en el modo CBC) suministrar un vector de inicialización (IV) a la correspondiente función de cifrado. El IV debe ser único y debe ser el mismo cuando desencriptas o encriptas. Con datos que son guardados encriptados, puedes cojer la salida de una función de índice bajo la cual los datos son almacenados (ej. la clave MD5 de un fichero). Alternativamente, puedes transmitir el IV junto con los datos encriptados (ver capítulo 9.3 de Applied Cryptography by Schneier (ISBN 0-471-11709-9) para una discusión de éste asunto).

mcrypt_get_cipher_name (PHP 3>= 3.0.8, PHP 4)

Obtiene el nombre del cifrado especificado

```
string mcrypt_get_cipher_name ( int cipher )
```

mcrypt_get_cipher_name() se usa para obtener el nombre del cifrado especificado.

mcrypt_get_cipher_name() toma como argumento el número de cifrado y devuelve el nombre del cifrado o false, si el cifrado no existe.

Ejemplo 1. Ejemplo de mcrypt_get_cipher_name

```
<?php
$cipher = MCRYPT_TripleDES;

print mcrypt_get_cipher_name($cipher);
?>
```

El ejemplo de más arriba da como resultado:

TripleDES

mcrypt_get_block_size (PHP 3>= 3.0.8, PHP 4)

Obtiene el tamaño de bloque del cifrado indicado

```
int mcrypt_get_block_size ( int cipher )
```

mcrypt_get_block_size() se usa para obtener el tamaño de bloque del cifrado indicado en *cipher*.

mcrypt_get_block_size() toma un argumento, el cifrado *cipher* y devuelve el tamaño en bytes.

Ver también: **mcrypt_get_key_size()**

mcrypt_get_key_size (PHP 3>= 3.0.8, PHP 4)

Obtiene el tamaño de la clave de un cifrado

```
int mcrypt_get_key_size ( int cipher )
```

mcrypt_get_key_size() se usa para obtener el tamaño de la clave del cifrado indicado en *cipher*.

mcrypt_get_key_size() toma un argumento, el cifrado *cipher* y devuelve el tamaño de la clave en bytes.

Ver también: **mcrypt_get_block_size()**

mcrypt_create_iv (PHP 3>= 3.0.8, PHP 4)

Crea un vector de inicialización (IV) a partir de una fuente aleatoria

```
string mcrypt_create_iv (int size, int source)
```

mcrypt_create_iv() se usa para crear un IV.

mcrypt_create_iv() toma dos argumentos, *size* determina el tamaño del IV, *source* especifica la fuente del IV.

La fuente puede ser MCRYPT_RAND (generador de números aleatorios del sistema), MCRYPT_DEV_RANDOM (que lee datos de /dev/random) y MCRYPT_DEV_URANDOM (que lee datos de /dev/urandom). Si usas MCRYPT_RAND, asegurate de llamar antes a *srand()* para inicializar el generador de números aleatorios.

Ejemplo 1. Ejemplo de mcrypt_create_iv

```
<?php
$cipher = MCRYPT_TripleDES;
$block_size = mcrypt_get_block_size($cipher);
$iv = mcrypt_create_iv($block_size, MCRYPT_DEV_RANDOM);
?>
```

mcrypt_cbc (PHP 3>= 3.0.8, PHP 4)

Encripta/desencripta datos en modo CBC

```
int mcrypt_cbc (int cipher, string key, string data, int mode [, string iv])
```

mcrypt_cbc() encripta o desencripta (dependiendo de *mode*) los datos *data* con el cifrado *cipher* y la clave *key* en el modo de cifrado CBC y devuelve la cadena resultante.

El parámetro *cipher* es una de las constantes con nombre MCRYPT_nombrecifrado.

key es la clave suministrada al algoritmo. Debe guardarse en secreto.

data son los datos que serán encriptados/desencriptados.

mode es MCRYPT_ENCRYPT o MCRYPT_DECRYPT.

iv es el vector de inicialización opcional.

Ver también: **mcrypt_cfb()**, **mcrypt_ecb()**, **mcrypt_ofb()**

mcrypt_cfb (PHP 3>= 3.0.8, PHP 4)

Encripta/desencripta datos en modo CFB

```
int mcrypt_cfb (int cipher, string key, string data, int mode, string iv)
```

mcrypt_cfb() encripta o desencripta (dependiendo de *mode*) los datos *data* con el cifrado *cipher* y la clave *key* en el modo de cifrado CFB y devuelve la cadena resultante.

El parámetro *cipher* es una de las constantes con nombre MCRYPT_nombrecifrado.

key es la clave suministrada al algoritmo. Debe guardarse en secreto.

data son los datos que serán encriptados/desencriptados.

mode es MCRYPT_ENCRYPT o MCRYPT_DECRYPT.

iv es el vector de inicialización.

Ver también: **mcrypt_cbc()**, **mcrypt_ecb()**, **mcrypt_ofb()**

mcrypt_ecb (PHP 3>= 3.0.8, PHP 4)

Encripta/desencripta datos en modo ECB

```
int mcrypt_ecb ( int cipher, string key, string data, int mode)
```

mcrypt_ecb() encripta o desencripta (dependiendo de *mode*) los datos *data* con el cifrado *cipher* y la clave *key* en el modo de cifrado ECB y devuelve la cadena resultante.

El parámetro *cipher* es una de las constantes con nombre MCRYPT_nombrecifrado.

key es la clave suministrada al algoritmo. Debe guardarse en secreto.

data son los datos que serán encriptados/desencriptados.

mode es MCRYPT_ENCRYPT o MCRYPT_DECRYPT.

Ver también: **mcrypt_cbc()**, **mcrypt_cfb()**, **mcrypt_ofb()**

mcrypt_ofb (PHP 3>= 3.0.8, PHP 4)

Encripta/desencripta datos en modo OFB

```
int mcrypt_ofb ( int cipher, string key, string data, int mode, string iv)
```

mcrypt_ofb() encripta o desencripta (dependiendo de *mode*) los datos *data* con el cifrado *cipher* y la clave *key* en el modo de cifrado OFB y devuelve la cadena resultante.

El parámetro *cipher* es una de las constantes con nombre MCRYPT_nombrecifrado.

key es la clave suministrada al algoritmo. Debe guardarse en secreto.

data son los datos que serán encriptados/desencriptados.

mode es MCRYPT_ENCRYPT o MCRYPT_DECRYPT.

iv es el vector de inicialización.

Ver también: **mcrypt_cbc()**, **mcrypt_cfb()**, **mcrypt_ecb()**

XLI. Funciones Hash

Estas funciones han sido realizadas para trabajar con mhash (<http://mhash.sourceforge.net/>).

Esta es una interfaz con la librería mhash. mhash soporta una amplia variedad de algoritmos hash como MD5, SHA1, GOST, y muchos otros.

Para usarla, hay que descargar la distribución desde su sitio web (<http://mhash.sourceforge.net/>) y seguir las instrucciones de instalación. Se necesita compilar PHP con el parámetro `-with-mhash` para activar esta extensión.

mhash puede ser usado para crear checksums, message digests, y más.

Ejemplo 1. Generar una clave SHA1 e imprimirla en hexadecimal

```
<?php  
$input = "Let us meet at 9 o' clock at the secret place.";  
$hash = mhash(MHASH_SHA1, $input);  
  
print "The hash is ".bin2hex($hash)."\n";  
  
?>
```

Esto generará:

```
The hash is d3b85d710d8f6e4e5efd4d5e67d041f9cecedafe
```

Para una lista completa de hash soportados, véase la documentación de mhash. La regla general es que se puede acceder a los algoritmos hash desde PHP con `MHASH_HASHNAME`. Como ejemplo, para acceder a HAVAL se debe usar la constante de PHP llamada `MHASH_HAVAL`.

Aquí hay una lista de hashes que están actualmente soportados por mhash. Si un hash no está en dicha lista pero aparece como soportado por mhash, entonces se asume con plena seguridad que esta documentación está desfasada.

- `MHASH_MD5`
- `MHASH_SHA1`
- `MHASH_HAVAL`
- `MHASH_RIPEMD160`
- `MHASH_RIPEMD128`
- `MHASH_SNEFRU`
- `MHASH_TIGER`
- `MHASH_GOST`
- `MHASH_CRC32`
- `MHASH_CRC32B`

mhash_get_hash_name (PHP 3>= 3.0.9, PHP 4)

Conseguir el nombre de un hash especifico

```
string mhash_get_hash_name (int hash)
```

mhash_get_hash_name() es usado para conseguir el nombre de el hash determinado.

mhash_get_hash_name() toma el id del hash como un argumento y devuelve el nombre de el hash o false, si el hash no existe.

Ejemplo 1. mhash_get_hash_name example

```
<?php
$hash = MHASH_MD5;

print mhash_get_hash_name($hash);
?>
```

El ejemplo anterior mostrara:

MD5

mhash_get_block_size (PHP 3>= 3.0.9, PHP 4)

Conseguir el tamaño de bloque de el hash especificado

```
int mhash_get_block_size (int hash)
```

mhash_get_block_size() es usado para obtener el tamaño de un bloque de el *hash* determinado.

mhash_get_block_size() toma un argumento, el *hash* y devuelve el tamaño en bytes o false, si el *hash* no existe.

mhash_count (PHP 3>= 3.0.9, PHP 4)

Obtener el valor mayor del id hash disponible

```
int mhash_count (void)
```

mhash_count() devuelve el valor mas alto id hash disponible. Los hash estan numerados desde 0 hasta este valor.

Ejemplo 1. Recorriendo todos los hash

```
<?php
$nr = mhash_count();

for($i = 0; $i <= $nr; $i++) {
    echo sprintf("The blocksize of %s is %d\n",
        mhash_get_hash_name($i),
```

```
    mhash_get_block_size($i));  
}  
?>
```

mhash (PHP 3>= 3.0.9, PHP 4)

Calcular el hash

```
string mhash (int hash, string data)
```

mhash() aplica una funcion hash especificada por *hash* a *data* y devuelve el valor hash resultante (tambien llamdo digest).

XLII. Funciones de Microsoft SQL Server

mssql_close (PHP 3, PHP 4)

cierra una conexión con MS SQL Server

```
int mssql_close (int link_identifier)
```

Devuelve: true si se finaliza con éxito, false si se produce un error

`mssql_close()` cierra la conexión con una base de datos MS SQL Server que está asociada al identificador especificado. Si el identificador no se especifica, se asume la última conexión abierta.

Observe que normalmente esto no es necesario, ya que las conexiones no-persistentes abiertas se cierran automáticamente en cuanto finaliza el script.

`mssql_close()` no cerrará conexiones persistentes generadas por `mssql_pconnect()`.

Ver también: **mssql_connect()**, **mssql_pconnect()**.

mssql_connect (PHP 3, PHP 4)

abre una conexión con MS SQL server

```
int mssql_connect (string servername, string username, string password)
```

Devuelve: Un identificador de MSSQL si se ejecuta correctamente, o false si se produce un error.

`mssql_connect()` establece una conexión con MS SQL server. El argumento `servername` debe ser un nombre de servidor válido, que está definido en el fichero 'interfaces'.

En caso de hacer una segunda llamada a `mssql_connect()` con los mismos argumentos, no se establecerá una nueva conexión, sino que se devolverá el identificador de la conexión establecida anteriormente.

La conexión con el servidor se cerrará tan pronto como finalice el script, a menos que se cierre antes, mediante una llamada explícita a la función `mssql_close()`.

Ver también **mssql_pconnect()**, **mssql_close()**.

mssql_data_seek (PHP 3, PHP 4)

move el puntero interno de las filas

```
int mssql_data_seek (int result_identifier, int row_number)
```

Devuelve: true si se ejecuta con éxito, false si falla.

`mssql_data_seek()` move el puntero interno de la consulta MS SQL asociada al `result_identifier` especificado, para que apunte al número de fila especificada. La siguiente llamada a `mssql_fetch_row()` devolverá esa fila.

Ver también: **mssql_data_seek()**.

mssql_fetch_array (PHP 3, PHP 4)

Captura la fila en un array

```
int mssql_fetch_array (int result)
```

Devuelve: Un array que corresponde a la fila capturada, o false si no hay más filas.

mssql_fetch_array() es una versión extendida de **mssql_fetch_row()**. Añade el almacenar los datos en los índices numéricos del array resultante, también almacena los datos en índices asociativos, usando los nombres de los campos como claves.

Una observación a tener en cuenta es, que usar **mssql_fetch_array()** NO es más lento que usar **mssql_fetch_row()**, mientras que esta provee un valor añadido significativo.

Para más detalles, ver también **mssql_fetch_row()**

mssql_fetch_field (PHP 3, PHP 4)

obtiene la información de los campos

```
object mssql_fetch_field ( int result, int field_offset )
```

Devuelve un objeto que contiene información de los campos.

mssql_fetch_field() se puede usar para obtener información acerca de los campos pertenecientes al resultado de una consulta. Si el parámetro *field_offset* no es especificado, se devuelve la información del siguiente campo que todavía no ha sido devuelto por **mssql_fetch_field()**.

Las propiedades de este objeto son:

- name - nombre de la columna. si la columna es el resultado de una función, esta propiedad vale #N, donde #N es un número de serie.
- column_source - la tabla de donde se tomó la columna
- max_length - longitud máxima de columna
- numeric - 1 si la columna es numérica

Ver también **mssql_field_seek()**

mssql_fetch_object (PHP 3, PHP 4)

captura la fila como un objeto

```
int mssql_fetch_object ( int result )
```

Devuelve: Un objeto con propiedades que se corresponden con la fila capturada, o false si no hay más filas.

mssql_fetch_object() es parecida a **mssql_fetch_array()**, con una diferencia - devuelve un objeto en vez de un array. Indirectamente, esto significa que sólo se puede acceder a los datos por el nombre de los campos, y no por sus posiciones en el objeto (los números no son nombres de propiedades válidas).

La función es idéntica a **mssql_fetch_array()**, y casi tan rápida como **mssql_fetch_row()** (la diferencia es insignificante).

Ver también: **mssql_fetch-array()** and **mssql_fetch-row()**.

mssql_fetch_row (PHP 3, PHP 4)

obtiene la fila como un array numerado

```
array mssql_fetch_row (int result)
```

Devuleve: Un array que corresponde a la fila capturada, o false si no hay más filas.

`mssql_fetch_row()` captura una fila de datos pertenecientes al resultado asociado con el identificador de resultado especificado. La fila es devuelta como un array. Cada columna de resultados es almacenada en una posición del array, comenzando en la posición 0.

Siguientes llamadas a `mssql_fetch_rows()` devolverían las filas siguientes del result set, o false si no hay mas filas.

Ver también: **[mssql_fetch_array\(\)](#)**, **[mssql_fetch_object\(\)](#)**, **[mssql_data_seek\(\)](#)**, **[mssql_fetch_lengths\(\)](#)**, and **[mssql_result\(\)](#)**.

mssql_field_seek (PHP 3, PHP 4)

set field offset

```
int mssql_field_seek (int result, int field_offset)
```

Se posiciona en el campo especificado por el parámetro `field_offset`. Si la siguiente llamada a `mssql_fetch_field()` no incluye el parámetro `field_offset`, lo que devuelve la función es el campo.

Ver también: **[mssql_fetch_field\(\)](#)**.

mssql_free_result (PHP 3, PHP 4)

libera de la memoria el resultado de una consulta

```
int mssql_free_result (int result)
```

`mssql_free_result()` sólo se necesita llamarla si le preocupa el estar usando mucha memoria mientras se está ejecutando el script. Toda el resultado en memoria será liberado automaticamente cuando finalice el script, puede llamar a `mssql_free_result()` con el identificador de la consulta como argumento y la consulta asociada será liberada de la memoria.

mssql_num_fields (PHP 3, PHP 4)

obtiene el número de campos de la consulta

```
int mssql_num_fields (int result)
```

`mssql_num_fields()` devuelve el número de campos de la consulta o result set.

Ver también: **[mssql_db_query\(\)](#)**, **[mssql_query\(\)](#)**, **[mssql_fetch_field\(\)](#)**, **[mssql_num_rows\(\)](#)**.

mssql_num_rows (PHP 3, PHP 4)

obtiene el número de filas de la consulta

```
int mssql_num_rows (string result)
```

mssql_num_rows() devuelve el número de filas de la consulta o result set.

Ver también: **mssql_db_query()**, **mssql_query()** and, **mssql_fetch_row()**.

mssql_pconnect (PHP 3, PHP 4)

abre una conexión persistente con MS SQL

```
int mssql_pconnect (string servername, string username, string password)
```

Devuelve: Un identificador persistente positivo si no hay error, o false si se produce alguno

mssql_pconnect() funciona de la misma forma que **mssql_connect()** aunque con dos grandes diferencias.

La primera es que cuando intenta conectar, la función intentará encontrar un enlace (persistente) que ya esté abierto en el mismo ordenador, nombre de usuario y contraseña. Si lo encuentra, la función devolverá el identificador de esta en vez de abrir una nueva conexión.

Y la segunda, la conexión con el servidor no se cerrará cuando finalice la ejecución del script. En vez de esto, el enlace permanecerá abierto para un uso futuro. (**mssql_close()** no cerrará enlaces establecidos por **mssql_pconnect()**).

Por consiguiente, este tipo de enlace es llamado 'persistente'.

mssql_query (PHP 3, PHP 4)

envia una consulta MS SQL

```
int mssql_query (string query, int link_identifier)
```

Devuelve: Un identificado de resultado valido si no hay error, o false en caso contrario.

mssql_query() envia una petición de consulta a la base de datos activa en el servidor asociada al identificador de enlace especificado. Si el identificador del enlace no es especificado, se asume como abierto el último enlace. Si no hay ningún enlace abierto, la función intenta establecer un enlace como si **mssql_connect()** hubiera sido llamada, y lo usa.

Ver también: **mssql_db_query()**, **mssql_select_db()**, and **mssql_connect()**.

mssql_result (PHP 3, PHP 4)

get result data

```
int mssql_result (int result, int i, mixed field)
```

Devuelve: El contenido de la celda en la fila y posición del result set especificado.

mssql_result() devuelve el contenido de una celda del result set. El parametro field puede ser la posición del campo, o el nombre del campo o bien nombretabla.nombrecampo. Si el nombre de la columna ha sido renombrado ('select foo as bar from...'), use el alias en vez del nombre de la columna.

Trabajando con result sets de gran tamaño, debería considerar el uso de una de las funciones que capturan una fila completa (especificadas abajo). Como estas funciones devuelven el contenido de múltiples celdas en una sola llamada, estas son MUCHO más rápidas que **mssql_result()**. También, observe que especificar una posición numérica para el

argumento field es mucho mas rápido que especificar el nombre de un campo o utilizar la forma nombretabla.nombrecampo como argumento.

Alternativas recomendadas para mayor rendimiento : **mssql_fetch_row()**, **mssql_fetch_array()**, y **mssql_fetch_object()**.

mssql_select_db (PHP 3, PHP 4)

selecciona una base de datos MS SQL

```
int mssql_select_db (string database_name, int link_identifier)
```

Devuelve: true si todo va bien, false si se produce un error

mssql_select_db() selecciona como base de datos activa del servidor, la que está asociada al identificador de enlace especificado. Si no se especifica ningún identificador, se asume el último enlace. Si no hay ningún enlace abierto, la función intentará establecer un enlace como si se llamara a la función **mssql_connect()**, y lo usa.

Cada llamada a **mssql_query()** será realizada sobre la base de datos activa.

Ver también: **mssql_connect()**, **mssql_pconnect()**, y **mssql_query()**

XLIII. Miscelánea de funciones

Estas funciones están colocadas aquí debido a que no parecen ajustarse a ninguna otra categoría.

connection_aborted (PHP 3>= 3.0.7, PHP 4 >= 4.0b4)

Devuelve true si el cliente está desconectado

```
int connection_aborted (void )
```

Devuelve true si el cliente está desconectado. Vea la descripción de la [Gestión de la Conexión](#) en el capítulo [Características](#) para una explicación completa.

connection_status (PHP 3>= 3.0.7, PHP 4 >= 4.0b4)

Devuelve el estado de la conexión en un campo de bits

```
int connection_status (void )
```

Devuelve el estado de la conexión en un campo de bits. Vea la descripción de la [Gestión de la Conexión](#) en el capítulo [Características](#) para una explicación completa.

connection_timeout (PHP 3>= 3.0.7, PHP 4 >= 4.0b4)

Devuelve true si el script ha alcanzado su time out

```
int connection_timeout (void )
```

Devuelve true si el script ha alcanzado su time out. Vea la descripción de la [Gestión de la Conexión](#) en el capítulo [Características](#) para una explicación completa.

define (PHP 3, PHP 4)

Define una constante con nombre.

```
int define (string name, mixed value [, int case_insensitive])
```

Define una constante con nombre, que es similar a una variable, excepto que:

- Las constantes no tienen un símbolo dólar '\$' precediéndolas;
- Las constantes son accesibles desde cualquier lugar sin tener en cuenta las reglas de ámbito de las variables.
- Las constantes no pueden ser redefinidas o iniciadas una vez que han sido establecidas, y
- Las constantes sólo pueden evaluar valores escalares

El nombre de la constante se da en *name* (nombre); el valor se da en *value* (valor).

El tercer parámetro opcional *case_insensitive* también se encuentra disponible. Si se da el valor 1, la constante se definirá no distinguiendo mayúsculas/minusculas. El comportamiento por defecto es si distinguir; i.e. CONSTANT y Constant representan valores diferentes.

Ejemplo 1. Definición de Constantes

```
<?php
define("CONSTANT", "Hello world.");
echo CONSTANT; // outputs "Hello world."
?>
```

define() devuelve TRUE en caso de éxito y FALSE si ocurre un error.

Véase también **defined()** y la sección [Constantes](#).

defined (PHP 3, PHP 4)

Comprueba que una constante con nombre dada existe.

```
int defined (string name)
```

Devuelve TRUE si la constante con nombre dada en *name* (nombre) ha sido definida, false en otro caso.

Véase también **define()** y la sección [Constantes](#).

die (unknown)

Envía a la salida un mensaje y finaliza el script actual

```
void die (string message)
```

Esta construcción del lenguaje envía a la salida un mensaje y finaliza la ejecución del script. No devuelve nada.

Ejemplo 1. Ejemplo die

```
<?php
$filename = '/path/to/data-file';
$file = fopen($filename, 'r')
    or die "unable to open file ($filename)";
?>
```

eval (unknown)

Evalúa una cadena de caracteres como código PHP

```
void eval (string code_str)
```

eval() evalúa la cadena de caracteres dada en *code_str* como código PHP. Entre otras cosas, ésto puede ser útil para almacenar código en un campo de texto de base de datos para una ejecución posterior.

Hay algunos aspectos a tener en cuenta cuando se utiliza **eval()**. Recuerde que la cadena de caracteres pasada debe ser código PHP válido, incluyendo aspectos como sentencias de terminación con un punto y coma para que el parser no finalice en la línea después de **eval()**, y secuencias de formato correctas en *code_str*.

Recuerde también que las variables a las que se les da valor en **eval()** retendrán estos valores posteriormente en el script principal.

Ejemplo 1. Ejemplo eval() - fusión en un único texto

```
<?php
$string = 'cup';
$name = 'coffee';
$str = 'This is a $string with my $name in it.<br>';
echo $str;
eval( "\$str = \"$str\";" );
echo $str;
?>
```

El ejemplo anterior mostrará:

```
This is a $string with my $name in it.
This is a cup with my coffee in it.
```

exit (unknown)

Finaliza el script actual

```
void exit(void);
```

Esta construcción del lenguaje finaliza la ejecución del script. No devuelve nada.

get_browser (PHP 3, PHP 4)

Informa sobre lo que es capaz de hacer el navegador (browser) del usuario.

```
object get_browser ([string user_agent])
```

get_browser() intenta determinar las características del navegador del usuario. Para ello consulta el fichero de información del navegador, *browscap.ini*. Por defecto, se utiliza el valor de *\$HTTP_USER_AGENT*; sin embargo, puede alterar ésto (i.e., consultando otra información del navegador) pasando el parámetro opcional *user_agent* a **get_browser()**.

La información se devuelve en un objeto, que contendrá varios elementos de datos que representan, por ejemplo, los números de versión (mayor y menor) del navegador y la cadena ID; valores true/false para características como los marcos, JavaScript, y cookies; etc.

browscap.ini contiene información de muchos navegadores, depende de las actualizaciones del usuario para mantener la base de datos actualizada. El formato del fichero es claramente auto-explicativo.

El ejemplo siguiente muestra como se puede listar toda la información disponible recuperada del navegador del usuario.

Ejemplo 1. ejemplo get_browser()

```
<?php
function list_array( $array ) {
    while ( list( $key, $value ) = each( $array ) ) {
```

```

    $str .= "<b>$key:</b> $value<br>\n";
}
return $str;
}
echo "$HTTP_USER_AGENT<hr>\n";
$browser = get_browser();
echo list_array( array ) $browser );
?>

```

La salida del script anterior debería paracerse a ésto:

```

Mozilla/4.5 [en] (X11; U; Linux 2.2.9 i586)<hr>
<b>browser_name_pattern:</b> Mozilla/4\5.*<br>
<b>parent:</b> Netscape 4.0<br>
<b>platform:</b> Unknown<br>
<b>majorver:</b> 4<br>
<b>minorver:</b> 5<br>
<b>browser:</b> Netscape<br>
<b>version:</b> 4<br>
<b>frames:</b> 1<br>
<b>tables:</b> 1<br>
<b>cookies:</b> 1<br>
<b>backgroundsounds:</b> <br>
<b>vbscript:</b> <br>
<b>javascript:</b> 1<br>
<b>javaapplets:</b> 1<br>
<b>activexcontrols:</b> <br>
<b>beta:</b> <br>
<b>crawler:</b> <br>
<b>authenticodeupdate:</b> <br>
<b>msn:</b> <br>

```

Para conseguir ésto, su opción de fichero de configuración [browscap](#) debe apuntar a la correcta localización del fichero `browscap.ini`.

Para más información (incluyendo localizaciones desde las que puede obtener un fichero `browscap.ini`), consulte las FAQ sobre PHP en <http://www.php.net/FAQ.html> (<http://www.php.net/FAQ.php>).

Nota: el soporte para browscap fue añadido en la versión 3.0b2 de PHP.

ignore_user_abort (PHP 3>= 3.0.7, PHP 4 >= 4.0b4)

Establece si la desconexión de un cliente debe suspender la ejecución del script

```
int ignore_user_abort ([int setting])
```

Esta función establece si la desconexión de un cliente debe provocar la suspensión del script. Devolverá el valor previo y puede ser llamada sin argumentos para devolver el valor actual y no cambiarlo. Véase la sección sobre la Gestión de la Conexión en el capítulo Características para una descripción completa de la gestión de la conexión en PHP.

iptcparse (PHP 3>= 3.0.6, PHP 4)

Divide un bloque binario IPTC <http://www.xe.net/iptc/> (<http://www.xe.net/iptc/>) en su tags (etiquetas) individuales.

```
array iptcparse (string iptcblock)
```

Esta función divide un bloque binario IPTC en sus tags individuales. Devuelve un array utilizando el tagmarker (marcador de etiqueta) como un índice y el valor como valor. Devuelve false en caso de error o si no hubiese datos IPTC. Véase **GetImageSize()** para un ejemplo.

leak (PHP 3, PHP 4)

Filtre memoria

```
void leak (int bytes)
```

Leak() filtra la cantidad de memoria especificada.

Es muy útil cuando se depura el gestor de memoria, que automáticamente libera la memoria "filtrada"después de que se completa cada petición.

pack (PHP 3, PHP 4)

empaquea datos en una cadena binaria

```
string pack (string format [, mixed args...])
```

Empaquea los argumentos dados en una cadena binaria siguiendo el formato *format*. Devuelve la cadena binaria que contiene los datos.

El concepto de esta función fue tomado de Perl y todos los códigos de formateo realizan la misma función. La cadena de formato consiste en códigos de formato seguidos por un argumento opcional de repetición. El argumento de repetición puede ser un valor entero o * para repetir hasta el fin de la entrada de datos. Para a, A, h, H la cuenta de repetición representa cuántos caracteres se toman de un argumento de datos, para @ es la posición absoluta donde poner los datos siguientes, para todo lo demás la cuenta de repetición especifica cuántos argumentos de datos se toman y empaquetan en la cadena binaria resultante. Actualmente están implementados:

- a cadena rellena de NUL
- A cadena rellena de ESPACIOS
- h cadena Hex, primero el medio byte inferior
- H cadena Hex, primero el medio byte superior
- c signed (con signo) char
- C unsigned (sin signo) char
- s signed short (siempre 16 bits, distribución de bytes de la máquina)
- S unsigned short (siempre 16 bits, distribución de bytes de la máquina)
- n unsigned short (siempre 16 bits, distribución de bytes gran endian)
- v unsigned short (siempre 16 bits, distribución de bytes pequeño endian)
- i signed integer (distribución de bytes y tamaños dependientes de la máquina)
- I unsigned integer (distribución de bytes y tamaños dependientes de la máquina)
- l signed long (siempre 32 bits, distribución de bytes de la máquina)

- L unsigned long (siempre 32 bits, distribución de bytes de la máquina)
- N unsigned long (siempre 32 bits, distribución de bytes gran endian)
- V unsigned long (siempre 32 bits, distribución de bytes pequeño endian)
- f float (representación y tamaño dependientes de la máquina)
- d double (representación y tamaño dependientes de la máquina)
- x byte NUL
- X Un byte hacia atrás
- @ relleno con NUL en la posición absoluta

Ejemplo 1. cadena de formato para pack

```
$binarydata = pack("nvc*", 0x1234, 0x5678, 65, 66);
```

La cadena binaria resultante tendrá 6 bytes de longitud y contendrá la secuencia de bytes 0x12, 0x34, 0x78, 0x56, 0x41, 0x42.

Adviértase que la distinción entre valores signed (con signo) y unsigned (sin signo) sólo afecta a la función **unpack()**, ya que la función **pack()** da el mismo resultado para códigos de formato con signo y sin signo.

Nótese también que internamente PHP almacena valores enteros como valores con signo de un tamaño dependiente de la máquina. Si le da un valor entero sin signo demasiado grande para ser almacenado, será convertido a un double (doble), lo que a menudo produce resultados no deseados.

serialize (PHP 3>= 3.0.5, PHP 4)

genera una representación almacenable de un valor

```
string serialize (mixed value)
```

serialize() devuelve una cadena que contiene una representación byte-stream de *value* (valor) que puede ser almacenada en algún lugar.

Esto es útil para almacenar o pasar valores PHP sin pérdida de su tipo y estructura.

Para convertir de nuevo la cadena serializada en un valor PHP, utilice **unserialize()**. **serialize()** gestiona los tipos integer, double, string, array (multidimensional) y object (las propiedades del objeto pueden ser serializadas, pero se pierden los métodos).

Ejemplo 1. ejemplo serialize

```
// $session_data contains a multi-dimensional array with session
// information for the current user. We use serialize() to store
// it in a database at the end of the request.

$conn = odbc_connect("webdb", "php", "chicken");
$stmt = odbc_prepare($conn,
                     "UPDATE sessions SET data = ? WHERE id = ?");
$sqldata = array(serialize($session_data), $PHP_AUTH_USER);
if (!odbc_execute($stmt, &$sqldata)) {
    $stmt = odbc_prepare($conn,
                         "INSERT INTO sessions (id, data) VALUES (?, ?)");
    if (!odbc_execute($stmt, &$sqldata)) {
```

```

        /* Something went wrong.  Bitch, whine and moan. */
    }
}

```

sleep (PHP 3, PHP 4)

Ejecución retardada

```
void sleep (int seconds)
```

La función **sleep** retarda la ejecución del programa durante el número de *seconds* (segundos) dado.

Véase también **usleep()**.

uniqid (PHP 3, PHP 4)

Genera un id único.

```
int uniqid (string prefix [, boolean lcg])
```

uniqid() devuelve un identificador único con un prefijo basado en la hora actual en microsegundos. El prefijo puede ser práctico por ejemplo si se generan identificadores simultáneamente en varios host que pueden haber generado el identificador en el mismo microsegundo. *prefix* (prefijo) puede ser de hasta 114 caracteres de longitud.

Si el parámetro opcional *lcg* es true, **uniqid()** añadirá entropía "LCG combinada" al final del valor devuelto, que hará el resultado más único.

Con un *prefix* (prefijo) vacío, la cadena devuelta tendrá una longitud de 13 caracteres. Si *lcg* es true, tendrá 23 caracteres.

Nota: El parámetro *lcg* está disponible sólo en PHP 4 y PHP 3.0.13 y posteriores.

Si necesita un identificador único o testigo, y tiene la intención de hacer público ese testigo al usuario por medio de una red (i.e. cookies de sesión) se recomienda que utilice algo parecido a estas líneas

```
$token = md5(uniqid("")); // no random portion
$better_token = md5(uniqid(rand())); // better, difficult to guess
```

Esto creará un identificador de 32 caracteres (un número hexadecimal de 128 bits) que es extremadamente difícil de predecir.

unpack (PHP 3, PHP 4)

desempaquetar datos de una cadena binaria

```
array unpack (string format, string data)
```

Desempaquetar datos de una cadena binaria en un array, de acuerdo al formato *format*. Devuelve un array que contiene los elementos de la cadena binaria desempaquetados.

Unpack funciona de manera ligeramente diferente a Perl, ya que los datos desempaquetados se almacenan en un array asociativo. Para conseguir ésto debe nombrar los diferentes códigos de formato y separarlos por una barra inclinada /.

Ejemplo 1. cadena de formato unpack

```
$array = unpack("c2chars/nint", $binarydata);
```

El array resultante contendrá las entradas "chars1", "chars2" y "int".

Para una explicación de los códigos de formato véase también: **pack()**

Advierta que PHP almacena internamente los valores enteros con signo. Si desempaquetá un unsigned long (largo sin signo) demasiado grande y es del mismo tamaño tal como PHP almacena internamente los valores, el resultado será un número negativo a pesar de que se especificara desempaquetamiento sin signo.

unserialize (PHP 3>= 3.0.5, PHP 4)

crea un valor PHP de una representación almacenada

```
mixed unserialize (string str)
```

unserialize() toma una variable serializada (véase **serialize()**) y la convierte en un valor PHP. Se devuelve el valor convertido, y puede ser un integer (entero), double (doble), string (cadena), array o object (objeto). Si fue serializado un objeto, sus métodos no son conservados en el valor devuelto.

Ejemplo 1. ejemplo unserialize

```
// Here, we use unserialize() to load session data from a database
// into $session_data. This example complements the one described
// with serialize().

$conn = odbc_connect("webdb", "php", "chicken");
$stmt = odbc_prepare($conn, "SELECT data FROM sessions WHERE id = ?");
$sqldata = array($PHP_AUTH_USER);
if (!odbc_execute($stmt, &$sqldata) || !odbc_fetch_into($stmt, &$tmp)) {
    // if the execute or fetch fails, initialize to empty array
    $session_data = array();
} else {
    // we should now have the serialized data in $tmp[0].
    $session_data = unserialize($tmp[0]);
    if (!is_array($session_data)) {
        // something went wrong, initialize to empty array
        $session_data = array();
    }
}
```

usleep (PHP 3, PHP 4)

Retrasa la ejecución, en microsegundos

```
void usleep (int micro_seconds)
```

La función usleep retrasa la ejecución del programa durante un número de *micro_seconds* (microsegundos) dado. Véase también **sleep()**.

XLIV. mnoGoSearch Functions

These functions allow you to access mnoGoSearch (former UdmSearch) free search engine. In order to have these functions available, you must compile php with mnogosearch support by using the `-with-mnogosearch` option. If you use this option without specifying the path to mnogosearch, php will look for mnogosearch under `/usr/local/mnogosearch` path by default. If you installed mnogosearch at other path you should specify it: `-with-mnogosearch=DIR`.

mnoGoSearch is a full-featured search engine software for intranet and internet servers, distributed under the GNU license. mnoGoSearch has number of unique features, which makes it appropriate for a wide range of application from search within your site to a specialized search system such as cooking recipes or newspaper search, ftp archive search, news articles search, etc. It offers full-text indexing and searching for HTML, PDF, and text documents. mnoGoSearch consists of two parts. The first is an indexing mechanism (indexer). The purpose of indexer is to walk through HTTP, FTP, NEWS servers or local files, recursively grabbing all the documents and storing meta-data about that documents in a SQL database in a smart and effective manner. After every document is referenced by its corresponding URL, meta-data collected by indexer is used later in a search process. The search is performed via Web interface. C CGI, PHP and Perl search front ends are included.

Nota: php contains built-in mysql access library, which can be used to access mysql. It is known that mnoGoSearch is not compatible with this built-in library and can work only with generic mysql libraries. Thus, if you use mnoGoSearch with mysql, during php configuration you have to indicate directory of mysql installation, that was used during mnoGoSearch configuration, i.e. for example: `-with-mnogosearch -with-mysql=/usr`

You need at least 3.1.10 version of mnoGoSearch installed to use these functions.

More information about mnoGoSearch can be found at <http://www.mnogosearch.ru/>.

udm_alloc_agent (PHP 4 CVS only)

Allocate mnoGoSearch session

```
int udm_alloc_agent (string dbaddr [, string dbmode])
```

udm_alloc_agent() returns mnogosearch agent identifier on success, FALSE on error. This function creates a session with database parameters.

dbaddr - URL-style database description. Options (type, host, database name, port, user and password) to connect to SQL database. Do not matter for built-in text files support. Format: DBAddr

DBType:[/][DBUser[:DBPass]@]JDBHost[:DBPort]/[DBName/] Currently supported DBType values are: mysql, pgsql, msql, solid, mssql, oracle, ibase. Actually, it does not matter for native libraries support. But ODBC users should specify one of supported values. If your database type is not supported, you may use "unknown" instead.

dbmode - You may select SQL database mode of words storage. When "single" is specified, all words are stored in the same table. If "multi" is selected, words will be located in different tables depending of their lengths. "multi" mode is usually faster but requires more tables in database. If "crc" mode is selected, mnoGoSearch will store 32 bit integer word IDs calculated by CRC32 algorythm instead of words. This mode requires less disk space and it is faster comparing with "single" and "multi" modes. "crc-multi" uses the same storage structure with the "crc" mode, but also stores words in different tables depending on words lengths like "multi" mode. Format: DBMode single/multi/crc/crc-multi

Nota: *dbaddr* and *dbmode* must match those used during indexing.

Nota: In fact this function does not open connection to database and thus does not check entered login and password. Actual connection to database and login/password verification is done by **udm_find()**.

udm_set_agent_param (PHP 4 CVS only)

Set mnoGoSearch agent session parameters

```
int udm_set_agent_param (int agent, int var, string val)
```

udm_set_agent_param() returns TRUE on success, FALSE on error. Defines mnoGoSearch session parameters.

The following parameters and their values are available:

- UDM_PARAM_PAGE_NUM - used to choose search results page number (results are returned by pages beginning from 0, with UDM_PARAM_PAGE_SIZE results per page).
- UDM_PARAM_PAGE_SIZE - number of search results displayed on one page.
- UDM_PARAM_SEARCH_MODE - search mode. The following values available: UDM_MODE_ALL - search for all words; UDM_MODE_ANY - search for any word; UDM_MODE_PHRASE - phrase search; UDM_MODE_BOOL - boolean search. See **udm_find()** for details on boolean search.
- UDM_PARAM_CACHE_MODE - turns on or off search result cache mode. When enabled, the search engine will store search results to disk. In case a similar search is performed later, the engine will take results from the cache for faster performance. Available values: UDM_CACHE_ENABLED, UDM_CACHE_DISABLED.
- UDM_PARAM_TRACK_MODE - turns on or off trackquery mode. Since version 3.1.2 mnoGoSearch has a query tracking support. Note that tracking is implemented in SQL version only and not available in built-in database. To use tracking, you have to create tables for tracking support. For MySQL, use create/mysql/track.txt. When doing a search,

front-end uses those tables to store query words, a number of found documents and current UNIX timestamp in seconds. Available values: UDM_TRACK_ENABLED, UDM_TRACK_DISABLED.

- UDM_PARAM_PHRASE_MODE - defines whether index database using phrases ("phrase" parameter in indexer.conf). Possible values: UDM_PHRASE_ENABLED and UDM_PHRASE_DISABLED. Please note, that if phrase search is enabled (UDM_PHRASE_ENABLED), it is still possible to do search in any mode (ANY, ALL, BOOL or PHRASE). In 3.1.10 version of mnoGoSearch phrase search is supported only in sql and built-in database modes, while beginning with 3.1.11 phrases are supported in cachemode as well.

Examples of phrase search:

"Arizona desert"- This query returns all indexed documents that contain "Arizona desert"as a phrase. Notice that you need to put double quotes around the phrase

- UDM_PARAM_CHARSET - defines local charset. Available values: set of charsets supported by mnoGoSearch, e.g. koi8-r, cp1251, ...
- UDM_PARAM_STOPFILE - Defines name and path to stopwords file. (There is a small difference with mnoGoSearch - while in mnoGoSearch if relative path or no path entered, it looks for this file in relation to UDM_CONF_DIR, the module looks for the file in relation to current path, i.e. to the path where the php script is executed.)
- UDM_PARAM_STOPTABLE - Load stop words from the given SQL table. You may use several StopwordTable commands. This command has no effect when compiled without SQL database support.
- UDM_PARAM_WEIGHT_FACTOR - represents weight factors for specific document parts. Currently body, title, keywords, description, url are supported. To activate this feature please use degrees of 2 in *Weight commands of the indexer.conf. Let's imagine that we have these weights:

URLWeight 1

BodyWeight 2

TitleWeight 4

KeywordWeight 8

DescWeight 16

As far as indexer uses bit OR operation for word weights when some word presents several time in the same document, it is possible at search time to detect word appearance in different document parts. Word which appears only in the body will have 00000010 argegate weight (in binary notation). Word used in all document parts will have 00011111 aggregate weight.

This parameter's value is a string of hex digits ABCDE. Each digit is a factor for corresponding bit in word weight. For the given above weights configuration:

E is a factor for weight 1 (URL Weight bit)

D is a factor for weight 2 (BodyWeight bit)

C is a factor for weight 4 (TitleWeight bit)

B is a factor for weight 8 (KeywordWeight bit)

A is a factor for weight 16 (DescWeight bit)

Examples:

UDM_PARAM_WEIGHT_FACTOR=00001 will search through URLs only.

UDM_PARAM_WEIGHT_FACTOR=00100 will search through Titles only.

UDM_PARAM_WEIGHT_FACTOR=11100 will search through Title,Keywords,Description but not through URL and Body.

UDM_PARAM_WEIGHT_FACTOR=F9421 will search through:

Description with factor 15 (F hex)

Keywords with factor 9

Title with factor 4

Body with factor 2

URL with factor 1

If UDM_PARAM_WEIGHT_FACTOR variable is omitted, original weight value is taken to sort results. For a given above weight configuration it means that document description has a most big weight 16.

- UDM_PARAM_WORD_MATCH - word match. You may use this parameter to choose word match type. This feature works only in "single" and "multi" modes using SQL based and built-in database. It does not work in cachemode and other modes since they use word CRC and do not support substring search. Available values:

UDM_MATCH_BEGIN - word beginning match;

UDM_MATCH_END - word ending match;

UDM_MATCH_WORD - whole word match;

UDM_MATCH_SUBSTR - word substring match.

- UDM_PARAM_MIN_WORD_LEN - defines minimal word length. Any word shorter this limit is considered to be a stopword. Please note that this parameter value is inclusive, i.e. if UDM_PARAM_MIN_WORD_LEN=3, a word 3 characters long will not be considered a stopword, while a word 2 characters long will be. Default value is 1.
- UDM_PARAM_ISPELL_PREFIXES - Possible values: UDM_PREFIXES_ENABLED and UDM_PREFIXES_DISABLED, that respectively enable or disable using prefixes. E.g. if a word "tested" is in search query, also words like "test", "testing", etc. Only suffixes are supported by default. Prefixes usually change word meanings, for example if somebody is searching for the word "tested" one hardly wants "untested" to be found. Prefixes support may also be found useful for site's spelling checking purposes. In order to enable ispell, you have to load ispell data with **udm_load_ispell_data()**.

udm_add_search_limit (PHP 4 CVS only)

Add various search limits

```
int udm_add_search_limit (int agent, int var, string val)
```

udm_add_search_limit() returns TRUE on success, FALSE on error. Adds search restrictions.

agent - a link to Agent, received after call to **udm_alloc_agent()**.

var - defines parameter, indicating limit.

val - defines value of the current parameter.

Possible *var* values:

- UDM_LIMIT_URL - defines document URL limitations to limit search through subsection of database. It supports SQL % and _ LIKE wildcards, where % matches any number of characters, even zero characters, and _ matches exactly one character. E.g. http://my.domain._/catalog may stand for http://my.domain.ru/catalog and http://my.domain.ua/catalog.
- UDM_LIMIT_TAG - defines site TAG limitations. In indexer-conf you can assign specific TAGs to various sites and parts of a site. Tags in mnoGoSearch 3.1.x are lines, that may contain metasymbols % and _. Metasymbols allow searching among groups of tags. E.g. there are links with tags ABCD and ABCE, and search restriction is by ABC_ - the search will be made among both of the tags.
- UDM_LIMIT_LANG - defines document language limitations.

- UDM_LIMIT_CAT - defines document category limitations. Categories are similar to tag feature, but nested. So you can have one category inside another and so on. You have to use two characters for each level. Use a hex number going from 0-F or a 36 base number going from 0-Z. Therefore a top-level category like 'Auto' would be 01. If it has a subcategory like 'Ford', then it would be 01 (the parent category) and then 'Ford' which we will give 01. Put those together and you get 0101. If 'Auto' had another subcategory named 'VW', then its id would be 01 because it belongs to the 'Ford' category and then 02 because it's the next category. So its id would be 0102. If VW had a sub category called 'Engine' then its id would start at 01 again and it would get the 'VW' id 02 and 'Auto' id of 01, making it 010201. If you want to search for sites under that category then you pass it cat=010201 in the url.
- UDM_LIMIT_DATE - defines limitation by date document was modified.

Format of parameter value: a string with first character < or >, then with no space - date in unixtime format, for example:

```
Udm_Add_Search_Limit($udm,UDM_LIMIT_DATE,"<908012006");
```

If > character is used, then search will be restricted to those documents having modification date greater than entered. If <, then smaller.

udm_clear_search_limits (PHP 4 CVS only)

Clear all mnoGoSearch search restrictions

```
int udm_clear_search_limits (int agent)
```

udm_clear_search_limits() resets defined search limitations and returns TRUE.

udm_find (PHP 4 CVS only)

Perform search

```
int udm_find (int agent, string query)
```

udm_find() returns result link identifier on success, FALSE on error.

The search itself. The first argument - session, the next one - query itself. To find something just type words you want to find and press SUBMIT button. For example, "mysql odbc". You should not use quotes "in query, they are written here only to divide a query from other text. mnoGoSearch will find all documents that contain word "mysql"and/or word "odbc". Best documents having bigger weights will be displayed first. If you use search mode ALL, search will return documents that contain both (or more) words you entered. In case you use mode ANY, the search will return list of documents that contain any of the words you entered. If you want more advanced results you may use query language. You should select "bool" match mode in the search from.

mnoGoSearch understands the following boolean operators:

& - logical AND. For example, "mysql & odbc". mnoGoSearch will find any URLs that contain both "mysql"and "odbc".

| - logical OR. For example "mysql|odbc". mnoGoSearch will find any URLs, that contain word "mysql"or word "odbc".

~ - logical NOT. For example "mysql & ~odbc". mnoGoSearch will find URLs that contain word "mysql"and do not contain word "odbc"at the same time. Note that ~ just excludes given word from results. Query "~odbc"will find nothing!

() - group command to compose more complex queries. For example "(mysql | msq) & ~postgres". Query language is simple and powerful at the same time. Just consider query as usual boolean expression.

udm_get_res_param (PHP 4 CVS only)

Get mnoGoSearch result parameters

```
string udm_get_res_param (int res, int param)
```

udm_get_res_param() returns result parameter value on success, FALSE on error.

res - a link to result identifier, received after call to **udm_find()**.

param - parameter identifier, may have the following values:

- UDM_PARAM_NUM_ROWS - number of received found links on the current page. It is equal to UDM_PARAM_PAGE_SIZE for all search pages, on the last page - the rest of links.
- UDM_PARAM_FOUND - total number of results matching the query.
- UDM_PARAM_WORDINFO - information on the words found. E.g. search for "a good book" will return "a: stopword, good:5637, book: 120"
- UDM_PARAM_SEARCHTIME - search time in seconds.
- UDM_PARAM_FIRST_DOC - the number of the first document displayed on current page.
- UDM_PARAM_LAST_DOC - the number of the last document displayed on current page.

udm_get_res_field (PHP 4 CVS only)

Fetch mnoGoSearch result field

```
string udm_get_res_field (int res, int row, int field)
```

udm_get_res_field() returns result field value on success, FALSE on error.

res - a link to result identifier, received after call to **udm_find()**.

row - the number of the link on the current page. May have values from 0 to *UDM_PARAM_NUM_ROWS*.

field - field identifier, may have the following values:

- UDM_FIELD_URL - document URL field
- UDM_FIELD_CONTENT - document Content-type field (for example, text/html).
- UDM_FIELD_TITLE - document title field.
- UDM_FIELD_KEYWORDS - document keywords field (from META KEYWORDS tag).
- UDM_FIELD_DESC - document description field (from META DESCRIPTION tag).
- UDM_FIELD_TEXT - document body text (the first couple of lines to give an idea of what the document is about).
- UDM_FIELD_SIZE - document size.
- UDM_FIELD_URLID - unique URL ID of the link.
- UDM_FIELD_RATING - page rating (as calculated by mnoGoSearch).
- UDM_FIELD_MODIFIED - last-modified field in unixtime format.
- UDM_FIELD_ORDER - the number of the current document in set of found documents.
- UDM_FIELD_CRC - document CRC.

udm_load_ispell_data (unknown)

Load ispell data

```
int udm_load_ispell_data (int agent, int var, string val1, string val2, int flag)
```

udm_load_ispell_data() loads ispell data. Returns TRUE on success, FALSE on error.

agent - agent link identifier, received after call to **udm_alloc_agent()**.

var - parameter, indicating the source for ispell data. May have the following values:

Nota: It is recommended to load ispell data from files, since in mnogosearch 3.1.10 it is the fastest. In later versions it is planned to optimize loading in UDM_ISPELL_TYPE_DB mode as well, so you just try several modes to find the best for you.

- UDM_ISPELL_TYPE_DB - indicates that ispell data should be loaded from SQL. In this case, parameters *val1* and *val2* are ignored and should be left blank. *flag* should be equal to 1.

Nota: *flag* indicates that after loading ispell data from defined source it should be sorted (it is necessary for correct functioning of ispell). In case of loading ispell data from files there may be several calls to **udm_load_ispell_data()**, and there is no sense to sort data after every call, but only after the last one. Since in db mode all the data is loaded by one call, this parameter should have the value 1. In this mode in case of error, e.g. if ispell tables are absent, the function will return FALSE and code and error message will be accessible through **udm_error()** and **udm_errno()**.

Example:

```
if (! Udm_Load_Ispell_Data($udm, UDM_ISPELL_TYPE_DB, "", "", 1)) {
    printf("Error #d: '%s'\n", Udm_Error($udm), Udm_errno($udm));
    exit;
}
```

- UDM_ISPELL_TYPE_SUFFIX - indicates that ispell data should be loaded from file and initiates loading affixes file. In this case *val1* defines double letter language code for which affixes are loaded, and *val2* - file path. Please note, that if a relative path entered, the module looks for the file not in UDM_CONF_DIR, but in relation to current path, i.e. to the path where the script is executed. In case of error in this mode, e.g. if file is absent, the function will return FALSE, and an error message will be displayed. Error message text cannot be accessed through **udm_error()** and **udm_errno()**, since those functions can only return messages associated with SQL. Please, see *flag* parameter description in UDM_ISPELL_TYPE_DB.

Example:

```
if (( ! Udm_Load_Ispell_Data($udm, UDM_ISPELL_TYPE_SUFFIX, 'en', '/opt/ispell/en.aff', 0))
    ( ! Udm_Load_Ispell_Data($udm, UDM_ISPELL_TYPE_SUFFIX, 'ru', '/opt/ispell/ru.aff', 0))
    ( ! Udm_Load_Ispell_Data($udm, UDM_ISPELL_TYPE_SPELL, 'en', '/opt/ispell/en.dict', 0))
    ( ! Udm_Load_Ispell_Data($udm, UDM_ISPELL_TYPE_SPELL, 'ru', '/opt/ispell/ru.dict', 1)))
    exit;
}
```

Nota: *flag* is equal to 1 only in the last call.

- UDM_ISPELL_TYPE_SPELL - indicates that ispell data should be loaded from file and initiates loading of ispell dictionary file. In this case *val1* defines double letter language code for which affixes are loaded, and *val2* - file path.

Please note, that if a relative path entered, the module looks for the file not in UDM_CONF_DIR, but in relation to current path, i.e. to the path where the script is executed. In case of error in this mode, e.g. if file is absent, the function will return FALSE, and an error message will be displayed. Error message text cannot be accessed through **udm_error()** and **udm_errno()**, since those functions can only return messages associated with SQL. Please, see *flag* parameter description in UDM_ISPELL_TYPE_DB.

Example:

```
if ((! Udm_Load_Ispell_Data($udm, UDM_ISPELL_TYPE_AFFIX, 'en', '/opt/ispell/en.aff', 0))
    (! Udm_Load_Ispell_Data($udm, UDM_ISPELL_TYPE_AFFIX, 'ru', '/opt/ispell/ru.aff', 0))
    (! Udm_Load_Ispell_Data($udm, UDM_ISPELL_TYPE_SPELL, 'en', '/opt/ispell/en.dict', 0))
    (! Udm_Load_Ispell_Data($udm, UDM_ISPELL_TYPE_SPELL, 'ru', '/opt/ispell/ru.dict', 1)))
    exit;
}
```

Nota: *flag* is equal to 1 only in the last call.

udm_free_spell_data (unknown)

Free memory allocated for ispell data

```
int udm_free_spell_data (int agent)
```

udm_free_spell_data() always returns TRUE.

agent - agent link identifier, received after call to **udm_alloc_agent()**.

Nota: In mnoGoSearch 3.1.10 this function is not yet implemented, it is added for compatibility with future versions and does not perform anything yet.

udm_free_res (PHP 4 CVS only)

Free mnoGoSearch result

```
int udm_free_res (int res)
```

udm_free_res() returns TRUE on success, FALSE on error.

res - a link to result identifier, received after call to **udm_find()**.

Freeing up memory allocated for results.

udm_free_agent (PHP 4 CVS only)

Free mnoGoSearch session

```
int udm_free_agent (int agent)
```

udm_free_agent() returns TRUE on success, FALSE on error.

agent - link to agent identifier, received after call to **udm_alloc_agent()**.

Freeing up memory allocated for agent session.

udm_errno (PHP 4 CVS only)

Get mnoGoSearch error number

```
int udm_errno (int agent)
```

udm_errno() returns mnoGoSearch error number, zero if no error.

agent - link to agent identifier, received after call to **udm_alloc_agent()**.

Receiving numeric agent error code.

udm_error (PHP 4 CVS only)

Get mnoGoSearch error message

```
string udm_error (int agent)
```

udm_error() returns mnoGoSearch error message, empty string if no error.

agent - link to agent identifier, received after call to **udm_alloc_agent()**.

Receiving agent error message.

XLV. funciones mSQL

msql (PHP 3, PHP 4)

ejecuta una consulta mSQL

```
int msql (string database, string query, int link_identifier)
```

Devuelve un identificador de consulta mSQL positivo en el resultado de la consulta, o false en caso de error.

msql() selecciona una base de datos y ejecuta una consulta en ella. Si no se especifica el identificador de conexión (link identifier), la función intentará encontrar una conexión abierta en el servidor mSQL y en el caso de que no se encontrase intentará crear uno como si se llamase a **msql_connect()** sin parámetros (véase **msql_connect()**).

msql_affected_rows (PHP 3>= 3.0.6, PHP 4)

devuelve el número de filas involucradas

```
int msql_affected_rows (int query_identifier)
```

Devuelve el número de filas involucradas ("tocadas") por una consulta específica (i.e. el número de filas devueltas por una SELECT, el número de filas modificadas por una actualización (update), o el número de filas suprimidas por una eliminación (delete)).

Véase también: **msql_query()**

msql_close (PHP 3, PHP 4)

cierra una conexión mSQL

```
int msql_close (int link_identifier)
```

Devuelve true si tiene éxito, false en caso de error.

msql_close() cierra la conexión con una base de datos mSQL que está asociada con el identificador de conexión (link identifier) especificado. Si no se especifica el identificador de conexión, se asume la última conexión abierta.

Advierta que ésto no es necesario habitualmente, las conexiones abiertas no-persistentes son cerradas automáticamente a la conclusión de la ejecución del script.

msql_close() no cerrará las conexiones permanentes creadas por **msql_pconnect()**.

Véase también: **msql_connect()** y **msql_pconnect()**.

msql_connect (PHP 3, PHP 4)

abre una conexión mSQL

```
int msql_connect (string hostname)
```

Devuelve un identificador de conexión mSQL positivo si tiene éxito, o false en caso de error.

msql_connect() establece una conexión con un servidor mSQL. El argumento hostname es opcional, y en caso de que falte, se asume localhost.

En caso de que se haga una segunda llamada a `msql_connect()` con los mismos argumentos, no se establece una nueva conexión, en lugar de eso, se devuelve el identificador de conexión ya abierto.

La conexión con el servidor se cerrará tan pronto como la ejecución del script finalice, a menos que sea cerrada antes explícitamente por una llamada a `msql_close()`.

Véase también: `msql_pconnect()`, `msql_close()`.

msql_create_db (PHP 3, PHP 4)

crea una base de datos mSQL

```
int msql_create_db (string database name [, int link_identifier])
```

`msql_create_db()` intenta crear una base de datos nueva en el servidor asociado con el identificador de conexión (link identifier) especificado.

Véase también: `msql_drop_db()`.

msql_createdb (PHP 3, PHP 4)

crea una base de datos mSQL

```
int msql_createdb (string database name [, int link_identifier])
```

Idéntica a `msql_create_db()`.

msql_data_seek (PHP 3, PHP 4)

desplaza el puntero interno de fila

```
int msql_data_seek (int query_identifier, int row_number)
```

Devuelve true si tiene éxito, false en caso de fallo.

`msql_data_seek()` desplaza el puntero interno de fila del resultado mSQL asociado con el identificador de consulta (query identifier) especificado para que apunte al número de fila (row number) proporcionado. La llamada posterior a `msql_fetch_row()` devolverá esa fila.

Véase también: `msql_fetch_row()`.

msql_dbname (PHP 3, PHP 4)

obtiene el nombre de la base de datos mSQL actual

```
string msql_dbname (int query_identifier, int i)
```

`msql_dbname()` devuelve el nombre de base de datos almacenado en la posición *i* del puntero devuelto desde la función `msql_listdbs()`. La función `msql_numrows()` puede utilizarse para determinar cuantos nombres de base de datos hay disponibles.

mSQL_drop_db (PHP 3, PHP 4)

elimina (suprime) una base de datos mSQL

```
int mSQL_drop_db (string database_name, int link_identifier)
```

Devuelve true si tiene éxito, false en caso de fallo.

mSQL_drop_db() intenta eliminar (suprimir) una base de datos completa del servidor asociado con el identificador de conexión (link identifier) especificado.

Véase también: **mSQL_create_db()**.

mSQL_dropdb (PHP 3, PHP 4)

elimina (suprime) una base de datos mSQL

Véase **mSQL_drop_db()**.

mSQL_error (PHP 3, PHP 4)

devuelve el mensaje de error de la última llamada mSQL

```
string mSQL_error ()
```

Los errores que devuelve el servidor de base de datos mSQL no dan mucha información sobre el problema. Por este motivo, utilice estas funciones para recuperar la cadena de caracteres del error.

mSQL_fetch_array (PHP 3, PHP 4)

recupera una fila como un array

```
int mSQL_fetch_array (int query_identifier [, int result_type])
```

Devuelve un array que se corresponde con la fila recuperada, o false si no hay más filas.

mSQL_fetch_array() es una versión ampliada de **mSQL_fetch_row()**. Además de almacenar los datos en los índices numéricos del array resultado, también almacena los datos en índices asociativos, utilizando los nombres de los campos como claves.

El segundo argumento opcional *result_type* en **mSQL_fetch_array()** es una constante y puede tomar los valores siguientes: MSQL_ASSOC, MSQL_NUM, y MYSQL_BOTH.

Sea precavido si está recuperando resultados de una consulta que puede devolver un registro que contiene un único campo que tiene un valor de 0 (o una cadena de caracteres vacía, o NULL).

Un aspecto importante a tener en cuenta es que el uso de **mSQL_fetch_array()** NO es significativamente más lento que el uso de **mSQL_fetch_row()**, mientras que proporciona un valor añadido significativo.

Para detalles adicionales, véase también **mSQL_fetch_row()**

msql_fetch_field (PHP 3, PHP 4)

obtiene información de campo

```
object msql_fetch_field (int query_identifier, int field_offset)
```

Devuelve un objeto que contiene la información del campo

msql_fetch_field() puede utilizarse para obtener información sobre campos del resultado de una consulta. Si no se especifica el desplazamiento del campo, se recupera el campo siguiente que no haya sido aún recuperado por **msql_fetch_field()**.

Las propiedades del objeto son:

- name - nombre de la columna
- table - nombre de la tabla a la que pertenece la columna
- not_null - 1 si la columna no puede ser nula
- primary_key - 1 si la columna es una clave primaria
- unique - 1 si la columna es una clave única
- type - tipo de la columna

Véase también **msql_field_seek()**.

msql_fetch_object (PHP 3, PHP 4)

recupera una fila como un objeto

```
int msql_fetch_object (int query_identifier [, int result_type])
```

Devuelve un objeto con las propiedades que corresponden a la fila recuperada, o false si no hay más filas.

msql_fetch_object() es similar a **msql_fetch_array()**, con una diferencia - se devuelve un objeto en vez de un array. Indirectamente esto significa que sólo tiene acceso a los datos por los nombres de los campos, y no por sus desplazamientos (los números son nombres de propiedad no válidos).

El segundo parámetro opcional *result_type* en **msql_fetch_array()** es una constante y puede tomar los valores siguientes: MSQL_ASSOC, MSQL_NUM, y MSQL_BOTH.

Resumiendo, la función es idéntica a **msql_fetch_array()**, y casi tan rápida como **msql_fetch_row()** (la diferencia es insignificante).

Véase también: **msql_fetch_array()** y **msql_fetch_row()**.

msql_fetch_row (PHP 3, PHP 4)

obtiene una fila como un array enumerado

```
array msql_fetch_row (int query_identifier)
```

Devuelve un array que se corresponde con la fila recuperada, o false si no hay más filas.

`mysql_fetch_row()` recupera una fila de datos del resultado asociado con el identificador de consulta (query identifier) especificado. La fila se devuelve en un array. Cada columna devuelta se almacena en un desplazamiento del array, comenzando en el desplazamiento 0.

Una llamada posterior a `mysql_fetch_row()` debería devolver la fila siguiente del conjunto resultado, o false si no hay más filas.

Véase también: **`mysql_fetch_array()`**, **`mysql_fetch_object()`**, **`mysql_data_seek()`**, y **`mysql_result()`**.

`mysql_fieldname` (PHP 3, PHP 4)

obtiene el nombre del campo

```
string mysql_fieldname (int query_identifier, int field)
```

`mysql_fieldname()` devuelve el nombre del campo especificado. *query_identifier* es el identificador de consulta, y *field* es el índice del campo. `mysql_fieldname($result, 2);` devolverá el nombre del segundo campo del resultado asociado con el identificador result.

`mysql_field_seek` (PHP 3, PHP 4)

establece el desplazamiento del campo

```
int mysql_field_seek (int query_identifier, int field_offset)
```

Se posiciona en el desplazamiento de campo (field offset) especificado. Si la siguiente llamada a **`mysql_fetch_field()`** no incluye un desplazamiento de campo, este campo será el que se devuelva.

Véase también: **`mysql_fetch_field()`**.

`mysql_fieldtable` (PHP 3, PHP 4)

obtiene el nombre de la tabla de un campo

```
int mysql_fieldtable (int query_identifier, int field)
```

Devuelve el nombre de la tabla desde la que se ha recuperado el campo (*field*)

`mysql_fieldtype` (PHP 3, PHP 4)

obtiene el tipo del campo

```
string mysql_fieldtype (int query_identifier, int i)
```

`mysql_fieldtype()` es similar a la función **`mysql_fieldname()`**. Los argumentos son idénticos, pero se devuelve el tipo del campo. Este será "int", "char" o "real".

mSQL_fieldflags (PHP 3, PHP 4)

obtiene los flags del campo

```
string msql_fieldflags (int query_identifier, int i)
```

msql_fieldflags() obtiene los flags del campo (field) especificado. Actualmente pueden ser, "not null", "primary key", una combinación de ambos, o ""(cadena vacía).

mSQL_fieldlen (PHP 3, PHP 4)

obtiene la longitud del campo

```
int msql_fieldlen (int query_identifier, int i)
```

msql_fieldlen() devuelve la longitud del campo especificado.

mSQL_free_result (PHP 3, PHP 4)

libera la memoria del resultado

```
int msql_free_result (int query_identifier)
```

msql_free_result() libera la memoria asociada con *query_identifier*. Cuando PHP completa una petición, esta memoria es liberada automáticamente, por este motivo solo es necesario llamar a esta función cuando se desea estar seguro de que no se utiliza demasiada memoria mientras se está ejecutando el script.

mSQL_freeResult (PHP 3, PHP 4)

libera la memoria del resultado

Véase **mysql_free_result()**

mSQL_list_fields (PHP 3, PHP 4)

lista los campos del resultado

```
int msql_list_fields (string database, string tablename)
```

msql_list_fields() recupera información sobre el nombre de tabla (tablename) dado. Los argumentos son el nombre de la base de datos (database name) y el nombre de la tabla (table name). Se devuelve un puntero al resultado que puede utilizarse con **mysql_fieldflags()**, **mysql_fieldlen()**, **mysql_fieldname()**, y **mysql_fieldtype()**. Un identificador de consulta (query identifier) es un entero positivo. La función devuelve -1 si ocurre un error. En \$phperrormsg se almacena una cadena de caracteres describiendo el error, y a menos que la función sea llamada como @mysql_list_fields() esta cadena de error puede ser impresa.

Véase también **mysql_error()**.

mSQL_listfields (PHP 3, PHP 4)

lista los campos del resultado

Véase **mSQL_list_fields()**.

mSQL_list_dbs (PHP 3, PHP 4)

lista las bases de datos mSQL en el servidor

```
int mSQL_list_dbs(void);
```

mSQL_list_dbs() devolverá un puntero al resultado que contiene las bases de datos disponibles desde el daemon mSQL en uso. Utilice la función **mSQL_dbname()** para recorrer este puntero.

mSQL_listdbs (PHP 3, PHP 4)

lista las bases de datos mSQL en el servidor

Véase **mSQL_list_dbs()**.

mSQL_list_tables (PHP 3, PHP 4)

lista las tablas de una base de datos mSQL

```
int mSQL_list_tables (string database)
```

mSQL_list_tables() toma un nombre de base de datos y devuelve un puntero similar al de la función **mSQL()**. La función **mSQL_tablename()** debería utilizarse para obtener los nombres reales de las tablas del puntero devuelto.

mSQL_listtables (PHP 3, PHP 4)

lista las tablas de una base de datos mSQL

Véase **mSQL_list_tables()**.

mSQL_num_fields (PHP 3, PHP 4)

obtiene el número de campos de un resultado

```
int mSQL_num_fields (int query_identifier)
```

mSQL_num_fields() devuelve el número de campos de un conjunto resultado.

Véase también: **mSQL()**, **mSQL_query()**, **mSQL_fetch_field()**, y **mSQL_num_rows()**.

msql_num_rows (PHP 3, PHP 4)

obtiene el número de filas de un resultado

```
int msql_num_rows (int query_identifier)
```

mysql_num_rows() devuelve el número de filas de un conjunto resultado.

Véase también: **msql()**, **mysql_query()**, y **mysql_fetch_row()**.

msql_numfields (PHP 3, PHP 4)

obtiene el número de campos de un resultado

```
int msql_numfields (int query_identifier)
```

Idéntica a **mysql_num_fields()**.

msql_numrows (PHP 3, PHP 4)

obtiene el número de filas en el resultado

```
int msql_numrows(void);
```

Idéntica a **mysql_num_rows()**.

msql_pconnect (PHP 3, PHP 4)

abre una conexión mSQL persistente

```
int msql_pconnect (string hostname)
```

En caso de éxito devuelve un identificador de conexión mSQL persistente positivo, o false en caso de error.

mysql_pconnect() se comporta de forma similar a **mysql_connect()** con dos diferencias importantes.

Primero, cuando se conecta, la función debe intentar primero localizar una conexión (persistente) que ya esté abierta en el mismo host. Si se encuentra uno, se devuelve un identificador para el mismo en vez de abrir una conexión nueva.

Segundo, la conexión con el servidor SQL no se cerrará cuando la ejecución del script finalice. Al contrario, la conexión permanecerá abierta para un uso futuro (**mysql_close()** no cerrará las conexiones abiertas por mysql_pconnect()).

Este tipo de conexiones son por ello denominadas 'persistentes'.

msql_query (PHP 3, PHP 4)

envía una consulta mSQL

```
int msql_query (string query, int link_identifier)
```

`msql_query()` envía una consulta a la base de datos activa actual en el servidor que está asociada con el identificador de conexión (link identifier) especificado. Si no se especifica el identificador de conexión, se asume la última conexión abierta. Si no hay ninguna conexión abierta, la función intenta establecer una conexión como si se hubiera llamado a `msql_connect()`, y la utiliza.

En caso de éxito devuelve un identificador de consulta mSQL positivo, o false en caso de error.

Véase también: `msql()`, `msql_select_db()`, y `msql_connect()`.

msql_regcase (PHP 3, PHP 4)

construye una expresión regular para una búsqueda que no distinga mayúsculas/minúsculas

Véase `sql_regcase()`.

msql_result (PHP 3, PHP 4)

obtiene datos resultado

```
int msql_result (int query_identifier, int i, mixed field)
```

Devuelve el contenido de la celda en la fila y desplazamiento del conjunto resultado mSQL especificado.

`msql_result()` devuelve el contenido de una celda de un conjunto resultado mSQL. El argumento campo (field) puede ser el desplazamiento del campo, el nombre del campo, o el nombre de la tabla punto nombre del campo (nombretabla.nombrecampo). Si el nombre de la columna tiene un alias ('select foo as bar from...'), utilice el alias en vez del nombre de la columna.

Cuando se trabaja con conjuntos de resultados grandes, debería considerar el uso de las funciones que recuperen filas completas (especificadas más abajo). Como estas funciones recuperan el contenido de varias celdas en una única llamada de función, son MUCHO más rápidas que `msql_result()`. Advierta también que especificar un desplazamiento numérico para el argumento campo (field) es mucho más rápido que especificar un argumento nombrecampo o nombretabla.nombrecampo.

Alternativas de alto-rendimiento recomendadas: `msql_fetch_row()`, `msql_fetch_array()`, y `msql_fetch_object()`.

msql_select_db (PHP 3, PHP 4)

selecciona una base de datos mSQL

```
int msql_select_db (string database_name, int link_identifier)
```

Devuelve true si tiene éxito, false en caso contrario.

`msql_select_db()` establece la base de datos activa actual en el servidor que está asociada con el identificador de conexión (link identifier) suministrado. Si no se especifica el identificador de conexión, se asumne la última conexión abierta. Si no hay ninguna conexión abierta la función intentará establecer una conexión como si se hubiera llamado a `sql_connect()`, y la utiliza.

Cada llamada posterior a `msql_query()` se hará en la base de datos activa.

Véase también: `msql_connect()`, `msql_pconnect()`, y `msql_query()`.

mSQL_selectdb (PHP 3, PHP 4)

selecciona una base de datos mSQL

Véase **mSQL_select_db()**.

mSQL_tablename (PHP 3, PHP 4)

obtiene el nombre de la tabla de un campo

```
string mSQL_tablename (int query_identifier, int field)
```

mSQL_tablename() toma un puntero resultado devuelto por la función **mSQL_list_tables()** como un índice entero y devuelve el nombre de una tabla. La función **mSQL_numrows()** puede utilizarse para determinar el número de tablas del puntero resultado.

Ejemplo 1. mSQL_tablename() example

```
<?php
mSQL_connect ("localhost");
$result = mSQL_list_tables("wisconsin");
$i = 0;
while ($i < mSQL_numrows($result)) {
    $tb_names[$i] = mSQL_tablename($result, $i);
    echo $tb_names[$i] . "<BR>";
    $i++;
}
?>
```

XLVI. Funciones MySQL

Estas funciones le permiten acceder a servidores de bases de datos MySQL.

Puede encontrar más información sobre MySQL en <http://www.mysql.com/>.

mysql_affected_rows (PHP 3, PHP 4)

Devuelve el número de filas afectadas de la última operación MySQL

```
int mysql_affected_rows ([int identificador_de_enlace])
```

mysql_affected_rows() devuelve el número de filas afectadas en la ultima sentencia INSERT, UPDATE o DELETE sobre el servidor asociado con el identificador de enlace especificado. Si el identificador de enlace no ha sido especificado, se asume por defecto el último enlace.

Si la última sentencia fue un DELETE sin clausula WHERE, todos los registros han sido borrados de la tabla pero esta función devolverá cero.

Este comando no es efectivo para las sentencias SELECT, sino sólo para las sentencias que modifican registros. Para conseguir el número de líneas devueltas por un SELECT, usar **mysql_num_rows()**.

mysql_change_user (PHP 3>= 3.0.13)

Cambia el usuario conectado en la conexión activa

```
int mysql_change_user (string usuario, string password [, string base_de_datos [, int identificador_de_enlace]])
```

mysql_change_user() cambia el usuario conectado en la actual conexión activa, o si se especifica, en la conexión determinada por el identificador de enlace. Si se especifica la base de datos, esta será la base por defecto después del cambio de usuario. Si la nueva combinación de usuario/ password no está autorizada, el usuario actualmente conectado permanece activo.

Nota: Esta función fue introducida en PHP 3.0.13 y requiere MySQL 3.23.3 o superior.

mysql_close (PHP 3, PHP 4)

cierra el enlace con MySQL

```
int mysql_close ([int identificador_de_enlace])
```

Devuelve: verdadero si éxito, falso si error.

mysql_close() cierra el enlace con la base MySQL que está asociada con el identificador de enlace especificado. Si no se especifica el identificador de enlace, se asume por defecto el último enlace.

Nota: Normalmente no es necesario ya que las aperturas no-persistentes son cerradas automáticamente al final de la ejecución del script.

mysql_close() no cerrará los enlaces persistentes generados con **mysql_pconnect()**.

Ejemplo 1. Ejemplo de MySQL close

```
<?php
$link = mysql_connect ("kraemer", "marliesle", "secret") {
    or die ("Could not connect");
```

```

    }
    print ("Connected successfully");
    mysql_close ($link);
?>

```

Ver también: **mysql_connect()**, y **mysql_pconnect()**.

mysql_connect (PHP 3, PHP 4)

Abre una conexión a un servidor MySQL

```
int mysql_connect ([string hostname [:puerto] [:/camino/al/socket] [, string usuario [, string password]]])
```

Devuelve: Un identificador de enlace positivo si tiene éxito, o falso si error.

mysql_connect() establece una conexión a un servidor MySQL. Todos los argumentos son opcionales, y si no hay, se asumen los valores por defecto ('localhost', usuario propietario del proceso del servidor, password vacía).

El *hostname* puede incluir también un número de puerto . ej. "hostname:puerto" o un camino al socket ej. ":/camino/al/socket" para localhost.

Nota: Soporte para ":puerto" fue añadido en PHP 3.0B4.

Soporte para ":/camino/al/socket" fue añadido en PHP 3.0.10.

En el caso de que se haga una llamada a **mysql_connect()** con los mismos argumentos, no se establecerá un nuevo enlace, sino que se devolverá el enlace ya abierto.

El enlace al servidor será cerrado tan pronto como la ejecución del script finalice, a menos que se cierre antes explicitamente llamando a **mysql_close()**.

Ejemplo 1. Ejemplo de MySQL connect

```
<?php
$link = mysql_connect ("kraemer", "marliesle", "secret") {
    or die ("Could not connect");
}
print ("Connected successfully");
mysql_close ($link);
?>
```

Ver también : **mysql_pconnect()**, y **mysql_close()**.

mysql_create_db (PHP 3, PHP 4)

Crea una base MySQL

```
int mysql_create_db (string base_de_datos [, int identificador_de_enlace])
```

mysql_create_db() intenta crear una base nueva en el servidor asociado al identificador de enlace.

Ejemplo 1. Ejemplo de MySQL create

```
<?php
$link = mysql_pconnect ("kron", "jutta", "geheim") {
    or die ("Could not connect");
}
if (mysql_create_db ("my_db")) {
    print ("Database created successfully\n");
} else {
    printf ("Error creating database: %s\n", mysql_error ());
}
?>
```

Por razones de compatibilidad puede usarse **mysql_createdb()** igualmente.

Ver también: **mysql_drop_db()**.

mysql_data_seek (PHP 3, PHP 4)

Mueve el puntero interno

```
int mysql_data_seek (int id_resultado, int numero_de_fila)
```

Devuelve: verdadero si exito, falso si error.

mysql_data_seek() mueve el puntero de fila interno a la fila especificada para el identificador de resultado. La próxima llamada a **mysql_fetch_row()** devolverá esa fila.

numero_de_fila empieza en 0.

Ejemplo 1. Ejemplo de MySQL data seek

```
<?php
$link = mysql_pconnect ("kron", "jutta", "geheim") {
    or die ("Could not connect");
}

mysql_select_db ("samp_db") {
    or die ("Could not select database");
}

$query = "SELECT last_name, first_name FROM friends";
$result = mysql_query ($query) {
    or die ("Query failed");
}

# fetch rows in reverse order

for ($i = mysql_num_rows ($result) - 1; $i >=0; $i--) {
    if (!mysql_data_seek ($result, $i)) {
        printf ("Cannot seek to row %d\n", $i);
        continue;
    }

    if (!$row = mysql_fetch_object ($result)))
        continue;

    printf ("%s %s<BR>\n", $row->last_name, $row->first_name);
}
```

```
    mysql_free_result ($result);
?>
```

mysql_db_query (PHP 3, PHP 4)

Envia una sentencia MySQL al servidor

```
int mysql_db_query (string base_de_datos, string sentencia [, int identificador_de_enlace])
```

Devuelve: Un identificador de resultado positivo o falso si error.

mysql_db_query() selecciona una base y ejecuta una sentencia en ella. Si el identificador de enlace no ha sido especificado, la función intenta encontrar un enlace abierto al servidor MySQL y si no lo encuentra, intentará crear uno como si fuera llamado **mysql_connect()** sin argumentos

Ver también**mysql_connect()**.

Por razones de compatibilidad puede usarse **mysql()** igualmente.

mysql_drop_db (PHP 3, PHP 4)

Borra una base de datos MySQL

```
int mysql_drop_db (string base_de_datos [, int identificador_de_enlace])
```

Devuelve: verdadero si éxito, falso si error.

mysql_drop_db() intenta suprimir una base de datos completa del servidor asociado al identificador de enlace.

Ver también: **mysql_create_db()**. Por razones de compatibilidad puede usarse **mysql_dropdb()** igualmente.

mysql_errno (PHP 3, PHP 4)

Deuelve el número del mensaje de error de la última operación MySQL

```
int mysql_errno ([int identificador_de_enlace])
```

Los errores devueltos por mySQL no indican los warnings. Usar estas funciones para encontrar el número de error.

```
<?php
mysql_connect("marliesle");
echo mysql_errno()." : ".mysql_error()."<BR>";
mysql_select_db("nonexistentdb");
echo mysql_errno()." : ".mysql_error()."<BR>";
$conn = mysql_query("SELECT * FROM nonexistenttable");
echo mysql_errno()." : ".mysql_error()."<BR>?
?>
```

Ver también: **mysql_error()**

mysql_error (PHP 3, PHP 4)

Devuelve el texto del mensaje de error de la última operación MySQL

```
string mysql_error ([int identificador_de_enlace])
```

Los errores devueltos por mySQL no indican los warnings. Usar estas funciones para encontrar el número de error.

```
<?php
mysql_connect("marliesle");
echo mysql_errno().".".mysql_error()."<BR>";
mysql_select_db("nonexistentdb");
echo mysql_errno().".".mysql_error()."<BR>";
$conn = mysql_query("SELECT * FROM nonexistenttable");
echo mysql_errno().".".mysql_error()."<BR>";
?>
```

Ver también: **mysql_errno()**

mysql_fetch_array (PHP 3, PHP 4)

Extrae la fila de resultado como una matriz asociativa

```
array mysql_fetch_array (int id_resultado [, int tipo_de_resultado])
```

Devuelve una matriz que corresponde a la sentencia extraída, o falso si no quedan más filas.

mysql_fetch_array() es una versión extendida de **mysql_fetch_row()**. Además de guardar los datos en el índice numérico de la matriz, guarda también los datos en los índices asociativos, usando el nombre de campo como clave.

Si dos o más columnas del resultado tienen el mismo nombre de campo, la última columna toma la prioridad. Para acceder a la(s) otra(s) columna(s) con el mismo nombre, se debe especificar el índice numérico o definir un alias para la columna.

```
select t1.f1 as foo t2.f1 as bar from t1, t2
```

La función **mysql_fetch_array()** no es significativamente más lenta que **mysql_fetch_row()**, sin embargo tiene un valor añadido importante.

El segundo argumento opcional *tipo_de_resultado* en **mysql_fetch_array()** es una constante y puede tomar los valores siguientes: **MYSQL_ASSOC**, **MYSQL_NUM**, y **MYSQL_BOTH**. (Esta funcionalidad fue añadida en PHP 3.0.7)

Para más detalles, ver también **mysql_fetch_row()**.

Ejemplo 1. mysql fetch array

```
<?php
mysql_connect($host,$user,$password);
$result = mysql_db_query("database","select * from table");
while($row = mysql_fetch_array($result)) {
    echo $row["user_id"];
    echo $row["fullname"];
}
mysql_free_result($result);
?>
```

mysql_fetch_field (PHP 3, PHP 4)

Extrae la información de una columna y la devuelve como un objeto.

```
object mysql_fetch_field ( int id_resultado [, int salto] )
```

Devuelve un objeto que contiene la información del campo.

Puede usarse **mysql_fetch_field()** para obtener información sobre campos en un resultado. Si no se especifica el salto, se extrae el siguiente campo que todavía no ha sido extraído. con **mysql_fetch_field()**.

Las propiedades del objeto son:

- name - nombre de la columna
- table - nombre de la tabla a la que pertenece la columna
- max_length - longitud máxima de la columna
- not_null - 1 si la columna no puede contener un valor nulo
- primary_key - 1 si la columna es clave primaria
- unique_key - 1 si la columna es clave única
- multiple_key - 1 si la columna es clave no única
- numeric - 1 si la columna es numérica
- blob - 1 si la columna es un BLOB
- type - el tipo de la columna
- unsigned - 1 si la columna es unsigned
- zerofill - 1 si la columna es zero-filled

Ver también **mysql_field_seek()**

mysql_fetch_lengths (PHP 3, PHP 4)

Devuelve la longitud de cada salida en un resultado

```
array mysql_fetch_lengths ( int id_resultado )
```

Devuelve: Una matriz que contiene las longitudes de cada campo de la última fila extraída por **mysql_fetch_row()**, o falso si error.

mysql_fetch_lengths() almacena las longitudes de cada columna en la última fila devuelta por **mysql_fetch_row()**, **mysql_fetch_array()**, y **mysql_fetch_object()** en una matriz, empezando por 0.

Ver también: **mysql_fetch_row()**.

mysql_fetch_object (PHP 3, PHP 4)

Extrae una fila de resultado como un objeto

```
object mysql_fetch_object (int id_resultado [, int tipo_de_resultado])
```

Devuelve un objeto con las propiedades que corresponden a la última fila extraída, o falso si no quedan más filas.

mysql_fetch_object() es similar a **mysql_fetch_array()**, con la diferencia que un objeto es devuelto en lugar de una matriz. Indirectamente, quiere decir que solo se puede acceder a los datos por el nombre del campo, y no por su posición.

El argumento opcional *tipo_de_resultado* es una constante y puede tomar los valores siguientes: MYSQL_ASSOC, MYSQL_NUM, y MYSQL_BOTH.

La función es idéntica a **mysql_fetch_array()**, y casi tan rápida como **mysql_fetch_row()** (la diferencia es insignificante).

Ejemplo 1. mysql fetch object

```
<?php
mysql_connect($host,$user,$password);
$result = mysql_db_query("database","select * from table");
while($row = mysql_fetch_object($result)) {
    echo $row->user_id;
    echo $row->fullname;
}
mysql_free_result($result);
?>
```

Ver también: **mysql_fetch_array()** y **mysql_fetch_row()**.

mysql_fetch_row (PHP 3, PHP 4)

Devuelve una fila de resultado como matriz

```
array mysql_fetch_row (int id_resultado)
```

Devuelve: Una matriz que corresponde a la fila seleccionada, o falso si no quedan más líneas.

mysql_fetch_row() selecciona una fila de datos del resultado asociado al identificador de resultado especificado. La fila es devuelta como una matriz. Cada columna del resultado es guardada en un offset de la matriz, empezando por el offset 0.

La llamada a **mysql_fetch_row()** debería devolver la próxima fila del resultado, o falso si no quedan más filas.

Ver también: **mysql_fetch_array()**, **mysql_fetch_object()**, **mysql_data_seek()**, **mysql_fetch_lengths()**, and **mysql_result()**.

mysql_field_name (PHP 3, PHP 4)

Devuelve el nombre del campo especificado en un resultado

```
string mysql_field_name (int id_resultado, int indice_del_campo)
```

mysql_field_name() devuelve el nombre del campo especificado. Los argumentos de la función son el identificador de resultado y el indice del campo. Por ejemplo: `mysql_field_name($result, 2);`

Devolverá el nombre del segundo campo asociado al identificador de resultado.

Por razones de compatibilidad puede usarse tambien **mysql_fieldname()**.

mysql_field_seek (PHP 3, PHP 4)

Asigna el puntero del resultado al offset del campo especificado

```
int mysql_field_seek (int id_resultado, int offset_del_campo)
```

Busca el offset del campo especificado. Si la próxima llamada a **mysql_fetch_field()** no incluye un offset de campo, se devolverá ese campo.

Ver también: **mysql_fetch_field()**.

mysql_field_table (PHP 3, PHP 4)

Devuelve el nombre de la tabla donde esta el campo especificado

```
string mysql_field_table (int id_resultado, int offset_del_campo)
```

Devuelve el nombre de la tabla del campo. Por razones de compatibilidad puede usarse tambien **mysql_fieldtable()**.

mysql_field_type (PHP 3, PHP 4)

Devuelve el tipo del campo especificado en un resultado

```
string mysql_field_type (int id_resultado, int offset_del_campo)
```

mysql_field_type() es similar a la función **mysql_field_name()**. Los argumentos son identicos, pero se devuelve el tipo de campo. El tipo sera "int", "real", "string", "blob", o otros detallados en la documentación de MySQL.

Ejemplo 1. mysql field types

```
<?php
mysql_connect("localhost:3306");
mysql_select_db("wisconsin");
$result = mysql_query("SELECT * FROM onek");
$fields = mysql_num_fields($result);
$rows = mysql_num_rows($result);
$i = 0;
$table = mysql_field_table($result, $i);
echo "Your '$table' table has ".$fields." fields and ".$rows." records <BR>";
echo "The table has the following fields <BR>";
while ($i < $fields) {
    $type = mysql_field_type ($result, $i);
    $name = mysql_field_name ($result, $i);
    $len = mysql_field_len ($result, $i);
    $flags = mysql_field_flags ($result, $i);
    echo $type." ".$name." ".$len." ".$flags."<BR>";
```

```

    $i++;
}
mysql_close();
?>

```

Por razones de compatibilidad puede usarse tambien **mysql_fieldtype()**.

mysql_field_flags (PHP 3, PHP 4)

Devuelve los flags asociados con el campo especificado en un resultado

```
string mysql_field_flags (int id_resultado, int offset_del_campo)
```

mysql_field_flags() devuelve los flags del campo especificado. Cada flag es devuelto como una palabra y estan separados un unico espacio, se puede dividir el resultado devuelto utilizando **explode()**.

Los siguientes flags pueden ser devueltos si tu versión de MySQL los soporta: "not_null", "primary_key", "unique_key", "multiple_key", "blob", "unsigned", "zerofill", "binary", "enum", "auto_increment", "timestamp".

Por razones de compatibilidad puede usarse tambien **mysql_fieldflags()**.

mysql_field_len (PHP 3, PHP 4)

Devuelve la longitud del campo especificado

```
int mysql_field_len (int id_resultado, int offset_del_campo)
```

mysql_field_len() devuelve la longitud del campo especificado. Por razones de compatibilidad puede usarse tambien **mysql_fieldlen()**.

mysql_free_result (PHP 3, PHP 4)

Libera la memoria del resultado

```
int mysql_free_result (int id_resultado)
```

mysql_free_result() solo necesita ser llamada si te preocupa usar demasiado memoria durante la ejecución de tu script. Toda la memoria del resultado especificado en el parametro del identificador de resultado sera automaticamente liberada.

Por razones de compatibilidad puede usarse tambien **mysql_freeresult()**.

mysql_insert_id (PHP 3, PHP 4)

Devuelve el identificador generado en la última llamada a INSERT

```
int mysql_insert_id ([int identificador_de_enlace])
```

mysql_insert_id() devuelve el identificador generado para un campo de tipo AUTO_INCREMENTED. Se devolverá el identificador generado por el último INSERT para el *identificador_de_enlace*. Si no se especifica el *identificador_de_enlace*, se asume por defecto el último enlace abierto.

mysql_list_fields (PHP 3, PHP 4)

Lista los campos del resultado de MySQL

```
int mysql_list_fields (string base_de_datos, string tabla [, int identificador_de_enlace])
```

mysql_list_fields() lista información sobre la tabla. Los argumentos son la base de datos y el nombre de la tabla. Se devuelve un puntero que puede ser usado por las funciones **mysql_field_flags()**, **mysql_field_len()**, **mysql_field_name()**, y **mysql_field_type()**.

Un identificador de resultado es un entero positivo. La función devuelve -1 si se produce un error. Una cadena de caracteres describiendo el error será introducida en \$phperrmsg, y a menos que la función sea llamada como @mysql() el literal del error también será impreso.

Por razones de compatibilidad puede usarse también **mysql_listfields()**.

mysql_list_dbs (PHP 3, PHP 4)

Lista las bases de datos disponibles en el servidor MySQL

```
int mysql_list_dbs ([int identificador_de_enlace])
```

mysql_list_dbs() devuelve un puntero de resultado que contiene las bases disponibles en el actual demonio mysql. Utiliza la función **mysql_tablename()** para explotar el puntero de resultado.

Por razones de compatibilidad puede usarse también **mysql_listdbs()**.

mysql_list_tables (PHP 3, PHP 4)

Lista las tablas en una base de datos MySQL

```
int mysql_list_tables (string base_de_datos [, int identificador_de_enlace])
```

mysql_list_tables() toma el nombre de la base y devuelve un puntero de resultado como la función **mysql_db_query()**. La función **mysql_tablename()** debe ser usada para extraer los nombres de las tablas del puntero.

Por razones de compatibilidad puede usarse también **mysql_listtables()**. can also be used.

mysql_num_fields (PHP 3, PHP 4)

devuelve el número de campos de un resultado

```
int mysql_num_fields (int id_resultado)
```

mysql_num_fields() devuelve el número de campos de un identificador de resultado.

Ver tambien: **mysql_db_query()**, **mysql_query()**, **mysql_fetch_field()**, **mysql_num_rows()**.

Por razones de compatibilidad puede usarse tambien **mysql_numfields()**.

mysql_num_rows (PHP 3, PHP 4)

Devuelve el numero de filas de un resultado

```
int mysql_num_rows (int id_resultado)
```

mysql_num_rows() Devuelve el numero de filas de un identificador de resultado.

Ver tambien: **mysql_db_query()**, **mysql_query()** and, **mysql_fetch_row()**.

Por razones de compatibilidad puede usarse tambien **mysql_numrows()**.

mysql_pconnect (PHP 3, PHP 4)

Abre una conexión persistente al servidor MySQL

```
int mysql_pconnect ([string hostname [:puerto] [:/camino/al/socket] [, string usuario [, string password]]])
```

Devuelve: un identificador de enlace persistente, o falso si se produce un error.

mysql_pconnect() establece una conexión a un servidor MySQL. Todos los argumentos son opcionales, y si no existen, se asumen los valores por defecto ('localhost', nombre del usuario propietario del proceso, password vacia).

El hostname puede incluir un numero de puerto. ej. "hostname:port" o un camino al socket ej. ":/camino/al/socket" para el puerto para el host local.

Nota: Soporte para ":puerto" fue añadido en 3.0B4.

Soporte para ":/camino/al/socket" fue añadido en 3.0.10.

mysql_pconnect() actua como **mysql_connect()** con dos diferencias fundamentales.

Primero, durante la conexión, la función intenta primero encontrar un enlace persistente abierto con el mismo host, usuario y password. Si lo encuentra, devuelve el identificador de enlace en lugar de abrir otra conexión.

Segundo, la conexión no sera cerrada cuando acabe la ejecución del script. El enlace permanecera abierta para ser usado en el futuro (**mysql_close()** will not cierra el enlace establecido con **mysql_pconnect()**).

Este tipo de enlaces son llamados 'persistentes'.

mysql_query (PHP 3, PHP 4)

Envia una sentencia SQL a MySQL

```
int mysql_query (string sentencia [, int identificador_de_enlace])
```

mysql_query() envia una sentencia a la base activa en el servidor asociado al identificador de enlace. Si no es especificado un *identificador_de_enlace*, se asumira el ultilmo enlace abierto. Si no hay ningun enlace abierto, la función intenta establecer un enlace como si se llamara función **mysql_connect()** sin argumentos, y lo utiliza.

La sentencia no puede terminar por punto y coma.

mysql_query() devuelve TRUE (no-cero) o FALSE para indicar si la sentencia se ha ejecutado correctamente o no. Un valor TRUE significa que la sentencia era correcta y pudo ser ejecutada en el servidor. No indica nada sobre el numero de fila devueltas. Es perfectamente posible que la sentencia se ejecute correctamente pero que no devuelva ninguna fila.

La siguiente sentencia es invalida sintacticamente, asi que **mysql_query()** falla y devuelve FALSE:

Ejemplo 1. mysql_query()

```
<?php
$result = mysql_query ("SELECT * WHERE 1=1")
    or die ("Invalid query");
?>
```

La siguiente sentencia es invalida semanticamente si *my_col* no es una columna de la tabla *my_tbl*, asi que **mysql_query()** falla y devuelve FALSE:

Ejemplo 2. mysql_query()

```
<?php
$result = mysql_query ("SELECT my_col FROM my_tbl")
    or die ("Invalid query");
?>
```

mysql_query() fallara tambien y devolvera FALSE si no se tiene el permiso de acceso a la tabla especificada en la sentencia.

Asumiendo la sentencia tenga exito, se puede llamar a **mysql_affected_rows()** para saber cuantas filas fueron afectadas (para DELETE, INSERT, REPLACE, o UPDATE) Para las sentencias SELECT, **mysql_query()** devuelve un nuevo identificador de resultado que se puede pasar a **mysql_result()**. Cuando se acabe de utilizar el resultado, se pueden liberar los recursos asociados utilizando **mysql_free_result()**.

Ver tambien: **mysql_affected_rows()**, **mysql_db_query()**, **mysql_free_result()**, **mysql_result()**, **mysql_select_db()**, and **mysql_connect()**.

mysql_result (PHP 3, PHP 4)

Devuelve datos de un resultado

```
int mysql_result (int id_resultado, int numero_de_fila [, mixed campo])
```

mysql_result() devuelve el contenido de una celda de un resultado MySQL. El campo argumento puede ser el nombre del campo o el offset o tabla.nombre_del_campo. Si el nombre de la columna tiene un alias ('select foo as bar from...'), utilice el alias en lugar del nombre de la columna.

Cuando se trabaja un un gran resultado, debe considerarse la utilizacion de una funcion que devuelva una fila entera ya que estas funciones son MUCHO mas rapidas que **mysql_result()**. Tambien, especificando un offset numerico en lugar del nombre del campo, la ejecucion sera mas rapida.

Las llamadas a **mysql_result()** no deben mezclarse con llamadas a las otras sentencias que trabajan con un identificador de resultado.

Alternativas recomendadas: **mysql_fetch_row()**, **mysql_fetch_array()**, y **mysql_fetch_object()**.

mysql_select_db (PHP 3, PHP 4)

Selecciona un base de datos MySQL

```
int mysql_select_db (string base_de_datos [, int identificador_de_enlace])
```

Devuelve : true si exito, false si error.

mysql_select_db() establece la base activa que estara asociada con el identificador de enlace especificado. Si no se especifica un identificador de enlace, se asume el ultimo enlace abierto. Si no hay ningun enlace abierto, la función intentara establecer un enlace como si se llamara a **mysql_connect()**.

Toda llamada posterior a **mysql_query()** utilizara la base activada.

Ver tambien: **mysql_connect()**, **mysql_pconnect()**, and **mysql_query()**.

Por razones de compatibilidad puede usarse tambien **mysql_selectdb()**.

mysql_tablename (PHP 3, PHP 4)

Devuelve el nombre de la tabla de un campo

```
string mysql_tablename (int id_resultado, int i)
```

mysql_tablename() toma un puntero de resultado devuelto por **mysql_list_tables()** asi como un indice (integer) y devuelve el nombre de una tabla. Se puede usar la función **mysql_num_rows()** para determinar el nombre de tablas en el puntero de resultado.

Ejemplo 1. Mysql_tablename() Example

```
<?php
mysql_connect ("localhost:3306");
$result = mysql_list_tables ("wisconsin");
$i = 0;
while ($i < mysql_num_rows ($result)) {
    $tb_names[$i] = mysql_tablename ($result, $i);
    echo $tb_names[$i] . "<BR>";
    $i++;
}
?>
```


XLVII. Funciones de Red

checkdnsrr (PHP 3, PHP 4)

Comprueba registros DNS correspondientes a nombres de máquinas en Internet o direcciones IP.

```
int checkdnsrr (string host [, string type])
```

Busca en DNS entradas del tipo *type* correspondientes a *host*. Devuelve verdadero si encuentra algún registro; devuelve falso si no encuentra ninguno o sucedió algún error.

type puede ser: A, MX, NS, SOA, PTR, CNAME, o ANY. Por defecto es MX.

host puede ser o la dirección IP de la forma xxxx.xxxx.xxxx.xxxx o el nombre de la máquina.

Ver también **getmxrr()**, **gethostbyaddr()**, **gethostbyname()**, **gethostbynamel()**, y **named(8)** en las páginas del manual.

closelog (PHP 3, PHP 4)

cierra la conexión con el logger del sistema

```
int closelog(void);
```

closelog() cierra el descriptor que se está usando para escribir en el logger del sistema. El uso de **closelog()** es opcional.

debugger_off (PHP 3)

deshabilita el depurador interno de PHP

```
int debugger_off(void);
```

Deshabilita el depurador interno de PHP. El depurador está aún en desarrollo.

debugger_on (PHP 3)

habilita el depurador interno de PHP

```
int debugger_on (string address)
```

Habilita el depurador interno de PHP, conectándolo a *address*. El depurador esta aún en desarrollo.

fsockopen (PHP 3, PHP 4)

Abre una conexión de dominio Internet o Unix via sockets.

```
int fsockopen (string hostname, int port [, int errno [, string errstr [, double timeout]]])
```

Inicia una conexión de dominio Internet (AF_INET) o Unix (AF_UNIX). Para el dominio Internet, abrirá una conexión TCP hacia el ordenador *hostname* en el puerto *port*. Para el dominio Unix, *hostname* se usará como ruta al socket,

port debe ser 0 para este caso. El parámetro opcional *timeout* se puede usar para especificar un timeout en segundos para establecer la conexión.

fsockopen() devuelve un puntero a fichero, el cual se puede usar junto con las otras funciones de ficheros (como **fgets()**, **fgetss()**, **fputs()**, **fclose()**, **feof()**).

Si la llamada falla, esta devolverá falso y si los parámetros opcionales *errno* y *errstr* están presentes, indicarán el error del sistema que ocurrió en la llamada **connect()**. Si *errno* es 0 y la función devolvió falso, nos indica que el error ocurrió antes de la llamada **connect()**. Esto es debido principalmente a problemas inicializando el socket. Observe que los argumentos *errno* y *errstr* deben ser pasados por referencia.

Dependiendo del entorno, el dominio Unix o el parámetro opcional, *timeout* puede no estar disponible.

Por defecto, el socket será abierto en modo de bloqueo. Puede cambiarlo a modo de no bloqueo usando **set_socket_blocking()**.

Ejemplo 1. ejemplo con fsockopen

```
$fp = fsockopen("www.php.net", 80, &$errno, &$errstr, 30);
if(!$fp) {
echo "$errstr ($errno)<br>\n";
} else {
fputs($fp,"GET / HTTP/1.0\n\n");
while(!feof($fp)) {
echo fgets($fp,128);
}
fclose($fp);
}
```

Ver también: **pfsckopen()**

gethostbyaddr (PHP 3, PHP 4)

Obtiene el nombre de una máquina en Internet mediante su dirección IP.

```
string gethostbyaddr (string ip_address)
```

Devuelve el nombre del ordenador conectado a Internet especificado por el parámetro *ip_address*. Si ocurre un error, devuelve *ip_address*.

Ver también **gethostbyname()**.

gethostbyname (PHP 3, PHP 4)

Obtiene la dirección IP correspondiente al nombre de una máquina conectada a Internet.

```
string gethostbyname (string hostname)
```

Devuelve la dirección IP de una máquina conectada a Internet especificada por *hostname*.

Ver también **gethostbyaddr()**.

gethostbynameI (PHP 3, PHP 4)

Obtiene una lista de direcciones IP correspondientes a los nombres de máquinas conectadas a Internet.

```
array gethostbynameI (string hostname)
```

Devuelve una lista de direcciones IP pertenecientes a ordenadores especificados por *hostname*.

Ver también **gethostbyname()**, **gethostbyaddr()**, **checkdnsrr()**, **getmxrr()**, y **named(8)** en las páginas del manual.

getmxrr (PHP 3, PHP 4)

Obtiene registros MX correspondientes a una máquina conectada a Internet.

```
int getmxrr (string hostname, array mxhosts [, array weight])
```

Busca DNS de registros MX correspondientes a *hostname*. Devuelve verdadero si encuentra algún registro; devuelve falso si no encuentra ninguno o se produce un error.

La lista de registros MX encontrados se colocan en el array *mxhosts*. Si se proporciona el array *weight*, se llenará con la información obtenida.

Ver también **checkdnsrr()**, **gethostbyname()**, **gethostbynameI()**, **gethostbyaddr()**, y **named(8)** de las páginas del manual.

getprotobynumber (PHP 4 >= 4.0b4)

Obtiene el número asociado al nombre del protocolo

```
int getprotobynumber (string name)
```

getprotobynumber() devuelve el número asociado al nombre del protocolo *name* del fichero /etc/protocols. Ver también **getprotobynumber()**.

getprotobynumber (PHP 4 >= 4.0b4)

obtiene el nombre asociado al número de protocolo

```
string getprotobynumber (int number)
```

getprotobynumber() devuelve el nombre del protocolo asociado al *number* del protocolo en el fichero /etc/protocols. Ver también **getprotobynumber()**.

getservbyname (PHP 4 >= 4.0b4)

obtiene el número del puerto asociado al servicio Internet especificado

```
int getservbyname (string service, string protocol)
```

getservbyname() devuelve el puerto que corresponde al *service* especificado por el *protocol* en /etc/services. *protocol* puede ser `tcp` o `udp`. Ver también **getservbyport()**.

getservbyport (PHP 4 >= 4.0b4)

obtiene el servicio Internet que correspondiente al puerto del protocolo especificado

```
string getservbyport (int port, string protocol)
```

getservbyport() devuelve el servicio Internet asociado al *port* para el *protocol* especificado en /etc/services. *protocol* puede ser `tcp` o `udp`. Ver también **getservbyname()**.

openlog (PHP 3, PHP 4)

abre una conexión con el logger del sistema

```
int openlog (string ident, int option, int facility)
```

openlog() abre una conexión con el logger del sistema. La cadena *ident* se añade a cada mensaje. Los valores de *option* y *facility* se exponen en la siguiente sección. El uso de **openlog()** es opcional; Esta será llamada automáticamente por **syslog()** si fuera necesario, en este caso *ident* valdrá por defecto `false`. Ver también **syslog()** y **closelog()**.

pfsockopen (PHP 3>= 3.0.7, PHP 4)

Abre conexiones persistentes de dominio Internet o Unix.

```
int pfsockopen (string hostname, int port [, int errno [, string errstr [, int timeout]]])
```

Esta función se comporta exactamente como **fsockopen()** con la diferencia que la conexión no se cierra después de que termine el script. Esta es la versión persistente de **fsockopen()**.

set_socket_blocking (PHP 3, PHP 4)

Set blocking/non-blocking modo de un socket

```
int set_socket_blocking (int socket descriptor, int mode)
```

Si *mode* es falso, el socket estará descriptor will be switched to non-blocking mode, y si es true, este pasará a modo bloqueo. Esto afecta a llamadas como **fgets()** que leen del socket. En el modo de no-bloqueo una llamada **fgets()** devolverá la información en el acto, mientras que en modo bloqueo esperará a que la información esté disponible en el socket.

syslog (PHP 3, PHP 4)

genera un mensaje de sistema

```
int syslog (int priority, string message)
```

syslog() genera un mensaje que será distribuido por el logger del sistema. *priority* es una combinación de la facility y el level, los valores se indicarán en la sección siguiente. El argumento restante es el mensaje a enviar, excepto que los dos caracteres %m sean reemplazados por la cadena de error (strerror) correspondiente al valor actual de errno.

Más información acerca de syslog se puede encontrar en las páginas del manual en equipos Unix.

En Windows NT, el servicio syslog es emulado usando el Log de Eventos.

XLVIII. ODBC functions

odbc_autocommit (PHP 3>= 3.0.6, PHP 4)

Interruptor de comportamiento de auto-entrega

```
int odbc_autocommit (int connection_id [, int OnOff])
```

Sin el parametro *OnOff*, esta funcion devuelve el estado de auto-entrega para *connection_id*. Devuelve true si auto-entrega esta habilitado, y false si no lo esta o ha ocurrido un error.

Si *OnOff* es true, auto-entrega esta activado, si es false auto-entrega esta desactivado. Devuelve true cuando se cumple, false cuando falla.

Por defecto, auto-entrega es para una conexion. Desabilitar auto-entrega es como comenzar una transaccion.

Ver tambien **odbc_commit()** y **odbc_rollback()**.

odbc_binmode (PHP 3>= 3.0.6, PHP 4)

Manejo de campos de datos binarios

```
int odbc_binmode (int result_id, int mode)
```

(Elementos afectados ODBC SQL: BINARY, VARBINARY, LONGVARBINARY)

- ODBC_BINMODE_PASSTHRU: Paso a traves de datos binarios
- ODBC_BINMODE_RETURN: Devuelve como es
- ODBC_BINMODE_CONVERT: Devuelve convertido en caracter

Cuando los datos binarios en SQL son convertidos a datos caracter en C, cada byte (8 bits) de datos fuente es representada como dos caracteres en ASCII. Esos caracteres son la representacion en ASCII de los numeros en su forma Hexadecimal. Por ejemplo, un 00000001 binario es convertido a "01" y un 11111111 binario es convertido a "FF".

Tabla 1. Manejo de LONGVARBINARY

modo binario	longreadlen	resultado
ODBC_BINMODE_PASSTHRU	0	passthru
ODBC_BINMODE_RETURN	0	passthru
ODBC_BINMODE_CONVERT	0	passthru
ODBC_BINMODE_PASSTHRU	0	passthru
ODBC_BINMODE_PASSTHRU	>0	passthru
ODBC_BINMODE_RETURN	>0	Devuleve como es
ODBC_BINMODE_CONVERT	>0	Devuelve como caracter

Si usamos **odbc_fetch_into()**, passthru significara que una cadena vacia es devuelta por esos campos.

Si *result_id* es 0, las definiciones se aplican por defecto para nuevos resultados.

Nota: Por defecto, longreadlen es 4096 y el modo binario por defecto es ODBC_BINMODE_RETURN. El manejo de campos binarias largas tambien esta afectado por **odbc_longreadlen()**

odbc_close (PHP 3>= 3.0.6, PHP 4)

Cierra una conexion ODBC

```
void odbc_close (int connection_id)
```

odbc_close() cerrara la conexion al servidor de bases datos asociado con el identificador de conexion dado.

Nota: Esta funcion fallara si hay transacciones abiertas sobre esta conexion. La conexion quedara abierta en ese caso.

odbc_close_all (PHP 3>= 3.0.6, PHP 4)

Cierra todas las conexiones ODBC

```
void odbc_close_all(void);
```

odbc_close_all() cerrara todas las conexiones a servidor(es) de bases de datos.

Nota: Esta funcion fallara si hay transacciones abiertas sobre esta conexion. La conexion quedara abierta en ese caso.

odbc_commit (PHP 3>= 3.0.6, PHP 4)

Entrega una transaccion ODBC

```
int odbc_commit (int connection_id)
```

Devuelve: `true` si la operacion se realiza con exito, `false` si falla. Todas las transacciones pendientes sobre *connection_id* son entregadas.

odbc_connect (PHP 3>= 3.0.6, PHP 4)

Conecta a una fuente de datos

```
int odbc_connect (string dsn, string user, string password [, int cursor_type])
```

Devuelve una conexion ODBC id, o 0 (`false`) cuando ocurre un error.

La conexion id devuelta por estas funciones es necesaria para otras funciones ODBC. Se pueden tener multiples conexiones abiertas a la vez. El opcional cuarto parametro asigna el tipo de cursor que va a ser usado para esta conexion. Este parametro normalmente no es necesario, pero puede ser util para trabajar sobre problemas con algunos drivers ODBC.

Con algunos drivers ODBC, si ejecutamos un procedimiento complejo, este puede fallar con un error similar a: "Cannot open a cursor on a stored procedure that has anything other than a single select statement in it". Usando

SQL_CUR_USE_ODBC se puede evitar ese error. Algunos drivers tampoco soportan el parametro row_number en **odbc_fetch_row()**. SQL_CUR_USE_ODBC tambien podria ayudar en ese caso.

Las siguientes constantes son definidas por tipos de cursor:

- SQL_CUR_USE_IF_NEEDED
- SQL_CUR_USE_ODBC
- SQL_CUR_USE_DRIVER
- SQL_CUR_DEFAULT

Para conexiones persistentes ver **odbc_pconnect()**.

odbc_cursor (PHP 3>= 3.0.6, PHP 4)

Toma un nombre de cursor

```
string odbc_cursor (int result_id)
```

odbc_cursor devolvera un nombre de cursor para el result_id dado.

odbc_do (PHP 3>= 3.0.6, PHP 4)

sinonimo de **odbc_exec()**

```
string odbc_do (int conn_id, string query)
```

odbc_do ejecutara una consulta (query) sobre la conexion dada

odbc_exec (PHP 3>= 3.0.6, PHP 4)

Prepara o ejecuta una declaracion SQL

```
int odbc_exec (int connection_id, string query_string)
```

Devuelve false en caso de error. Devuelve un indetificador ODBC si el comando SQL fue ejecutado satisfactoriamente.

odbc_exec() enviara una declaracion SQL al servidor de bases de datos especificado por *connection_id*. Este parametro debe ser un indetificador valido devuelto por **odbc_connect()** o **odbc_pconnect()**.

Ver tambien: **odbc_prepare()** y **odbc_execute()** para ejecucion multiple de declaraciones SQL.

odbc_execute (PHP 3>= 3.0.6, PHP 4)

ejecuta una declaracion preparada

```
int odbc_execute (int result_id [, array parameters_array])
```

Ejecuta una declaracion preparada con **odbc_prepare()**. Devuelve `true` cuando la ejecucion es satisfactoria, `false` en otro caso. Introducir el vector `parameters_array` solo es necesario si realmente tenemos parametros en la declaracion.

odbc_fetch_into (PHP 3>= 3.0.6, PHP 4)

Busca un registro de resultados dentro de un vector

```
int odbc_fetch_into (int result_id [, int rownumber, array result_array])
```

Devuelve el numero de campos en el resultado; `false` on error. `result_array` debe ser pasado por referencia, pero puede ser de cualquier tipo, desde este sera convertido a tipo vector. El vector contendra el valor de campo inicial empezando en indice de vector 0.

odbc_fetch_row (PHP 3>= 3.0.6, PHP 4)

Busca un registro

```
int odbc_fetch_row (int result_id [, int row_number])
```

Si **odbc_fetch_row()** fue exitoso (there was a row), `true` is returned. If there are no more rows, `false` is returned.

odbc_fetch_row() busca un registro de datos que fue devuelta por **odbc_do()** / **odbc_exec()**. Despues de que **odbc_fetch_row()** sea llamado, se puede acceder a los campos de este registro con **odbc_result()**.

Si no se especifica `row_number`, **odbc_fetch_row()** intentara buscar el siguiente registro en los resultados. Lamar a **odbc_fetch_row()** con 0 sin `row_number` puede ser mezclado.

Para pasar a traves del resultado mas de una vez, se puede llamar a **odbc_fetch_row()** con `row_number` 1, y despues continuar haciendo **odbc_fetch_row()** sin `row_number` para revisar el resultado. Si un driver no admitiese busquedas de registros por numero, el parametro `row_number` seria ignorado.

odbc_field_name (PHP 3>= 3.0.6, PHP 4)

Devuelve el nombre de campo

```
string odbc_fieldname (int result_id, int field_number)
```

odbc_field_name() devolvera el nombre del campo almacenado en el numero de campo elegido dentro del identificador ODBC. La numeracion de campos comienza en 1. En caso de error devolveria `false`.

odbc_field_type (PHP 3>= 3.0.6, PHP 4)

Tipo de datos de un campo

```
string odbc_field_type (int result_id, int field_number)
```

odbc_field_type() Devolvera el tipo SQL de un campo referenciado por numero en el identificador ODBC. identifier. La numeracion de campos comienza en 1.

odbc_field_len (PHP 3>= 3.0.6, PHP 4)

Da la longitud de un campo

```
int odbc_field_len (int result_id, int field_number)
```

odbc_field_len() devolvera la longitud de un campo referenciado por numero en un identificador ODBC La numeracion de campos comienza en 1.

odbc_free_result (PHP 3>= 3.0.6, PHP 4)

recursos libres asociados con un resultado

```
int odbc_free_result (int result_id)
```

Always returns true.

odbc_free_result() solo necesita ser llamado en caso de preocupacion por demasiado uso de memoria cuando se ejecuta un script. Toda la memoria resultante quedara automaticamente liberada cuando el script finalice. Pero si es seguro que no se vaya a necesitar la informacion nada mas que en un script, se debera llamar a la funcion **odbc_free_result()**, y la memoria asociada con *result_id* sera liberada.

Nota: Si la auto-entrega no esta activada la (ver **odbc_autocommit()**) y se ejecuta **odbc_free_result()** antes de la entrega, todo queda pendiente de las transacciones que esten en lista.

odbc_longreadlen (PHP 3>= 3.0.6, PHP 4)

manejo de LONGITUD de columnas

```
int odbc_longreadlen (int result_id, int length)
```

(ODBC SQL tipos relacionados: LONG, LONGVARBINARY) El numero de bytes devueltos para PHP es controlado por el parametro length. Si es asignado a 0, la longitud del campo es pasado al cliente.

Nota: El manejo de campos LONGVARBINARY tambien esta afectado por **odbc_binmode()**

odbc_num_fields (PHP 3>= 3.0.6, PHP 4)

numero de campos de un resultado

```
int odbc_num_fields (int result_id)
```

odbc_num_fields() devolvera el numero de campos dentro de un ODBC. Esta funcion devolvera -1 en caso de error. El argumento es un identificador valido devuelto por **odbc_exec()**.

odbc_pconnect (PHP 3>= 3.0.6, PHP 4)

Abre una conexion permanente de base de datos

```
int odbc_pconnect (string dsn, string user, string password [, int cursor_type])
```

Devuelve un identificador de conexion ODBC o 0 (*false*) en caso de error. Esta funcion es **odbc_connect()**, excepto que la conexion no sea realmente cerrada cuando el script ha finalizado. Las respuestas futuras para una conexion con la misma combinacion *dsn*, *user*, *password* (via **odbc_connect()** y **odbc_pconnect()**) puede reusar la conexion permanente.

Nota: Las conexiones permanentes no tienen efecto si PHP es usado como programa CGI.

Para informacion acerca del paramentor opcional *cursor_type* ver la funcion **odbc_connect()**. Para mas informacion sobre conexiones permanentes, ir al apartado PHP FAQ.

odbc_prepare (PHP 3>= 3.0.6, PHP 4)

Prepara una declaracion para su ejecucion

```
int odbc_prepare (int connection_id, string query_string)
```

Devuelve *false* en caso de error.

Devuelve un identificador ODBC si el comando SQL esta preparado. El identificador resultante puede ser usado mas tarde para ejecutar la declaracion con **odbc_execute()**.

odbc_num_rows (PHP 3>= 3.0.6, PHP 4)

Numero de campos en un resultado

```
int odbc_num_rows (int result_id)
```

odbc_num_rows() devolvera el numero de registros de un ODBC. Esta funcion devolvera -1 en caso de error. Para declaraciones INSERT, UPDATE y DELETE **odbc_num_rows()** devolvera el numero de registros afectados. Para una clausula SELECT esta puede ser el numero de registros permitidos.

Nota: El uso de **odbc_num_rows()** para determinar el numero de registros permitidos despues de un SELECT devolvera -1.

odbc_result (PHP 3>= 3.0.6, PHP 4)

coge informacion de un campo

```
string odbc_result (int result_id, mixed field)
```

Devuelve el contenido de un campo.

field puede ser cualquier contenido del campo que queramos; o puede ser una cadena que contenga el nombre del campo; Por ejemplo:

```
$item_3 = odbc_result($Query_ID, 3 );
$item_val = odbc_result($Query_ID, "val");
```

La primera sentencia **odbc_result()** devuelve el valor del tercer campo dentro del registro actual de la cola resultante. La segunda función llama a **odbc_result()** y devuelve el valor de un campo cuyo nombre es "val" en el registro actual de la cola resultante. Ocurre un error si un número de columna para un campo es menor que uno o excede el número de campos en el registro actual. Similarmente, ocurre un error si un campo con un nombre que no sea uno de los nombres de campo de una tabla o tablas que sea o sean encoladas.

Los índices de campo comienzan en 1. Recordando el método binario de campos con gran información, es devuelto con referencia a **odbc_binmode()** y **odbc_longreadlen()**.

odbc_result_all (PHP 3>= 3.0.6, PHP 4)

Print result as HTML table

```
int odbc_result_all (int result_id [, string format])
```

En caso de error, como resultado, devuelve **false**.

odbc_result_all() Imprime todos los registros de un identificador producido por **odbc_exec()**. El resultado es impreso en una tabla formato HTML. Con el argumento de cadena opcional *format*, además, todas las formas de tablas pueden ser realizadas.

odbc_rollback (PHP 3>= 3.0.6, PHP 4)

Volver a pasar una transacción

```
int odbc_rollback (int connection_id)
```

Vuelve a pasar todas las declaraciones pendientes *connection_id*. Devuelve **true** cuando el resultado es satisfactorio, **false** cuando no lo es.

odbc_setoption (PHP 3>= 3.0.6, PHP 4)

Ajusta la configuración de ODBC. Devuelve **false** en caso de error, en otro caso **true**.

```
int odbc_setoption (int id, int function, int option, int param)
```

Esta función permite buscar las opciones ODBC para una conexión particular o consulta resultante. Esto está escrito para trabajar sobre problemas en peculiares drivers ODBC. Esta función solo se debería usar siendo un programador de ODBC y comprendiendo los efectos que las opciones tendrán. Debemos tener la certeza de que necesitamos una buena referencia de

reference to explicar todas las diferentes opciones y valores que pueden ser usados. Las diferentes versiones de drivers soportan diferentes opciones.

Ya que los efectos pueden variar dependiendo del driver ODBC, deberiamos usar la function en scripts para ser hecho publico lo que permitira que sea fuertemente desalentado. Algunas opciones ODBC no estan permitidas para esta funcion porque debe ser configurada antes de que la conexion sea establecida o la consulta este preparada. Sin embargo, si un determinado trabajo hace la tarea de PHP, el jefe no contaria con nosotros para usar un producto comercial, esto es lo que realmente suele pasar.

Id es una coexion id o resultado id sobre la que cambiaremos la configuracion. Para SQLSetConnectOption(), esta es una conexion id. Para SQLSetStmtOption(), este es un resultado id.

function es la funcion ODBC a usar. El valor deberia ser 1 para SQLSetConnectOption() y 2 para SQLSetStmtOption().

Parameter *option* es la opcion a configurar.

El parametro *param* es el valor para la escogida opcion *option*.

Ejemplo 1. Ejemplos ODBC Setoption

```
// 1. Option 102 of SQLSetConnectOption() is SQL_AUTOCOMMIT.
//     Value 1 of SQL_AUTOCOMMIT is SQL_AUTOCOMMIT_ON.
//     Este ejemplo tiene el mismo efecto que
//     odbc_autocommit($conn, true);

odbc_setoption ($conn, 1, 102, 1);

// 2. Option 0 of SQLSetStmtOption() is SQL_QUERY_TIMEOUT.
//     Este ejemplo asigna el tiempo de espera de la consulta a 30 segundos.

$result = odbc_prepare ($conn, $sql);
odbc_setoption ($result, 2, 0, 30);
odbc_execute ($result);
```

XLIX. Funciones de Oracle 8

Estas funciones permiten acceder a bases de datos Oracle8 y Oracle7. Estas usan la Oracle8 Call-Interface (OCI8). Necesitará las librerías clientes de Oracle8 para usar esta extensión.

Esta extensión es más flexible que las estándar de Oracle. Soporta el enlace de variables locales y globales de PHP con placeholders de Oracle, tiene soporte completo para LOB, FILE y ROWID y le permiten usar las variables definidas por el usuario.

OCIDefineByName (PHP 3>= 3.0.7, PHP 4)

Usa una variable de PHP para el define-step durante una sentencia SELECT

```
int OCIDefineByName (int stmt, string Column-Name, mixed &variable [, int type])
```

OCIDefineByName() busca el valor de las Columnas-SQL dentro de variables PHP definidas por el usuario. Cuidado que Oracle nombra todas las columnas en MAYUSCULAS, mientras que en su select puede usar también minúsculas write lower-case. **OCIDefineByName()** espera que *Column-Name* esté en mayúsculas. Si define una variable que no existe en la sentencia SELECT, no se producirá ningún error.

Si necesita definir un tipo de dato abstracto (LOB/ROWID/BFILE) tendrá que alojarlo primero usando la función **OCINewDescriptor()** function. Vea también la función **OCIBindByName()**.

Ejemplo 1. OCIDefineByName

```
<?php
/* OCIDefineByPos example thies@digicoll.de (980219) */

$conn = OCILogon("scott","tiger");

$stmt = OCIParse($conn,"select empno, ename from emp");

/* la definición DEBE hacerse ANTES del ociexecute! */

OCIDefineByName($stmt, "EMPNO",&$empno);
OCIDefineByName($stmt, "ENAME",&$ename);

OCIExecute($stmt);

while (OCIFetch($stmt)) {
    echo "empno:". $empno . "\n";
    echo "ename:". $ename . "\n";
}

OCIFreeStatement($stmt);
OCILogoff($conn);
?>
```

OCIBindByName (PHP 3>= 3.0.4, PHP 4)

Enlaza una variable PHP a un Placeholder de Oracle

```
int OCIBindByName (int stmt, string ph_name, mixed &variable, int length [, int type])
```

OCIBindByName() enlaza la variable PHP *variable* a un placeholder de Oracle *ph_name*. Si esta será usada para entrada o salida se determinará en tiempo de ejecución, y sera reservado el espacio necesario de almacenamiento. El parámetro *length* establece el tamaño máximo del enlace. Si establece *length* a -1 **OCIBindByName()** usará el tamaño de la *variable* para establecer el tamaño máximo.

Si necesita enlazar tipos de datos abstractos (LOB/ROWID/BFILE) necesitará primero reservar la memoria con la función **OCINewDescriptor()**. *length* no se usa para tipos de datos abstractos y debería establecerse a -1. La variable *type* le informa a Oracle, que tipo de descriptor queremos usar. Los valores posibles son: OCI_B_FILE (Binary-File), OCI_B_CFILE (Character-File), OCI_B_CLOB (Character-LOB), OCI_B_BLOB (Binary-LOB) and OCI_B_ROWID (ROWID).

Ejemplo 1. OCIDefineByName

```
<?php
/* OCIBindByName example thies@digicoll.de (980221)

    inserts 3 records into emp, and uses the ROWID for updating the
    records just after the insert.
*/
$conn = OCILogon("scott","tiger");

$stmt = OCIParse($conn,"insert into emp (empno, ename) ".
                  "values (:empno,:ename) ".
                  "returning ROWID into :rid");

$data = array(1111 => "Larry", 2222 => "Bill", 3333 => "Jim");

$rowid = OCINewDescriptor($conn,OCI_D_ROWID);

OCIBindByName($stmt,:empno,&$empno,32);
OCIBindByName($stmt,:ename,&$ename,32);
OCIBindByName($stmt,:rid,&$rowid,-1,OCI_B_ROWID);

$update = OCIParse($conn,"update emp set sal = :sal where ROWID = :rid");
OCIBindByName($update,:rid,&$rowid,-1,OCI_B_ROWID);
OCIBindByName($update,:sal,&$sal,32);

$sal = 10000;

while (list($empno,$ename) = each($data)) {
    OCIEexecute($stmt);
    OCIEexecute($update);
}
$rowid->free();

OCIFreeStatement($update);
OCIFreeStatement($stmt);

$stmt = OCIParse($conn,"select * from emp where empno in (1111,2222,3333)");
OCIEexecute($stmt);
while (OCIFetchInto($stmt,&$arr,OCI_ASSOC)) {
    var_dump($arr);
}
OCIFreeStatement($stmt);

/* delete our "junk" from the emp table.... */
$stmt = OCIParse($conn,"delete from emp where empno in (1111,2222,3333)");
OCIEexecute($stmt);
OCIFreeStatement($stmt);

OCILogoff($conn);
?>
```

OCILogon (PHP 3>= 3.0.4, PHP 4)

Establece la conexión con Oracle

```
int OCILogon (string username, string password [, string db])
```

OCILogon() devuelve el identificador de conexión necesario en la mayoría de las funciones OCI. El tercer parámetro, que es opcional, puede contener el nombre de la instancia a Oracle o el nombre dado en el fichero tnsnames.ora de la base de datos a la que nos queremos conectar. Si este parámetro no se especifica, PHP usa la variable de entorno ORACLE_SID (Oracle instance) o TWO_TASK (tnsnames.ora) para determinar la base de datos con la que queremos conectar.

Las conexiones son compartidas a nivel de página cuando usemos **OCILogon()**. Lo cual significa que los "commits" y "rollbacks" son aplicadas a todas las transacciones abiertas en la página, incluso si usted ha creado conexiones múltiples.

Este ejemplo demuestra como son compartidas las conexiones.

Ejemplo 1. OCILogon

```
<?php
print "<HTML><PRE>" ;
$db = "" ;

$c1 = ocilogon("scott","tiger",$db);
$c2 = ocilogon("scott","tiger",$db);

function create_table($conn)
{ $stmt = ociparse($conn,"create table scott.hallo (test varchar2(64))");
  ociexecute($stmt);
  echo $conn." created table\n\n";
}

function drop_table($conn)
{ $stmt = ociparse($conn,"drop table scott.hallo");
  ociexecute($stmt);
  echo $conn." dropped table\n\n";
}

function insert_data($conn)
{ $stmt = ociparse($conn,"insert into scott.hallo
    values('$conn' || ' ' || to_char(sysdate,'DD-MON-YY HH24:MI:SS'))");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn." inserted hallo\n\n";
}

function delete_data($conn)
{ $stmt = ociparse($conn,"delete from scott.hallo");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn." deleted hallo\n\n";
}

function commit($conn)
{ ocicommit($conn);
  echo $conn." committed\n\n";
}

function rollback($conn)
{ ocirollback($conn);
  echo $conn." rollback\n\n";
}

function select_data($conn)
{ $stmt = ociparse($conn,"select * from scott.hallo");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn."---selecting\n\n";
  while (ocifetch($stmt))
    echo $conn." <".ociresult($stmt,"TEST").">>\n\n";
  echo $conn."---done\n\n";
}
```

```

}

create_table($c1);
insert_data($c1);    // Insert a row using c1
insert_data($c2);    // Insert a row using c2

select_data($c1);    // Results of both inserts are returned
select_data($c2);

rollback($c1);        // Rollback using c1

select_data($c1);    // Both inserts have been rolled back
select_data($c2);

insert_data($c2);    // Insert a row using c2
commit($c2);         // commit using c2

select_data($c1);    // result of c2 insert is returned

delete_data($c1);    // delete all rows in table using c1
select_data($c1);    // no rows returned
select_data($c2);    // no rows returned
commit($c1);         // commit using c1

select_data($c1);    // no rows returned
select_data($c2);    // no rows returned

drop_table($c1);
print "</PRE></HTML>";
?>

```

See also **OCIPLogon()** and **OCINLogon()**.

OCIPLogon (PHP 3>= 3.0.8, PHP 4)

Conecta con una base de datos Oracle usando una conexión persistente. Devuelve una nueva sesión.

```
int OCIPLogon (string username, string password [, string db])
```

OCIPLogon() crea una conexión persistente con una base de datos Oracle 8. El tercer parámetro, que es opcional, puede contener el nombre de la instancia a Oracle o el nombre dado en el fichero tnsnames.ora de la base de datos a la que nos queremos conectar. Si este parámetro no se especifica, PHP usa la variable de entorno ORACLE_SID (Oracle instance) o TWO_TASK (tnsnames.ora) para determinar la base de datos con la que queremos conectar.

Vea también **OCILogon()** y **OCINLogon()**.

OCINLogon (PHP 3>= 3.0.8, PHP 4)

Conecta con una base de datos Oracle usando una nueva conexión. Devuelve una nueva sesión.

```
int OCINLogon (string username, string password [, string db])
```

OCINLogon() crea una nueva conexión con una base de datos Oracle 8. El tercer parámetro, que es opcional, puede contener el nombre de la instancia a Oracle o el nombre dado en el fichero tnsnames.ora de la base de datos a la que nos queremos conectar. Si este parámetro no se especifica, PHP usa la variable de entorno ORACLE_SID (Oracle instance) o TWO_TASK (tnsnames.ora) para determinar la base de datos con la que queremos conectar.

OCINLogon() fuerza una nueva conexión. Se debe usar si necesita aislar un conjunto de transacciones. Por defecto, las conexiones son compartidas a nivel de página si usa **OCILogon()** o a nivel del proceso del servidor web si usa **OCIPLogon()**. Si posee múltiples conexiones abiertas usando **OCINLogon()**, todos los "commits" y "rollbacks" se aplican sólo a la conexión especificada.

Este ejemplo demuestra como las conexiones están separadas.

Ejemplo 1. OCINLogon

```
<?php
print "<HTML><PRE>" ;
$db = " ";

$c1 = ocilogon("scott","tiger",$db);
$c2 = ocinlogon("scott","tiger",$db);

function create_table($conn)
{ $stmt = ociparse($conn,"create table scott.hallo (test
varchar2(64))");
ociexecute($stmt);
echo $conn." created table\n\n";
}

function drop_table($conn)
{ $stmt = ociparse($conn,"drop table scott.hallo");
ociexecute($stmt);
echo $conn." dropped table\n\n";
}

function insert_data($conn)
{ $stmt = ociparse($conn,"insert into scott.hallo
values('$conn' || ' ' || to_char(sysdate,'DD-MON-YY HH24:MI:SS'))");
ociexecute($stmt,OCI_DEFAULT);
echo $conn." inserted hallo\n\n";
}

function delete_data($conn)
{ $stmt = ociparse($conn,"delete from scott.hallo");
ociexecute($stmt,OCI_DEFAULT);
echo $conn." deleted hallo\n\n";
}

function commit($conn)
{ ocicommit($conn);
echo $conn." committed\n\n";
}

function rollback($conn)
{ ocirollback($conn);
echo $conn." rollback\n\n";
}

function select_data($conn)
{ $stmt = ociparse($conn,"select * from scott.hallo");
ociexecute($stmt,OCI_DEFAULT);
echo $conn."---selecting\n\n";
while (ocifetch($stmt))
echo $conn." <".ociresult($stmt,"TEST").">\n\n";
}
```

```

    echo $conn."---done\n\n";
}

create_table($c1);
insert_data($c1);

select_data($c1);
select_data($c2);

rollback($c1);

select_data($c1);
select_data($c2);

insert_data($c2);
commit($c2);

select_data($c1);

delete_data($c1);
select_data($c1);
select_data($c2);
commit($c1);

select_data($c1);
select_data($c2);

drop_table($c1);
print "</PRE></HTML>";
?>

```

See also **OCILogon()** and **OCIPLogon()**.

OCILogOff (PHP 3>= 3.0.4, PHP 4)

Termina la conexión con Oracle

```
int OCILogOff ( int connection )
```

OCILogOff() cierra una conexión con Oracle.

OCIExecute (PHP 3>= 3.0.4, PHP 4)

Ejecuta una sentencia

```
int OCIExecute ( int statement [, int mode] )
```

OCIExecute() ejecuta una sentencia previamente analizada. (see **OCIParse()**). El parámetro opcional *mode* le permite especificar el modo de ejecución (default is OCI_COMMIT_ON_SUCCESS). Si no desea que las sentencias se confirmen automáticamente, especifique OCI_DEFAULT como su modo.

OCICommit (PHP 3>= 3.0.7, PHP 4)

Confirma transacciones pendientes

```
int OCICommit (int connection)
```

OCICommit() confirma todas las sentencias pendientes para la conexión con Oracle *connection*.

OCIRollback (PHP 3>= 3.0.7, PHP 4)

Restablece todas las transacciones sin confirmar

```
int OCIRollback (int connection)
```

OCIRollback() restablece todas las transacciones sin confirmar para la conexión Oracle *connection*.

OCINewDescriptor (PHP 3>= 3.0.7, PHP 4)

Inicializa un nuevo descriptor vacío LOB/FILE (LOB por defecto)

```
string OCINewDescriptor (int connection [, int type])
```

OCINewDescriptor() Reserva espacio para mantener descriptores o localizadores LOB. Los valores válidos para el tipo *type* son OCI_D_FILE, OCI_D_LOB, OCI_D_ROWID. Para descriptores LOB, los métodos load, save, y savefile están asociados con el descriptor, para BFILE sólo el método load existe. Vea el segundo ejemplo.

Ejemplo 1. OCINewDescriptor

```
<?php
/* This script is designed to be called from a HTML form.
 * It expects $user, $password, $table, $where, and $commitsize
 * to be passed in from the form. The script then deletes
 * the selected rows using the ROWID and commits after each
 * set of $commitsize rows. (Use with care, there is no rollback)
 */
$conn = OCILogon($user, $password);
$stmt = OCIParse($conn, "select rowid from $table $where");
$rowid = OCINewDescriptor($conn, OCI_D_ROWID);
OCIDefineByName($stmt, "ROWID", &$rowid);
OCIEexecute($stmt);
while ( OCIFetch($stmt) ) {
    $nrows = OCIRowCount($stmt);
    $delete = OCIParse($conn, "delete from $table where ROWID = :rid");
    OCIBindByName($delete, ":rid", &$rowid, -1, OCI_B_ROWID);
    OCIEexecute($delete);
    print "$nrows\n";
    if ( ($nrows % $commitsize) == 0 ) {
        OCICommit($conn);
    }
}
$nrows = OCIRowCount($stmt);
print "$nrows deleted...\n";
OCIFreeStatement($stmt);
```

```

    OCILogoff($conn);
?>

<?php
/* This script demonstrates file upload to LOB columns
 * The formfield used for this example looks like this
 * <form action="upload.php3" method="post" enctype="multipart/form-data">
 *   <input type="file" name="lob_upload">
 *   ...
 */
if(!isset($lob_upload) || $lob_upload == 'none'){
?>
<form action="upload.php3" method="post" enctype="multipart/form-data">
Upload file: <input type="file" name="lob_upload"><br>
<input type="submit" value="Upload"> - <input type="reset">
</form>
<?php
} else {
    // $lob_upload contains the temporary filename of the uploaded file
    $conn = OCILogon($user, $password);
    $lob = OCINewDescriptor($conn, OCI_D_LOB);
    $stmt = OCIParse($conn, "insert into $table (id, the_blob)
        values(my_seq.NEXTVAL, EMPTY_BLOB()) returning the_blob into :the_blob");
    OCIBindByName($stmt, ':the_blob', &$lob, -1, OCI_B_BLOB);
    OCIEexecute($stmt);
    if($lob->savefile($lob_upload)){
        OCICCommit($conn);
        echo "Blob successfully uploaded\n";
    }else{
        echo "Couldn't upload Blob\n";
    }
    OCIFreeDescriptor($lob);
    OCIFreeStatement($stmt);
    OCILogoff($conn);
}
?>

```

OCIRowCount (PHP 3>= 3.0.7, PHP 4)

Obtiene el número de filas afectadas

```
int OCIRowCount (int statement)
```

OCIRowCount() devuelve el número de filas afectadas, por ej. en sentencias de actualización. !Esta función no indicará el número de filas que devuelve una sentencia SELECT!

Ejemplo 1. OCIRowCount

```
<?php
print "<HTML><PRE>";
$conn = OCILogon("scott", "tiger");
$stmt = OCIParse($conn, "create table emp2 as select * from emp");
OCIEexecute($stmt);
print OCIRowCount($stmt) . " rows inserted.<BR>";
OCIFreeStatement($stmt);
$stmt = OCIParse($conn, "delete from emp2");
OCIEexecute($stmt);
```

```

print OCIRowCount($stmt) . " rows deleted.<BR>";
OCICCommit($conn);
OCIFreeStatement($stmt);
$stmt = OCIParse($conn, "drop table emp2");
OCIEexecute($stmt);
OCIFreeStatement($stmt);
OCILogOff($conn);
print "</PRE></HTML>";
?>

```

OCINumCols (PHP 3>= 3.0.4, PHP 4)

Devuelve el número de columnas resultantes en una sentencia

```
int OCINumCols ( int stmt )
```

OCINumCols() devuelve el número de columnas en una sentencia

Ejemplo 1. OCINumCols

```

<?php
    print "<HTML><PRE>\n";
    $conn = OCILogon("scott", "tiger");
    $stmt = OCIParse($conn, "select * from emp");
    OCIEexecute($stmt);
    while ( OCIFetch($stmt) ) {
        print "\n";
        $ncols = OCINumCols($stmt);
        for ( $i = 1; $i <= $ncols; $i++ ) {
            $column_name = OCIColumnName($stmt,$i);
            $column_value = OCIResult($stmt,$i);
            print $column_name . ':' . $column_value . "\n";
        }
        print "\n";
    }
    OCIFreeStatement($stmt);
    OCILogoff($conn);
    print "</PRE>";
    print "</HTML>\n";
?>

```

OCIResult (PHP 3>= 3.0.4, PHP 4)

Devuelve el valor de una columna en la fila buscada

```
mixed OCIResult ( int statement, mixed column)
```

OCIResult() devuelve el valor de la columna *column* de la fila actual (vea **OCIFetch()**). **OCIResult()** devolverá todo como una cadena excepto para los tipo de datos abstractos (ROWIDs, LOBs and FILEs).

OCIFetch (PHP 3>= 3.0.4, PHP 4)

Busca la siguiente fila en el result-buffer

```
int OCIFetch (int statement)
```

OCIFetch() Busca la siguiente fila (para sentencias SELECT) dentro del result-buffer interno.

OCIFetchInto (PHP 3>= 3.0.4, PHP 4)

Busca la siguiente fila dentro del result-array

```
int OCIFetchInto (int stmt, array &result [, int mode])
```

OCIFetchInto() busca la siguiente fila (for SELECT statements) dentro del array *result*. **OCIFetchInto()** sobreescribirá el contenido previo de *result*. Por defecto *result* contendrá un array basado en todas las columnas que no son NULL.

El parámetro *mode* le permite cambiar el comportamiento por defecto. Puede especificar más de una flag simplemente añadiéndolas (ej. OCI_ASSOC+OCI_RETURN_NULLS). Las flags son:

- OCI_ASSOC Devuelve un array asociativo.
- OCI_NUM Devuelve un array numerado empezando en 1. (POR DEFECTO)
- OCI_RETURN_NULLS Devuelve columnas vacias.
- OCI_RETURN_LOBS Devuelve el valor de un LOB en vez de el descriptor.

OCIFetchStatement (PHP 3>= 3.0.8, PHP 4)

Busca todas la filas de un resultset dentro de un array.

```
int OCIFetchStatement (int stmt, array &variable)
```

OCIFetchStatement() busca todas las filas de un resultset dentro de un array definido por el usuario.
OCIFetchStatement() devuelve el numero de filas buscadas.

Ejemplo 1. OCIFetchStatement

```
<?php
/* OCIFetchStatement example mbritton@verinet.com (990624) */

$conn = OCILogon("scott","tiger");

$stmt = OCIParse($conn,"select * from emp");

OCIEexecute($stmt);

$nrows = OCIFetchStatement($stmt,$results);
if ( $nrows > 0 ) {
    print "<TABLE BORDER=\\"1\\">\n";
    print "<TR>\n";
    print "<TD>";
```

```

while ( list( $key, $val ) = each( $results ) ) {
    print "<TH>$key</TH>\n";
}
print "</TR>\n";

for ( $i = 0; $i < $nrows; $i++ ) {
    reset($results);
    print "<TR>\n";
    while ( $column = each($results) ) {
        $data = $column['value'];
        print "<TD>$data[$i]</TD>\n";
    }
    print "</TR>\n";
}
print "</TABLE>\n";
} else {
    echo "No data found<BR>\n";
}
print "$nrows Records Selected<BR>\n";

OCIFreeStatement($stmt);
OCILogoff($conn);

?>

```

OCIColumnIsNULL (PHP 3>= 3.0.4, PHP 4)

comprueba si una una columna es NULL

```
int OCIColumnIsNULL (int stmt, mixed column)
```

OCIColumnIsNULL() devuelve vedadero si la columna devuelta *column* en el resultset de la sentencia *stmt* es NULL. Puede usar el número de la columna (1-Based) o el nombre de la columna indicado por el parámetro *col*.

OCIColumnSize (PHP 3>= 3.0.4, PHP 4)

devuelve el tamaño de la columna

```
int OCIColumnSize (int stmt, mixed column)
```

OCIColumnSize() devuelve el tamaño de la columna indicada por Oracle Puede utilizar el número de la columna (1-Based) o el nombre indicado en el parámetro *col*.

Ejemplo 1. OCIColumnSize

```
<?php
    print "<HTML><PRE>\n";
    $conn = OCILogon("scott", "tiger");
    $stmt = OCIParse($conn, "select * from emp");
    OCIExecute($stmt);
    print "<TABLE BORDER=\\"1\\>";
    print "<TR>";
    print "<TH>Name</TH>";
    print "<TH>Type</TH>";
```

```

print "<TH>Length</TH>" ;
print "</TR>" ;
$ncols = OCINumCols($stmt) ;
for ( $i = 1; $i <= $ncols; $i++ ) {
    $column_name = OCIColumnName($stmt,$i);
    $column_type = OCIColumnType($stmt,$i);
    $column_size = OCIColumnSize($stmt,$i);
    print "<TR>" ;
    print "<TD>$column_name</TD>" ;
    print "<TD>$column_type</TD>" ;
    print "<TD>$column_size</TD>" ;
    print "</TR>" ;
}
print "</TABLE>" ;
OCIFreeStatement($stmt) ;
OCILogoff($conn) ;
print "</PRE>" ;
print "</HTML>\n" ;
?>

```

Vea también **OCINumCols()**, **OCIColumnName()**, y **OCIColumnSize()**.

OCIServerVersion (PHP 3>= 3.0.4, PHP 4)

Devuelve una cadena conteniendo información a cerca de la version del servidor.

```
string OCIServerVersion (int conn)
```

Ejemplo 1. OCIServerVersion

```
<?php
$conn = OCILogon("scott","tiger");
print "Server Version: " . OCIServerVersion($conn);
OCILogOff($conn);
?>
```

OCIStatementType (PHP 3>= 3.0.5, PHP 4)

Devuelve el tipo de una sentencia OCI.

```
string OCIStatementType (int stmt)
```

OCIStatementType() devuelve uno de los siguiente valores:

1. "SELECT"
2. "UPDATE"
3. "DELETE"
4. "INSERT"
5. "CREATE"

6. "DROP"
7. "ALTER"
8. "BEGIN"
9. "DECLARE"
10. "UNKNOWN"

Ejemplo 1. Code examples

```
<?php
    print "<HTML><PRE>";
    $conn = OCILogon("scott","tiger");
    $sql  = "delete from emp where deptno = 10";

    $stmt = OCIParse($conn,$sql);
    if ( OCIStatementType($stmt) == "DELETE" ) {
        die "You are not allowed to delete from this table<BR>";
    }

    OCILogoff($conn);
    print "</PRE></HTML>";
?>
```

OCINewCursor (PHP 3>= 3.0.8, PHP 4)

devuelve un cursor nuevo (Statement-Handle) - use esto para enlazar ref-cursors!

```
int OCINewCursor (int conn)
```

OCINewCursor() allocates a new statement handle on the specified connection.

Ejemplo 1. Usando un REF CURSOR de un procedimiento almacenado

```
<?php
// suppose your stored procedure info.output returns a ref cursor in :data

$conn = OCILogon("scott","tiger");
$curs = OCINewCursor($conn);
$stmt = OCIParse($conn,"begin info.output(:data); end;");

ocibindbyname($stmt,"data",&$curs,-1,OCI_B_CURSOR);
ociexecute($stmt);
ociexecute($curs);

while (OCIFetchInto($curs,&$data)) {
    var_dump($data);
}

OCIFreeCursor($stmt);
OCIFreeStatement($curs);
OCILogoff($conn);
?>
```

Ejemplo 2. Usando un REF CURSOR en una sentencia select

```
<?php
print "<HTML><BODY>" ;
$conn = OCILogon("scott","tiger");
$count_cursor = "CURSOR(select count(empno) num_emps from emp " .
                 "where emp.deptno = dept.deptno) as EMPCNT from dept";
$stmt = OCIParse($conn,"select deptno,dname,$count_cursor");

ociexec($stmt);
print "<TABLE BORDER=\\"1\\">" ;
print "<TR>" ;
print "<TH>DEPT NAME</TH>" ;
print "<TH>DEPT #</TH>" ;
print "<TH># EMPLOYEES</TH>" ;
print "</TR>" ;

while (OCIFetchInto($stmt,&$data,OCI_ASSOC)) {
    print "<TR>" ;
    $dname = $data[ "DNAME" ];
    $deptno = $data[ "DEPTNO" ];
    print "<TD>$dname</TD>" ;
    print "<TD>$deptno</TD>" ;
    ociexec($data[ "EMPCNT" ]);
    while (OCIFetchInto($data[ "EMPCNT" ],&$subdata,OCI_ASSOC)) {
        $num_emps = $subdata[ "NUM_EMPS" ];
        print "<TD>$num_emps</TD>" ;
    }
    print "</TR>" ;
}
print "</TABLE>" ;
print "</BODY></HTML>" ;
OCIFreeStatement($stmt);
OCILogoff($conn);
?>
```

OCIFreeStatement (PHP 3>= 3.0.5, PHP 4)

Libera todos los recursos asociados con una sentencia.

```
int OCIFreeStatement (int stmt)
```

OCIFreeStatement() devuelve cierto si la operacion se lleva a cabo, o falso en caso contrario.

OCIFreeCursor (PHP 3>= 3.0.8, PHP 4)

Libera todos los recursos asociados con cursor.

```
int OCIFreeCursor (int stmt)
```

OCIFreeCursor() devuelve cierto si la operacion se lleva a cabo, o falso en caso contrario.

OCIColumnName (PHP 3>= 3.0.4, PHP 4)

Devuelve el nombre de una columna.

```
string OCIColumnName ( int stmt, int col )
```

OCIColumnName() Devuelve el nombre de la columna correspondiente al número de la columna (1-based) que es pasado.

Ejemplo 1. OCIColumnName

```
<?php
    print "<HTML><PRE>\n";
    $conn = OCILogon("scott", "tiger");
    $stmt = OCIParse($conn,"select * from emp");
    OCIEexecute($stmt);
    print "<TABLE BORDER=\\"1\\">";
    print "<TR>";
    print "<TH>Name</TH>";
    print "<TH>Type</TH>";
    print "<TH>Length</TH>";
    print "</TR>";
    $ncols = OCINumCols($stmt);
    for ( $i = 1; $i <= $ncols; $i++ ) {
        $column_name = OCIColumnName($stmt,$i);
        $column_type = OCIColumnType($stmt,$i);
        $column_size = OCIColumnSize($stmt,$i);
        print "<TR>";
        print "<TD>$column_name</TD>";
        print "<TD>$column_type</TD>";
        print "<TD>$column_size</TD>";
        print "</TR>";
    }
    OCIFreeStatement($stmt);
    OCILogoff($conn);
    print "</PRE>";
    print "</HTML>\n";
?>
```

Vea también **OCINumCols()**, **OCIColumnType()**, y **OCIColumnSize()**.

OCIColumnType (PHP 3>= 3.0.4, PHP 4)

Devuelve el tipo de dato de una columna.

```
mixed OCIColumnType ( int stmt, int col )
```

OCIColumnType() devuelve el tipo de dato de una columna correspondiente al número de la columna (1-based) que es pasado.

Ejemplo 1. OCIColumnType

```
<?php
print "<HTML><PRE>\n";
$conn = OCILogon("scott", "tiger");
$stmt = OCIParse($conn, "select * from emp");
OCIEexecute($stmt);
print "<TABLE BORDER=\\"1\\">";
print "<TR>";
print "<TH>Name</TH>";
print "<TH>Type</TH>";
print "<TH>Length</TH>";
print "</TR>";
$ncols = OCINumCols($stmt);
for ( $i = 1; $i <= $ncols; $i++ ) {
    $column_name = OCIColumnName($stmt,$i);
    $column_type = OCIColumnType($stmt,$i);
    $column_size = OCIColumnSize($stmt,$i);
    print "<TR>";
    print "<TD>$column_name</TD>";
    print "<TD>$column_type</TD>";
    print "<TD>$column_size</TD>";
    print "</TR>";
}
OCIFreeStatement($stmt);
OCILogoff($conn);
print "</PRE>";
print "</HTML>\n";
?>
```

Vea también **OCINumCols()**, **OCIColumnName()**, y **OCIColumnSize()**.

OCIParse (PHP 3>= 3.0.4, PHP 4)

Analiza una consulta y devuelve una sentencia

```
int OCIParse (int conn, strint query)
```

OCIParse() analiza la *query* usando *conn*. Devuelve el identificador de la sentencia si la consulta es válida, y falso si no lo es. La *query* puede ser cualquier sentencia SQL válida.

OCIError (PHP 3>= 3.0.7, PHP 4)

Devuelve el último error de *stmt|conn|global*. Si no ocurre ningún error devuelve falso.

```
array OCIError ([int stmt|conn/global])
```

OCIError() devuelve el último error encontrado. Si el parámetro opcional *stmt|conn/global* no es usado, es devuelto el último error encontrado. Si no se encuentra ningún error, **OCIError()** devuelve falso. **OCIError()** devuelve el error como un array asociativo. En este array, *code* consiste en el código de error de Oracle y *message* en la cadena de descripción del error.

OCILInternalDebug (PHP 3>= 3.0.4, PHP 4)

Habilita o deshabilita la salida del depurador interno. Por defecto este está deshabilitado

```
void OCILInternalDebug (int onoff)
```

OCILInternalDebug() habilita la salida del depurador interno. Asigne 0 a *onoff* para deshabilitar la salida y 1 para habilitarla.

L. OpenSSL functions

This module uses the functions of OpenSSL (<http://www.openssl.org/>) for generation and verification of signatures and for sealing (encrypting) and opening (decrypting) data. You need to use OpenSSL >= 0.9.6 with this module.

OpenSSL offers many features that this module currently doesn't support. Some of these may be added in the future.

openssl_free_key (PHP 4 >= 4.0.4)

Free key resource

```
void openssl_free_key (int key_identifier)
```

openssl_free_key() frees the key associated with the specified *key_identifier* from memory.

openssl_get_privatekey (PHP 4 >= 4.0.4)

Prepare a PEM formatted private key for use

```
int openssl_get_privatekey (string key [, string passphrase])
```

Returns a positive key identifier on success, or false on error.

openssl_get_privatekey() parses the PEM formatted private key specified by *key* and prepares it for use by other functions. The optional parameter *passphrase* must be used if the specified key is encrypted (protected by a passphrase).

openssl_get_publickey (PHP 4 >= 4.0.4)

Extract public key from certificate and prepare it for use

```
int openssl_get_publickey (string certificate)
```

Returns a positive key identifier on success, or false on error.

openssl_get_publickey() extracts the public key from an X.509 certificate specified by *certificate* and prepares it for use by other functions.

openssl_open (PHP 4 >= 4.0.4)

Open sealed data

```
bool openssl_open (string sealed_data, string open_data, string env_key, int priv_key_id)
```

Returns true on success, or false on error. If successful the opened data is returned in *open_data*.

openssl_open() opens (decrypts) *sealed_data* using the private key associated with the key identifier *priv_key_id* and the envelope key *env_key*. The envelope key is generated when the data are sealed and can only be used by one specific private key. See **openssl_seal()** for more information.

Ejemplo 1. **openssl_open()** example

```
// $sealed and $env_key are assumed to contain the sealed data
// and our envelope key, both given to us by the sealer.

// fetch private key from file and ready it
$fp = fopen("/src/openssl-0.9.6/demos/sign/key.pem", "r");
```

```

$priv_key = fread($fp, 8192);
fclose($fp);
$pkeyid = openssl_get_privatekey($priv_key);

// decrypt the data and store it in $open
if (openssl_open($sealed, $open, $env_key, $pkeyid))
    echo "here is the opened data: ", $open;
else
    echo "failed to open data";

// free the private key from memory
openssl_free_key($pkeyid);

```

See also **[openssl_seal\(\)](#)**.

openssl_seal (PHP 4 >= 4.0.4)

Seal (encrypt) data

```
int openssl_seal (string data, string sealed_data, array env_keys, array pub_key_ids)
```

Returns the length of the sealed data on success, or false on error. If successful the sealed data is returned in *sealed_data*, and the envelope keys in *env_keys*.

openssl_seal() seals (encrypts) *data* by using RC4 with a randomly generated secret key. The key is encrypted with each of the public keys associated with the identifiers in *pub_key_ids* and each encrypted key is returned in *env_keys*. This means that one can send sealed data to multiple recipients (provided one has obtained their public keys). Each recipient must receive both the sealed data and the envelope key that was encrypted with the recipient's public key.

Ejemplo 1. **openssl_seal()** example

```

// $data is assumed to contain the data to be sealed

// fetch public keys for our recipients, and ready them
$fp = fopen("/src/openssl-0.9.6/demos/maurice/cert.pem", "r");
$cert = fread($fp, 8192);
fclose($fp);
$pk1 = openssl_get_publickey($cert);
// Repeat for second recipient
$fp = fopen("/src/openssl-0.9.6/demos/sign/cert.pem", "r");
$cert = fread($fp, 8192);
fclose($fp);
$pk2 = openssl_get_publickey($cert);

// seal message, only owners of $pk1 and $pk2 can decrypt $sealed with keys
// $ekeys[0] and $ekeys[1] respectively.
openssl_seal($data, $sealed, $ekeys, array($pk1,$pk2));

// free the keys from memory
openssl_free_key($pk1);
openssl_free_key($pk2);

```

See also **[openssl_open\(\)](#)**.

openssl_sign (PHP 4 >= 4.0.4)

Generate signature

```
bool openssl_sign (string data, string signature, int priv_key_id)
```

Returns true on success, or false on failure. If successful the signature is returned in *signature*.

openssl_sign() computes a signature for the specified *data* by using SHA1 for hashing followed by encryption using the private key associated with *priv_key_id*. Note that the data itself is not encrypted.

Ejemplo 1. openssl_sign() example

```
// $data is assumed to contain the data to be signed

// fetch private key from file and ready it
$fp = fopen("/src/openssl-0.9.6/demos/sign/key.pem", "r");
$priv_key = fread($fp, 8192);
fclose($fp);
$pkeyid = openssl_get_privatekey($priv_key);

// compute signature
openssl_sign($data, $signature, $pkeyid);

// free the key from memory
openssl_free_key($pkeyid);
```

See also **openssl_verify()**.

openssl_verify (PHP 4 >= 4.0.4)

Verify signature

```
int openssl_verify (string data, string signature, int pub_key_id)
```

Returns 1 if the signature is correct, 0 if it is incorrect, and -1 on error.

openssl_verify() verifies that the *signature* is correct for the specified *data* using the public key associated with *pub_key_id*. This must be the public key corresponding to the private key used for signing.

Ejemplo 1. openssl_verify() example

```
// $data and $signature are assumed to contain the data and the signature

// fetch public key from certificate and ready it
$fp = fopen("/src/openssl-0.9.6/demos/sign/cert.pem", "r");
$cert = fread($fp, 8192);
fclose($fp);
$pubkeyid = openssl_get_publickey($cert);

// state whether signature is okay or not
$ok = openssl_verify($data, $signature, $pubkeyid);
if ($ok == 1)
    echo "good";
```

```
elseif ($ok == 0)
    echo "bad";
else
    echo "ugly, error checking signature";

// free the key from memory
openssl_free_key($pubkeyid);
```

See also **openssl_sign()**.

LI. Funciones Oracle

Ora_Bind (PHP 3, PHP 4)

Vincula una variable PHP a un parámetro Oracle

```
int ora_bind (int cursor, string nombre de variable PHP, string nombre de parámetro SQL,
int longitud [, int tipo])
```

Devuelve verdadero si el vínculo se realiza con éxito, y sino devuelve falso. Los detalles de los errores pueden examinarse usando la funciones **ora_error()** y **ora_errorcode()**.

Esta función liga la variable PHP nombrada con el parámetro SQL. El parámetro SQL debe estar en la forma ":name". Con el parámetro optativo tipo, se define si el parámetro SQL se trata de un parámetro de entrada/salida (0 y por defecto), entrada (1) o salida (2). Como en PHP 3.0.1, se puede usar las constantes ORA_BIND_INOUT, ORA_BIND_IN y ORA_BIND_OUT en lugar de los números.

ora_bind debe ser llamada después de **ora_parse()** y antes de **ora_exec()**. Los valores de entrada pueden pasarse por asignación a las variables PHP vinculadas, después de la llamada a **ora_exec()** dichas variables contendrán los valores de salida, si éstos estuvieran disponibles.

```
<?php
ora_parse($curs, "declare tmp INTEGER; begin tmp := :in; :out := tmp; :x := 7.77; end;");
ora_bind($curs, "result", ":x", $len, 2);
ora_bind($curs, "input", ":in", 5, 1);
ora_bind($curs, "output", ":out", 5, 2);
$input = 765;
ora_exec($curs);
echo "Result: $result<BR>Out: $output<BR>In: $input";
?>
```

Ora_Close (PHP 3, PHP 4)

Cierra un cursor Oracle

```
int ora_close (int cursor)
```

Devuelve verdadero si el cierre fué exitoso, o falso de lo contrario. Los detalles de los errores se recuperan usando las funciones **ora_error()** y **ora_errorcode()**.

Esta función cierra un cursor de datos abierto con **ora_open()**.

Ora_ColumnName (PHP 3, PHP 4)

toma el nombre de una columna de resultados Oracle

```
string Ora_ColumnName (int cursor, int column)
```

Devuelve el nombre de un campo/columna *column* en el cursor *cursor*. el nombre devuelto estará en letras mayúsculas.

Ora_ColumnType (PHP 3, PHP 4)

toma el tipo de una columna de resultados Oracle

```
string Ora_ColumnType (int cursor, int column)
```

Devuelve el nombre del tipo de datos del campo o columna *column* en el cursor *cursor*. Se devolverá un tipo de datos, de entre los siguientes:

```
"VARCHAR2"
"VARCHAR"
"CHAR"
"NUMBER"
"LONG"
"LONG RAW"
"ROWID"
"DATE"
"CURSOR"
```

Ora_Commit (PHP 3, PHP 4)

realiza una transacción Oracle

```
int ora_commit (int conn)
```

Devuelve verdadero si es exitosa, de lo contrario devuelve falso. Puede verse los detalles del error usando las funciones **ora_error()** y **ora_errorcode()**. Esta función realiza una transacción Oracle. Se define como transacción cualquier cambio en una conexión dada, desde la última tarea/retroceso en la ejecución (rollback), anulación de la ejecución automática de tareas (autocommit), o cuando se ha establecido la conexión.

Ora_CommitOff (PHP 3, PHP 4)

deshabilita el modo automático de ejecución de tareas (autocommit)

```
int ora_commitoff (int conn)
```

Devuelve verdadero si se ejecuta con éxito, sino devuelve falso. Los pormenores del error en cuestión, pueden revisarse invocando las funciones **ora_error()** y **ora_errorcode()**.

Esta función deshabilita la ejecución automática luego de cada instancia **ora_exec()**.

Ora_CommitOn (PHP 3, PHP 4)

Habilita la ejecución automática de tareas (autocommit)

```
int ora_commiton (int conn)
```

Esta función habilita la ejecución automática luego de cada instancia **ora_exec()** en la conexión dada.

Devuelve verdadero si se ejecuta con éxito, sino devuelve falso. Los pormenores del error en cuestión, pueden revisarse invocando las funciones **ora_error()** y **ora_errorcode()**.

Ora_Error (PHP 3, PHP 4)

toma los mensajes de error de Oracle

```
string Ora_Error (int cursor_or_connection)
```

Devuelve los mensajes de error en la forma XXX-NNNNN donde XXX es la procedencia del error y NNNNN es la identificación del mensaje de error.

Nota: El soporte para las identificaciones de conexión fue agregado en la versión 3.0.4.

En las versiones UNIX de Oracle, pueden encontrarse detalles acerca de un mensaje de error como este: \$ **oerr ora 00001 00001**, 00000, "unique constraint (%s.%s) violated"// *Cause: An update or insert statement attempted to insert a duplicate key // For Trusted ORACLE configured in DBMS MAC mode, you may see // this message if a duplicate entry exists at a different level. // *Action: Either remove the unique restriction or do not insert the key

Ora_ErrorCode (PHP 3, PHP 4)

captura el código de error Oracle

```
int Ora_ErrorCode (int cursor_or_connection)
```

Devuelve el código numérico de error de la última declaración ejecutada en el cursor o conexión especificada.

* *FIXME: se listarán los valores posibles?*

Nota: El soporte para las identificaciones de conexión fue agregado en la versión 3.0.4.

Ora_Exec (PHP 3, PHP 4)

ejecuta las declaraciones interpretadas en un cursor Oracle

```
int ora_exec (int cursor)
```

Devuelve verdadero ante la ejecución exitosa, de lo contrario, devuelve falso. Los detalles del error pueden verse invocando las funciones **ora_error()** y **ora_errorcode()**.

Ora_Fetch (PHP 3, PHP 4)

extrae una fila de datos a partir de un cursor

```
int ora_fetch (int cursor)
```

Devuelve verdadero (se extrajo una fila) o falso (no hay mas filas, o ha ocurrido un error). Si ocurre un error, los detalles del mismo pueden verse invocando las funciones **ora_error()** y **ora_errorcode()**. Si no hubo errores, **ora_errorcode()** devolverá 0. Recupera una hilera de datos partiendo de un cursor especificado.

Ora_GetColumn (PHP 3, PHP 4)

toma datos de la fila extraída

```
mixed ora_getcolumn (int cursor, mixed column)
```

Devuelve la columna de datos. Si hay un error, se devuelve falso y **ora_errorcode()** devulve un valor distinto de cero. Note, de igual manera, que la busqueda de un resultado Falso en esta función, puede resultar verdadera, aún en casos en que no ocurran errores:(resultado NULO, cadenas vacias, valor 0 o cadenas "0"). Extrae los datos para una columna o resultado de función.

Ora_Logoff (PHP 3, PHP 4)

cierra una conexión Oracle

```
int ora_logoff (int connection)
```

Devuelve verdadero si es exitosa, o falso si hay errores. Los detalles del error pueden verse invocando las funciones **ora_error()** y **ora_errorcode()**. Cierra la sesión de trabajo del usuario, y lo desconecta del servidor.

Ora_Logon (PHP 3, PHP 4)

Abre una conexión Oracle

```
int ora_logon (string usuario, string contraseña)
```

Establece una conexión entre PHP y una base de datos Oracle, con los datos de nombre de usuario y contraseña especificados.

Las conexiones pueden llevarse adelante usando SQL*Net indicando el nombre TNS al *usuario* de este modo:

```
$conn = Ora_Logon( "usuario@TNSNAME" , "contraseña" );
```

Si hubiesen datos con caracteres no-ASCII, habría que asegurarse de que esté presente la variable de entorno NLS_LANG en el sistema. Para los modulos de servidor, deberían definirse en el entorno del servidor antes de iniciarlos.

Devuelve el índice de la conexión si aquella tuvo éxito, de lo contrario devuelve falso. Los detalles del error pueden verse invocando las funciones **ora_error()** y **ora_errorcode()**.

Ora_Open (PHP 3, PHP 4)

abre un cursor Oracle

```
int ora_open (int connection)
```

Abre un cursor asociado con la conexión.

Devuelve el índice del cursor o Falso si hay un error. Los detalles del error pueden verse invocando las funciones **ora_error()** y **ora_errorcode()**.

Ora_Parse (PHP 3, PHP 4)

interpreta una declaración SQL

```
int ora_parse (int cursor_ind, string sql_statement, int defer)
```

Esta función interpreta una declaración SQL o un bloque PL/SQL y los asocia con el cursor dado. Devuelve 0 si se ejecuta con éxito, o -1 ante un error.

Ora_Rollback (PHP 3, PHP 4)

retrocede en la lista de transacciones (hace un roll back)

```
int ora_rollback (int connection)
```

Deshace una transacción Oracle. (Ver **ora_commit()** para la definición de transacción.)

Devuelve verdadero si tiene éxito, o falso si hay un error. Los detalles del error pueden verse invocando las funciones **ora_error()** y **ora_errorcode()**.

LII. Ovrimos SQL functions

Ovrimos SQL Server, is a client/server, transactional RDBMS combined with Web capabilities and fast transactions.

Ovrimos SQL Server is available at www.ovrimos.com (<http://www.ovrimos.com/>). To enable ovrimos support in PHP just compile php with the '--with-ovrimos' parameter to configure script. You'll need to install the sqlcli library available in the Ovrimos SQL Server distribution.

Ejemplo 1. Connect to Ovrimos SQL Server and select from a system table

```
<?php
$conn = ovrimos_connect ("server.domain.com", "8001", "admin", "password");
if ($conn != 0) {
    echo ("Connection ok!");
    $res = ovrimos_exec ($conn, "select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Statement ok!";
        ovrimos_result_all ($res);
        ovrimos_free_result ($res);
    }
    ovrimos_close($conn);
}
?>
```

This will just connect to SQL Server.

ovrimos_connect (PHP 4 >= 4.0.3)

Connect to the specified database

```
int ovrimos_connect (string host, string db, string user, string password)
```

ovrimos_connect() is used to connect to the specified database.

ovrimos_connect() returns a connection id (greater than 0) or 0 for failure. The meaning of 'host' and 'port' are those used everywhere in Ovrinos APIs. 'Host' is a host name or IP address and 'db' is either a database name, or a string containing the port number.

Ejemplo 1. ovrimos_connect() Example

```
<?php
$conn = ovrimos_connect ("server.domain.com", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res=ovrimos_exec ($conn, "select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Statement ok!";
        ovrimos_result_all ($res);
        ovrimos_free_result ($res);
    }
    ovrimos_close ($conn);
}
?>
```

The above example will connect to the database and print out the specified table.

ovrimos_close (PHP 4 >= 4.0.3)

Closes the connection to ovrimos

```
void ovrimos_close (int connection)
```

ovrimos_close() is used to close the specified connection.

ovrimos_close() closes a connection to Ovrinos. This has the effect of rolling back uncommitted transactions.

ovrimos_close_all (PHP 4 >= 4.0.3)

Closes all the connections to ovrimos

```
void ovrimos_close_all (void)
```

ovrimos_close_all() is used to close all the connections.

ovrimos_close_all() closes all connections to Ovrinos. This has the effect of rolling back uncommitted transactions.

ovrimos_longreadlen (PHP 4 >= 4.0.3)

Specifies how many bytes are to be retrieved from long datatypes

```
int ovrimos_longreadlen (int result_id, int length)
```

ovrimos_longreadlen() is used to specify how many bytes are to be retrieved from long datatypes.

ovrimos_longreadlen() specifies how many bytes are to be retrieved from long datatypes (long varchar and long varbinary). Default is zero. Regardless of its taking a result_id as an argument, it currently sets this parameter for all result sets. Returns true.

ovrimos_prepare (PHP 4 >= 4.0.3)

Prepares an SQL statement

```
int ovrimos_prepare (int connection_id, string query)
```

ovrimos_prepare() is used to prepare an SQL statement.

ovrimos_prepare() prepares an SQL statement and returns a result_id (or false on failure).

Ejemplo 1. Connect to Ovrimos SQL Server and prepare a statement

```
<?php
$conn=ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn!=0) {
    echo "Connection ok!";
    $res=ovrimos_prepare ($conn, "select table_id, table_name
                                from sys.tables where table_id=1");
    if ($res != 0) {
        echo "Prepare ok!";
        if (ovrimos_execute ($res)) {
            echo "Execute ok!\n";
            ovrimos_result_all ($res);
        } else {
            echo "Execute not ok!";
        }
        ovrimos_free_result ($res);
    } else {
        echo "Prepare not ok!\n";
    }
    ovrimos_close ($conn);
}
?>
```

This will connect to Ovrimos SQL Server, prepare a statement and the execute it.

ovrimos_execute (PHP 4 >= 4.0.3)

Executes a prepared SQL statement

```
int ovrimos_execute (int result_id [, array parameters_array])
```

ovrimos_execute() is used to execute an SQL statement.

ovrimos_execute() executes a prepared statement. Returns true or false. If the prepared statement contained parameters (question marks in the statement), the correct number of parameters should be passed in an array. Notice that I don't follow the PHP convention of placing just the name of the optional parameter inside square brackets. I couldn't bring myself on liking it.

ovrimos_cursor (PHP 4 >= 4.0.3)

Returns the name of the cursor

```
int ovrimos_cursor (int result_id)
```

ovrimos_cursor() is used to get the name of the cursor.

ovrimos_cursor() returns the name of the cursor. Useful when wishing to perform positioned updates or deletes.

ovrimos_exec (PHP 4 >= 4.0.3)

Executes an SQL statement

```
int ovrimos_exec (int connection_id, string query)
```

ovrimos_exec() is used to execute an SQL statement.

ovrimos_exec() executes an SQL statement (query or update) and returns a result_id or false. Evidently, the SQL statement should not contain parameters.

ovrimos_fetch_into (PHP 4 >= 4.0.3)

Fetches a row from the result set

```
int ovrimos_fetch_into (int result_id, array result_array [, string how [, int rownumber]])
```

ovrimos_fetch_into() is used to fetch a row from the result set.

ovrimos_fetch_into() fetches a row from the result set into 'result_array', which should be passed by reference. Which row is fetched is determined by the two last parameters. 'how' is one of 'Next' (default), 'Prev', 'First', 'Last', 'Absolute', corresponding to forward direction from current position, backward direction from current position, forward direction from the start, backward direction from the end and absolute position from the start (essentially equivalent to 'first' but needs 'rownumber'). Case is not significant. 'Rownumber' is optional except for absolute positioning. Returns true or false.

Ejemplo 1. A fetch into example

```
<?php
$conn=ovrimos_connect ("neptune", "8001", "admin", "password");
if ($conn!=0) {
    echo "Connection ok!";
    $res=ovrimos_exec ($conn,"select table_id, table_name from sys.tables");
```

```

if ($res != 0) {
    echo "Statement ok!";
    if (ovrimos_fetch_into ($res, &$row)) {
        list ($table_id, $table_name) = $row;
        echo "table_id=".$table_id.", table_name=".$table_name."\n";
        if (ovrimos_fetch_into ($res, &$row)) {
            list ($table_id, $table_name) = $row;
            echo "table_id=".$table_id.", table_name=".$table_name."\n";
        } else {
            echo "Next: error\n";
        }
    } else {
        echo "First: error\n";
    }
    ovrimos_free_result ($res);
}
ovrimos_close ($conn);
}
?>

```

This example will fetch a row.

ovrimos_fetch_row (PHP 4 >= 4.0.3)

Fetches a row from the result set

```
int ovrimos_fetch_row (int result_id [, int how [, int row_number]])
```

ovrimos_fetch_row() is used to fetch a row from the result set.

ovrimos_fetch_row() fetches a row from the result set. Column values should be retrieved with other calls. Returns true or false.

Ejemplo 1. A fetch row example

```
<?php
$conn = ovrimos_connect ("remote.host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res=ovrimos_exec ($conn, "select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Statement ok!";
        if (ovrimos_fetch_row ($res, "First")) {
            $table_id = ovrimos_result ($res, 1);
            $table_name = ovrimos_result ($res, 2);
            echo "table_id=".$table_id.", table_name=".$table_name."\n";
            if (ovrimos_fetch_row ($res, "Next")) {
                $table_id = ovrimos_result ($res, "table_id");
                $table_name = ovrimos_result ($res, "table_name");
                echo "table_id=".$table_id.", table_name=".$table_name."\n";
            } else {
                echo "Next: error\n";
            }
        } else {
            echo "First: error\n";
        }
        ovrimos_free_result ($res);
    }
}
```

```

        ovrimos_close ($conn);
}
?>

```

This will fetch a row and print the result.

ovrimos_result (PHP 4 >= 4.0.3)

Retrieves the output column

```
int ovrimos_result (int result_id, mixed field)
```

ovrimos_result() is used to retrieve the output column.

ovrimos_result() retrieves the output column specified by 'field', either as a string or as an 1-based index.

ovrimos_result_all (PHP 4 >= 4.0.3)

Prints the whole result set as an HTML table

```
int ovrimos_result_all (int result_id [, string format])
```

ovrimos_result_all() is used to print an HTML table containing the whole result set.

ovrimos_result_all() prints the whole result set as an HTML table. Returns true or false.

Ejemplo 1. Prepare a statement, execute, and view the result

```

<?php
$conn = ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res = ovrimos_prepare ($conn, "select table_id, table_name
                                from sys.tables where table_id = 7");
    if ($res != 0) {
        echo "Prepare ok!";
        if (ovrimos_execute ($res, array(3))) {
            echo "Execute ok!\n";
            ovrimos_result_all ($res);
        } else {
            echo "Execute not ok!";
        }
        ovrimos_free_result ($res);
    } else {
        echo "Prepare not ok!\n";
    }
    ovrimos_close ($conn);
}
?>

```

This will execute an SQL statement and print the result in an HTML table.

Ejemplo 2. Ovrimos_result_all with meta-information

```
<?php
$conn = ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res = ovrimos_exec ($conn, "select table_id, table_name
                                from sys.tables where table_id = 1")
if ($res != 0) {
    echo "Statement ok! cursor=".ovrimos_cursor ($res)."\n";
    $colnb = ovrimos_num_fields ($res);
    echo "Output columns=".$colnb."\n";
    for ($i=1; $i<=$colnb; $i++) {
        $name = ovrimos_field_name ($res, $i);
        $type = ovrimos_field_type ($res, $i);
        $len = ovrimos_field_len ($res, $i);
        echo "Column ".$i." name=".$name." type=".$type." len=".$len."\n";
    }
    ovrimos_result_all ($res);
    ovrimos_free_result ($res);
}
ovrimos_close ($conn);
}
?>
```

Ejemplo 3. ovrimos_result_all example

```
<?php
$conn = ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res = ovrimos_exec ($conn, "update test set i=5");
    if ($res != 0) {
        echo "Statement ok!";
        echo ovrimos_num_rows ($res)." rows affected\n";
        ovrimos_free_result ($res);
    }
    ovrimos_close ($conn);
}
?>
```

ovrimos_num_rows (PHP 4 >= 4.0.3)

Returns the number of rows affected by update operations

```
int ovrimos_num_rows (int result_id)
```

ovrimos_num_rows() is used to get the number of rows affected by update operations.

ovrimos_num_rows() returns the number of rows affected by update operations.

ovrimos_num_fields (PHP 4 >= 4.0.3)

Returns the number of columns

```
int ovrimos_num_fields (int result_id)
```

ovrimos_num_fields() is used to get the number of columns.

ovrimos_num_fields() returns the number of columns in a result_id resulting from a query.

ovrimos_field_name (PHP 4 >= 4.0.3)

Returns the output column name

```
int ovrimos_field_name (int result_id, int field_number)
```

ovrimos_field_name() is used to get the output column name.

ovrimos_field_name() returns the output column name at the (1-based) index specified.

ovrimos_field_type (PHP 4 >= 4.0.3)

Returns the (numeric) type of the output column

```
int ovrimos_field_type (int result_id, int field_number)
```

ovrimos_field_type() is used to get the (numeric) type of the output column.

ovrimos_field_type() returns the (numeric) type of the output column at the (1-based) index specified.

ovrimos_field_len (PHP 4 >= 4.0.3)

Returns the length of the output column

```
int ovrimos_field_len (int result_id, int field_number)
```

ovrimos_field_len() is used to get the length of the output column.

ovrimos_field_len() returns the length of the output column at the (1-based) index specified.

ovrimos_field_num (PHP 4 >= 4.0.3)

Returns the (1-based) index of the output column

```
int ovrimos_field_num (int result_id, string field_name)
```

ovrimos_field_num() is used to get the (1-based) index of the output column.

ovrimos_field_num() returns the (1-based) index of the output column specified by name, or false.

ovrimos_free_result (PHP 4 >= 4.0.3)

Frees the specified result_id

```
int ovrimos_free_result (int result_id)
```

ovrimos_free_result() is used to free the result_id.

ovrimos_free_result() frees the specified result_id. Returns true.

ovrimos_commit (PHP 4 >= 4.0.3)

Commits the transaction

```
int ovrimos_commit (int connection_id)
```

ovrimos_commit() is used to commit the transaction.

ovrimos_commit() commits the transaction.

ovrimos_rollback (PHP 4 >= 4.0.3)

Rolls back the transaction

```
int ovrimos_rollback (int connection_id)
```

ovrimos_rollback() is used to roll back the transaction.

ovrimos_rollback() rolls back the transaction.

LIII. Output Control Functions

The Output Control functions allow you to control when output is sent from the script. This can be useful in several different situations, especially if you need to send headers to the browser after your script has began outputting data. The Output Control functions do not affect headers sent using **header()** or **setcookie()**, only functions such as **echo()** and data between blocks of PHP code.

Ejemplo 1. Output Control example

```
<?php  
  
ob_start();  
echo "Hello\n";  
  
setcookie ("cookiename", "cookiedata");  
  
ob_end_flush();  
  
?>
```

In the above example, the output from **echo()** would be stored in the output buffer until **ob_end_flush()** was called. In the mean time, the call to **setcookie()** successfully stored a cookie without causing an error. (You can not normally send headers to the browser after data has already been sent.)

See also **header()** and **setcookie()**.

flush (PHP 3, PHP 4)

Flush the output buffer

```
void flush(void);
```

Flushes the output buffers of PHP and whatever backend PHP is using (CGI, a web server, etc.) This effectively tries to push all the output so far to the user's browser.

ob_start (PHP 4)

Turn on output buffering

```
void ob_start(void);
```

This function will turn output buffering on. While output buffering is active no output is sent from the script, instead the output is stored in an internal buffer.

The contents of this internal buffer may be copied into a string variable using **ob_get_contents()**. To output what is stored in the internal buffer, use **ob_end_flush()**. Alternatively, **ob_end_clean()** will silently discard the buffer contents.

See also **ob_get_contents()**, **ob_end_flush()**, **ob_end_clean()**, and **ob_implicit_flush()**

ob_get_contents (PHP 4)

Return the contents of the output buffer

```
string ob_get_contents(void);
```

This will return the contents of the output buffer or FALSE, if output buffering isn't active.

See also **ob_start()** and **ob_get_length()**.

ob_get_length (PHP 4 >= 4.0.2)

Return the length of the output buffer

```
string ob_get_length(void);
```

This will return the length of the contents in the output buffer or FALSE, if output buffering isn't active.

See also **ob_start()** and **ob_get_contents()**.

ob_end_flush (PHP 4)

Flush (send) the output buffer and turn off output buffering

```
void ob_end_flush(void);
```

This function will send the contents of the output buffer (if any) and turn output buffering off. If you want to further process the buffer's contents you have to call **ob_get_contents()** before **ob_end_flush()** as the buffer contents are discarded after **ob_get_contents()** is called.

See also **ob_start()**, **ob_get_contents()**, and **ob_end_clean()**.

ob_end_clean (PHP 4)

Clean (erase) the output buffer and turn off output buffering

```
void ob_end_clean(void);
```

This function discards the contents of the output buffer and turns off output buffering.

See also **ob_start()** and **ob_end_flush()**.

ob_implicit_flush (PHP 4 >= 4.0b4)

Turn implicit flush on/off

```
void ob_implicit_flush ([int flag])
```

ob_implicit_flush() will turn implicit flushing on or off (if no *flag* is given, it defaults to on). Implicit flushing will result in a flush operation after every output call, so that explicit calls to **flush()** will no longer be needed.

Turning implicit flushing on will disable output buffering, the output buffers current output will be sent as if **ob_end_flush()** had been called.

See also **flush()**, **ob_start()**, and **ob_end_flush()**.

LIV. PDF functions

You can use the PDF functions in PHP to create PDF files if you have the PDF library by Thomas Merz (available at <http://www.pdflib.com/pdflib/index.html>; you will also need the JPEG library (<ftp://ftp.uu.net/graphics/jpeg/>) and the TIFF library (<http://www.libtiff.org/>) to compile this. These two libs also quite often make problems when configuring php. Follow the messages of configure to fix possible problems. If you use pdflib 2.01 check how the lib was installed. There should be file or link libpdf.so. Version 2.01 just creates a lib with the name libpdf2.01.so which cannot be found when linking the test programm in configure. You will have to create a symbolic link from libpdf.so to libpdf2.01.so.).

Version 2.20 of pdflib has introduced more changes to its API and support for chinese and japanese fonts. This unfortunately causes some changes of the pdf module of php4 (not php3). If you use pdflib 2.20 handle the in memory generation of PDF documents with care. Until pdflib 3.0 is released it might be unstable. The encoding parameter of **pdf_set_font()** has changed to a string. This means that instead of e.g. 4 you have to use 'winansi'.

If you use pdflib 2.30 the **pdf_set_text_matrix()** will have gone. It is not supported any more. In general it is a good advise to consult the release notes of the used version of pdflib for possible changes.

Since version 3.0 of pdflib you should configure pdflib with the option `-enable-shared-pdflib`.

Any version of PHP4 after March, 9th 2000 do not support versions of pdflib older than 3.0. PHP3 on the other hand should not be used with version newer than 2.01.

Please consult the excellent documentation for pdflib shipped with the source distribution of pdflib. It provides a very good overview of what pdflib capable of doing. Most of the functions in pdflib and the PHP module have the same name. The parameters are also identical. You should also understand some of the concepts of PDF or Postscript to efficiently use this module. All lengths and coordinates are measured in Postscript points. There are generally 72 PostScript points to an inch, but this depends on the output resolution.

There is another PHP module for pdf document creation based on FastIO's (<http://www.fastio.com/>). ClibPDF. It has a slightly different API. Check the [ClibPDF functions](#) section for details.

Currently all versions of pdflib are supported. It is recommended that you use the newest version since it has more features and fixes some problems which required a patch for the old version. Unfortunately, the changes of the pdflib API in 2.x compared to 0.6 have been so severe that even some PHP functions had to be altered. Here is a list of changes:

- The Info structure does not exist anymore. Therefore the function **pdf_get_info()** is obsolete and the functions **pdf_set_info_creator()**, **pdf_set_info_title()**, **pdf_set_info_author()**, **pdf_set_info_subject()** and **pdf_set_info_keywords()** do not take the info structure as the first parameter but the pdf document. This also means that the pdf document must be opened before these functions can be called. The above functions can and should also be replaced by **pdf_set_info()**
- The way a new document is opened has changed. The function **pdf_open()** takes only one parameter which is the file handle of a file opened with **fopen()**.

There were some more changes with the release 2.01 of pdflib which should be covered by PHP. Some functions are not required anymore (e.g. **pdf_put_image()**). You will get a warning so don't be shocked.

The pdf module introduces two new types of variables (if pdflib 2.x is used it is only one new type). They are called **pdflibdoc** and **pdflibinfo** (**pdflibinfo** is not existent if pdflib 2.x is used. **pdflibdoc** is a pointer to a pdf document and almost all functions need it as its first parameter. **pdflibinfo** contains meta data about the PDF document. It has to be set before **pdf_open()** is called.

Nota: The following is only true for pdflib 0.6. Read the pdflib manual for newer version

In order to output text into a PDF document you will need to provide the afm file for each font. Afm files contain font metrics for a Postscript font. By default these afm files are searched for in a directory named 'fonts' relative to the directory where the PHP script is located. (Again, this was true for pdflib 0.6, newer versions do not necessarily need the afm files.)

Most of the functions are fairly easy to use. The most difficult part is probably to create a very simple pdf document at all. The following example should help to get started. It uses the PHP functions for pdflib 0.6. It creates the file test.pdf with one page. The page contains the text "Times-Roman" in an outlined 30pt font. The text is also underlined.

Ejemplo 1. Creating a PDF document with pdflib 0.6

```
<?php
$fp = fopen("test.pdf", "w");
$info = PDF_get_info();
pdf_set_info_author($info, "Uwe Steinmann");
PDF_set_info_title($info, "Test for PHP wrapper of PDFlib 0.6");
PDF_set_info_author($info, "Name of Author");
pdf_set_info_creator($info, "See Author");
pdf_set_info_subject($info, "Testing");
$pdf = PDF_open($fp, $info);
PDF_begin_page($pdf, 595, 842);
PDF_add_outline($pdf, "Page 1");
pdf_set_font($pdf, "Times-Roman", 30, 4);
pdf_set_text_rendering($pdf, 1);
PDF_show_xy($pdf, "Times Roman outlined", 50, 750);
pdf_moveto($pdf, 50, 740);
pdf_lineto($pdf, 330, 740);
pdf_stroke($pdf);
PDF_end_page($pdf);
PDF_close($pdf);
fclose($fp);
echo "<A HREF=getpdf.php3>finished</A>";
?>
```

The PHP script getpdf.php3 just outputs the pdf document.

```
<?php
$fp = fopen("test.pdf", "r");
header("Content-type: application/pdf");
fpassthru($fp);
fclose($fp);
?>
```

Doing the same with pdflib 2.x looks like the following:

Ejemplo 2. Creating a PDF document with pdflib 2.x

```
<?php
$fp = fopen("test.pdf", "w");
$pdf = PDF_open($fp);
pdf_set_info_author($pdf, "Uwe Steinmann");
PDF_set_info_title($pdf, "Test for PHP wrapper of PDFlib 2.0");
PDF_set_info_author($pdf, "Name of Author");
pdf_set_info_creator($pdf, "See Author");
pdf_set_info_subject($pdf, "Testing");
PDF_begin_page($pdf, 595, 842);
PDF_add_outline($pdf, "Page 1");
pdf_set_font($pdf, "Times-Roman", 30, 4);
pdf_set_text_rendering($pdf, 1);
PDF_show_xy($pdf, "Times Roman outlined", 50, 750);
pdf_moveto($pdf, 50, 740);
pdf_lineto($pdf, 330, 740);
pdf_stroke($pdf);
PDF_end_page($pdf);
PDF_close($pdf);
```

```
fclose($fp);
echo "<A HREF=getpdf.php3>finished</A>";
?>
```

The PHP script getpdf.php3 is the same as above.

The pdflib distribution contains a more complex example which creates a series of pages with an analog clock. This example converted into PHP using pdflib 2.x looks as the following (you can see the same example in the documentation for the [clipdf module](#)):

Ejemplo 3. pdfclock example from pdflib 2.x distribution

```
<?php
$pdffilename = "clock.pdf";
$radius = 200;
$margin = 20;
$pagecount = 40;

$fp = fopen($pdffilename, "w");
$pdf = pdf_open($fp);
pdf_set_info_creator($pdf, "pdf_clock.php3");
pdf_set_info_author($pdf, "Uwe Steinmann");
pdf_set_info_title($pdf, "Analog Clock");

while($pagecount- > 0) {
    pdf_begin_page($pdf, 2 * ($radius + $margin), 2 * ($radius + $margin));

    pdf_set_transition($pdf, 4); /* wipe */
    pdf_set_duration($pdf, 0.5);

    pdf_translate($pdf, $radius + $margin, $radius + $margin);
    pdf_save($pdf);
    pdf_setrgbcolor($pdf, 0.0, 0.0, 1.0);

    /* minute strokes */
    pdf_setlinewidth($pdf, 2.0);
    for ($alpha = 0; $alpha < 360; $alpha += 6) {
        pdf_rotate($pdf, 6.0);
        pdf_moveto($pdf, $radius, 0.0);
        pdf_lineto($pdf, $radius-$margin/3, 0.0);
        pdf_stroke($pdf);
    }

    pdf_restore($pdf);
    pdf_save($pdf);

    /* 5 minute strokes */
    pdf_setlinewidth($pdf, 3.0);
    for ($alpha = 0; $alpha < 360; $alpha += 30) {
        pdf_rotate($pdf, 30.0);
        pdf_moveto($pdf, $radius, 0.0);
        pdf_lineto($pdf, $radius-$margin, 0.0);
        pdf_stroke($pdf);
    }

    $ltime = getdate();

    /* draw hour hand */
    pdf_save($pdf);
    pdf_rotate($pdf,-(($ltime['minutes']/60.0)+$ltime['hours']-3.0)*30.0);
```

```

pdf_moveto($pdf, -$radius/10, -$radius/20);
pdf_lineto($pdf, $radius/2, 0.0);
pdf_lineto($pdf, -$radius/10, $radius/20);
pdf_closepath($pdf);
pdf_fill($pdf);
pdf_restore($pdf);

/* draw minute hand */
pdf_save($pdf);
pdf_rotate($pdf,-(($ltime['seconds']/60.0)+$ltime['minutes']-15.0)*6.0);
pdf_moveto($pdf, -$radius/10, -$radius/20);
pdf_lineto($pdf, $radius * 0.8, 0.0);
pdf_lineto($pdf, -$radius/10, $radius/20);
pdf_closepath($pdf);
pdf_fill($pdf);
pdf_restore($pdf);

/* draw second hand */
pdf_setrgbcolor($pdf, 1.0, 0.0, 0.0);
pdf_setlinewidth($pdf, 2);
pdf_save($pdf);
pdf_rotate($pdf, -((($ltime['seconds'] - 15.0) * 6.0)));
pdf_moveto($pdf, -$radius/5, 0.0);
pdf_lineto($pdf, $radius, 0.0);
pdf_stroke($pdf);
pdf_restore($pdf);

/* draw little circle at center */
pdf_circle($pdf, 0, 0, $radius/30);
pdf_fill($pdf);

pdf_restore($pdf);

pdf_end_page($pdf);
}

$pdf = pdf_close($pdf);
fclose($fp);
echo "<A HREF=getpdf.php3?filename=". $pdffilename . ">finished</A>";
?>

```

The PHP script getpdf.php3 just outputs the pdf document.

```

<?php
$fp = fopen($filename, "r");
header("Content-type: application/pdf");
fpassthru($fp);
fclose($fp);
?>

```

PDF_get_info (PHP 3>= 3.0.6, PHP 4 <= 4.0b1)

Returns an empty info structure for a pdf document

```
info pdf_get_info (string filename)
```

The **PDF_get_info()** function returns an empty info structure for the pdf document. It should be filled with appropriate information like the author, subject etc. of the document.

Nota: This functions is not available if pdflib 2.x support is activated.

See also **PDF_set_info_creator()**, **PDF_set_info_author()**, **PDF_set_info_keywords()**, **PDF_set_info_title()**, **PDF_set_info_subject()**.

PDF_set_info (PHP 4 >= 4.0.1)

Fills a field of the document information

```
void pdf_set_info (int pdf document, string fieldname, string value)
```

The **PDF_set_info()** function sets an information field of a pdf document. Possible values for the fieldname are 'Subject', 'Title', 'Creator', 'Author', 'Keywords' and one user-defined name. It can be called before beginning a page.

Ejemplo 1. Setting document information

```
<?php
$fd = fopen("test.pdf", "w");
$pdfdoc = pdf_open($fd);
pdf_set_info($pdfdoc, "Author", "Uwe Steinmann");
pdf_set_info($pdfdoc, "Creator", "Uwe Steinmann");
pdf_set_info($pdfdoc, "Title", "Testing Info Fields");
pdf_set_info($pdfdoc, "Subject", "Test");
pdf_set_info($pdfdoc, "Keywords", "Test, Fields");
pdf_set_info($pdfdoc, "CustomField", "What ever makes sense");
pdf_begin_page($pdfdoc, 595, 842);
pdf_end_page($pdfdoc);
pdf_close($pdfdoc);
?>
```

Nota: This function replaces **PDF_set_info_keywords()**, **PDF_set_info_title()**, **PDF_set_info_subject()**, **PDF_set_info_creator()**, **PDF_set_info_sybject()**.

PDF_open (PHP 3>= 3.0.6, PHP 4)

Opens a new pdf document

```
int pdf_open (int file, int info)
```

The **PDF_open()** function opens a new pdf document. The corresponding file has to be opened with **fopen()** and the file descriptor passed as argument *file*. *info* is the info structure that has to be created with **pdf_get_info()**. The info structure will be deleted within this function.

Nota: The return value is needed as the first parameter in all other functions writing to the pdf document.

Nota: This function does not allow the second parameter if pdflib 2.0 support is activated.

See also **fopen()**, **PDF_get_info()**, **PDF_close()**.

PDF_close (PHP 3>= 3.0.6, PHP 4)

Closes a pdf document

```
void pdf_close (int pdf document)
```

The **PDF_close()** function closes the pdf document.

Nota: Due to an unclean implementation of the pdflib 0.6 the internal closing of the document also closes the file. This should not be done because pdflib did not open the file, but expects an already open file when **PDF_open()** is called. Consequently it shouldn't close the file. In order to fix this just take out line 190 of the file p_basic.c in the pdflib 0.6 source distribution until the next release of pdflib will fix this.

Nota: This function works properly without any patches to pdflib if pdflib 2.0 support is activated.

See also **PDF_open()**, **fclose()**.

PDF_begin_page (PHP 3>= 3.0.6, PHP 4)

Starts new page

```
void pdf_begin_page (int pdf document, double width, double height)
```

The **PDF_begin_page()** function starts a new page with height *height* and width *width*. In order to create a valid document you must call this function and **PDF_end_page()**.

See also **PDF_end_page()**.

PDF_end_page (PHP 3>= 3.0.6, PHP 4)

Ends a page

```
void pdf_end_page (int pdf document)
```

The **PDF_end_page()** function ends a page. Once a page is ended it cannot be modified anymore.

See also **PDF_begin_page()**.

PDF_show (PHP 3>= 3.0.6, PHP 4)

Output text at current position

```
void pdf_show (int pdf_document, string text)
```

The **PDF_show()** function outputs the string *text* at the current position using the current font.

See also **PDF_show_xy()**, **PDF_set_text_pos()**, **PDF_set_font()**.

PDF_show_boxed (PHP 4 >= 4.0RC1)

Output text in a box

```
int pdf_show_boxed (int pdf_document, string text, double x-coor, double y-coor, double width, double height, string mode)
```

The **PDF_show_boxed()** function outputs the string *text* in a box with its lower left position at (*x-coor*, *y-coor*). The boxes dimension is *height* by *width*. The parameter *mode* determines how the text is type set. If *width* and *height* are zero, the *mode* can be "left", "right" or "center". If *width* or *height* is unequal zero it can also be "justify" and "fulljustify".

Returns the number of characters that could not be processed because they did not fit into the box.

See also **PDF_show()**, **PDF_show_xy()**.

PDF_show_xy (PHP 3>= 3.0.6, PHP 4)

Output text at given position

```
void pdf_show_xy (int pdf_document, string text, double x-coor, double y-coor)
```

The **PDF_show_xy()** function outputs the string *text* at position (*x-coor*, *y-coor*).

See also **PDF_show()**.

PDF_set_font (PHP 3>= 3.0.6, PHP 4)

Selects a font face and size

```
void pdf_set_font (int pdf_document, string font_name, double size, string encoding [, int embed])
```

The **PDF_set_font()** function sets the current font face, font size and encoding. If you use pdflib 0.6 you will need to provide the Adobe Font Metrics (afm-files) for the font in the font path (default is ./fonts). If you use php3 or a version of pdflib older than 2.20 the fourth parameter *encoding* can take the following values: 0 = builtin, 1 = pdfdoc, 2 =

macroman, 3 = macexpert, 4 = winansi. An encoding greater than 4 and less than 0 will default to winansi. winansi is often a good choice. If you use php4 and a version of pdflib >= 2.20 the encoding parameter has changed to a string. Use 'winansi', 'builtin', 'host', 'macroman' etc. instead. If the last parameter is set to 1 the font is embedded into the pdf document otherwise it is not. To embed a font is usually a good idea if the font is not widely spread and you cannot ensure that the person watching your document has access on fonts in the document. A font is only embedded once even if you call **PDF_set_font()** several times.

Nota: This function has to be called after **PDF_begin_page()** in order to create a valid pdf document.

Nota: If you reference a font in a .upr file make sure the name in the afm file and the font name are the same. Otherwise, the font will be embedded several times (Thanks to Paul Haddon for finding this.)

PDF_set_leading (PHP 3>= 3.0.6, PHP 4)

Sets distance between text lines

```
void pdf_set_leading (int pdf document, double distance)
```

The **PDF_set_leading()** function sets the distance between text lines. This will be used if text is output by **PDF_continue_text()**.

See also **PDF_continue_text()**.

PDF_set_parameter (PHP 4 >= 4.0RC1)

Sets certain parameters

```
void pdf_set_parameter (int pdf document, string name, string value)
```

The **PDF_set_parameter()** function sets several parameters of pdflib which are of the type string.

See also **PDF_get_value()**, **PDF_set_value()**, **PDF_get_parameter()**.

PDF_get_parameter (PHP 4 >= 4.0.1)

Gets certain parameters

```
string pdf_get_parameter (int pdf document, string name, double modifier)
```

The **PDF_get_parameter()** function gets several parameters of pdflib which are of the type string. The function parameter *modifier* characterizes the parameter to get. If the modifier is not needed it has to be 0.

See also **PDF_get_value()**, **PDF_set_value()**, **PDF_set_parameter()**.

PDF_set_value (PHP 4 >= 4.0.1)

Sets certain numerical value

```
void pdf_set_value (int pdf document, string name, double value)
```

The **PDF_set_value()** function sets several numerical parameters of pdflib.

See also **PDF_get_value()**, **PDF_get_parameter()**, **PDF_set_parameter()**.

PDF_get_value (PHP 4 >= 4.0.1)

Gets certain numerical value

```
double pdf_get_value (int pdf document, string name, double modifier)
```

The **PDF_get_value()** function gets several numerical parameters of pdflib. The function parameter *modifier* characterizes the parameter to get. If the modifier is not needed it has to be 0.

See also **PDF_set_value()**, **PDF_get_parameter()**, **PDF_set_parameter()**.

PDF_set_text_rendering (PHP 3>= 3.0.6, PHP 4)

Determines how text is rendered

```
void pdf_set_text_rendering (int pdf document, int mode)
```

The **PDF_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.

PDF_set_horiz_scaling (PHP 3>= 3.0.6, PHP 4)

Sets horizontal scaling of text

```
void pdf_set_horiz_scaling (int pdf document, double scale)
```

The **PDF_set_horiz_scaling()** function sets the horizontal scaling to *scale* percent.

PDF_set_text_rise (PHP 3>= 3.0.6, PHP 4)

Sets the text rise

```
void pdf_set_text_rise (int pdf document, double rise)
```

The **PDF_set_text_rise()** function sets the text rising to *rise* points.

PDF_set_text_matrix (PHP 3>= 3.0.6, PHP 4 <= 4.0b4)

Sets the text matrix

```
void pdf_set_text_matrix (int pdf_document, array matrix)
```

The **PDF_set_text_matrix()** function sets a matrix which describes a transformation applied on the current text font. The matrix has to be passed as an array with six elements.

PDF_set_text_pos (PHP 3>= 3.0.6, PHP 4)

Sets text position

```
void pdf_set_text_pos (int pdf_document, double x-coor, double y-coor)
```

The **PDF_set_text_pos()** function sets the position of text for the next **pdf_show()** function call.

See also **PDF_show()**, **PDF_show_xy()**.

PDF_set_char_spacing (PHP 3>= 3.0.6, PHP 4)

Sets character spacing

```
void pdf_set_char_spacing (int pdf_document, double space)
```

The **PDF_set_char_spacing()** function sets the spacing between characters.

See also **PDF_set_word_spacing()**, **PDF_set_leading()**.

PDF_set_word_spacing (PHP 3>= 3.0.6, PHP 4)

Sets spacing between words

```
void pdf_set_word_spacing (int pdf_document, double space)
```

The **PDF_set_word_spacing()** function sets the spacing between words.

See also **PDF_set_char_spacing()**, **PDF_set_leading()**.

PDF_skew (PHP 4 >= 4.0RC1)

Skews the coordinate system

```
void pdf_skew (int pdf_document, double alpha, double beta)
```

The **PDF_skew()** function skews the coordinate system by *alpha* (x) and *beta* (y) degrees. *alpha* and *beta* may not be 90 or 270 degrees.

PDF_continue_text (PHP 3>= 3.0.6, PHP 4)

Outputs text in next line

```
void pdf_continue_text (int pdf document, string text)
```

The **PDF_continue_text()** function outputs the string in *text* in the next line. The distance between the lines can be set with **PDF_set_leading()**.

See also **PDF_show_xy()**, **PDF_set_leading()**, **PDF_set_text_pos()**.

PDF_stringwidth (PHP 3>= 3.0.6, PHP 4)

Returns width of text using current font

```
double pdf_stringwidth (int pdf document, string text)
```

The **PDF_stringwidth()** function returns the width of the string in *text* by using the current font. It requires a font to be set before with **PDF_set_font()**.

See also **PDF_set_font()**.

PDF_save (PHP 3>= 3.0.6, PHP 4)

Saves the current environment

```
void pdf_save (int pdf document)
```

The **PDF_save()** function saves the current environment. It works like the postscript command gsave. Very useful if you want to translate or rotate an object without effecting other objects. **PDF_save()** should always be followed by **PDF_restore()** to restore the environment before **PDF_save()**.

See also **PDF_restore()**.

PDF_restore (PHP 3>= 3.0.6, PHP 4)

Restores formerly saved environment

```
void pdf_restore (int pdf document)
```

The **PDF_restore()** function restores the environment saved with **PDF_save()**. It works like the postscript command grestore.

Ejemplo 1. Save and Restore

```
<?php PDF_save($pdf);
// do all kinds of rotations, transformations, ...
PDF_restore($pdf) ?>
```

See also **PDF_save()**.

PDF_translate (PHP 3>= 3.0.6, PHP 4)

Sets origin of coordinate system

```
void pdf_translate (int pdf document, double x-coor, double y-coor)
```

The **PDF_translate()** function sets the origin of coordinate system to the point (*x-coor*, *y-coor*) relativ the current origin. The following example draws a line from (0, 0) to (200, 200) relative to the initial coordinate system. You have to set the current point after **PDF_translate()** and before you start drawing more objects.

Ejemplo 1. Translation

```
<?php PDF_moveto($pdf, 0, 0);
PDF_lineto($pdf, 100, 100);
PDF_stroke($pdf);
PDF_translate($pdf, 100, 100);
PDF_moveto($pdf, 0, 0);
PDF_lineto($pdf, 100, 100);
PDF_stroke($pdf);
?>
```

PDF_scale (PHP 3>= 3.0.6, PHP 4)

Sets scaling

```
void pdf_scale (int pdf document, double x-scale, double y-scale)
```

The **PDF_scale()** function sets the scaling factor in both directions. The following example scales x and y direction by 72. The following line will therefore be drawn one inch in both directions.

Ejemplo 1. Scaling

```
<?php PDF_scale($pdf, 72.0, 72.0);
PDF_lineto($pdf, 1, 1);
PDF_stroke($pdf);
?>
```

PDF_rotate (PHP 3>= 3.0.6, PHP 4)

Sets rotation

```
void pdf_rotate (int pdf document, double angle)
```

The **PDF_rotate()** function sets the rotation in degress to *angle*.

PDF_setflat (PHP 3>= 3.0.6, PHP 4)

Sets flatness

```
void pdf_setflat (int pdf_document, double value)
```

The **PDF_setflat()** function sets the flatness to a value between 0 and 100.

PDF_setlinejoin (PHP 3>= 3.0.6, PHP 4)

Sets linejoin parameter

```
void pdf_setlinejoin (int pdf_document, long value)
```

The **PDF_setlinejoin()** function sets the linejoin parameter between a value of 0 and 2.

PDF_setlinecap (PHP 3>= 3.0.6, PHP 4)

Sets linecap parameter

```
void pdf_setlinecap (int pdf_document, int value)
```

The **PDF_setlinecap()** function sets the linecap parameter between a value of 0 and 2.

PDF_setmiterlimit (PHP 3>= 3.0.6, PHP 4)

Sets miter limit

```
void pdf_setmiterlimit (int pdf_document, double value)
```

The **PDF_setmiterlimit()** function sets the miter limit to a value greater of equal than 1.

PDF_setlinewidth (PHP 3>= 3.0.6, PHP 4)

Sets line width

```
void pdf_setlinewidth (int pdf_document, double width)
```

The **PDF_setlinewidth()** function sets the line width to *width*.

PDF_setdash (PHP 3>= 3.0.6, PHP 4)

Sets dash pattern

```
void pdf_setdash (int pdf_document, double white, double black)
```

The **PDF_setdash()** function sets the dash pattern *white* white points and *black* black points. If both are 0 a solid line is set.

PDF_moveto (PHP 3>= 3.0.6, PHP 4)

Sets current point

```
void pdf_moveto (int pdf_document, double x-coor, double y-coor)
```

The **PDF_moveto()** function sets the current point to the coordinates *x-coor* and *y-coor*.

PDF_curveto (PHP 3>= 3.0.6, PHP 4)

Draws a curve

```
void pdf_curveto (int pdf_document, double x1, double y1, double x2, double y2, double x3,
double y3)
```

The **PDF_curveto()** function draws a Bezier curve from the current point to the point (*x3*, *y3*) using (*x1*, *y1*) and (*x2*, *y2*) as control points.

See also **PDF_moveto()**, **PDF_lineto()**, **PDF_stroke()**.

PDF_lineto (PHP 3>= 3.0.6, PHP 4)

Draws a line

```
void pdf_lineto (int pdf_document, double x-coor, double y-coor)
```

The **PDF_lineto()** function draws a line from the current point to the point with coordinates (*x-coor*, *y-coor*).

See also **PDF_moveto()**, **PDF_curveto()**, **PDF_stroke()**.

PDF_circle (PHP 3>= 3.0.6, PHP 4)

Draws a circle

```
void pdf_circle (int pdf_document, double x-coor, double y-coor, double radius)
```

The **PDF_circle()** function draws a circle with center at point (*x-coor*, *y-coor*) and radius *radius*.

See also **PDF_arc()**, **PDF_stroke()**.

PDF_arc (PHP 3>= 3.0.6, PHP 4)

Draws an arc

```
void pdf_arc (int pdf_document, double x-coor, double y-coor, double radius, double start,
double end)
```

The **PDF_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*.

See also **PDF_circle()**, **PDF_stroke()**.

PDF_rect (PHP 3>= 3.0.6, PHP 4)

Draws a rectangle

```
void pdf_rect (int pdf_document, double x-coor, double y-coor, double width, double
height)
```

The **PDF_rect()** function draws a rectangle with its lower left corner at point (*x-coor*, *y-coor*). This width is set to *width*. This height is set to *height*.

See also **PDF_stroke()**.

PDF_closepath (PHP 3>= 3.0.6, PHP 4)

Closes path

```
void pdf_closepath (int pdf_document)
```

The **PDF_closepath()** function closes the current path. This means, it draws a line from current point to the point where the first line was started. Many functions like **PDF_moveto()**, **PDF_circle()** and **PDF_rect()** start a new path.

PDF_stroke (PHP 3>= 3.0.6, PHP 4)

Draws line along path

```
void pdf_stroke (int pdf_document)
```

The **PDF_stroke()** function draws a line along current path. The current path is the sum of all line drawing. Without this function the line would not be drawn.

See also **PDF_closepath()**, **PDF_closepath_stroke()**.

PDF_closepath_stroke (PHP 3>= 3.0.6, PHP 4)

Closes path and draws line along path

```
void pdf_closepath_stroke (int pdf_document)
```

The **PDF_closepath_stroke()** function is a combination of **PDF_closepath()** and **PDF_stroke()**. It also clears the path.

See also **PDF_closepath()**, **PDF_stroke()**.

PDF_fill (PHP 3>= 3.0.6, PHP 4)

Fills current path

```
void pdf_fill (int pdf document)
```

The **PDF_fill()** function fills the interior of the current path with the current fill color.

See also **PDF_closepath()**, **PDF_stroke()**, **PDF_setgray_fill()**, **PDF_setgray()**, **PDF_setrgbcolor_fill()**, **PDF_setrgbcolor()**.

PDF_fill_stroke (PHP 3>= 3.0.6, PHP 4)

Fills and strokes current path

```
void pdf_fill_stroke (int pdf document)
```

The **PDF_fill_stroke()** function fills the interior of the current path with the current fill color and draws current path.

See also **PDF_closepath()**, **PDF_stroke()**, **PDF_fill()**, **PDF_setgray_fill()**, **PDF_setgray()**, **PDF_setrgbcolor_fill()**, **PDF_setrgbcolor()**.

PDF_closepath_fill_stroke (PHP 3>= 3.0.6, PHP 4)

Closes, fills and strokes current path

```
void pdf_closepath_fill_stroke (int pdf document)
```

The **PDF_closepath_fill_stroke()** function closes, fills the interior of the current path with the current fill color and draws current path.

See also **PDF_closepath()**, **PDF_stroke()**, **PDF_fill()**, **PDF_setgray_fill()**, **PDF_setgray()**, **PDF_setrgbcolor_fill()**, **PDF_setrgbcolor()**.

PDF_endpath (PHP 3>= 3.0.6, PHP 4)

Ends current path

```
void pdf_endpath (int pdf document)
```

The **PDF_endpath()** function ends the current path but does not close it.

See also **PDF_closepath()**.

PDF_clip (PHP 3>= 3.0.6, PHP 4)

Clips to current path

```
void pdf_clip (int pdf_document)
```

The **PDF_clip()** function clips all drawing to the current path.

PDF_setgray_fill (PHP 3>= 3.0.6, PHP 4)

Sets filling color to gray value

```
void pdf_setgray_fill (int pdf_document, double gray_value)
```

The **PDF_setgray_fill()** function sets the current gray value to fill a path.

See also **PDF_setrgbcolor_fill()**.

PDF_setgray_stroke (PHP 3>= 3.0.6, PHP 4)

Sets drawing color to gray value

```
void pdf_setgray_stroke (int pdf_document, double gray_value)
```

The **PDF_setgray_stroke()** function sets the current drawing color to the given gray value.

See also **PDF_setrgbcolor_stroke()**.

PDF_setgray (PHP 3>= 3.0.6, PHP 4)

Sets drawing and filling color to gray value

```
void pdf_setgray (int pdf_document, double gray_value)
```

The **PDF_setgray()** function sets the current drawing and filling color to the given gray value.

See also **PDF_setrgbcolor_stroke()**, **PDF_setrgbcolor_fill()**.

PDF_setrgbcolor_fill (PHP 3>= 3.0.6, PHP 4)

Sets filling color to rgb color value

```
void pdf_setrgbcolor_fill (int pdf_document, double red_value, double green_value, double blue_value)
```

The **PDF_setrgbcolor_fill()** function sets the current rgb color value to fill a path.

See also **PDF_setrgbcolor_fill()**.

PDF_setrgbcolor_stroke (PHP 3>= 3.0.6, PHP 4)

Sets drawing color to rgb color value

```
void pdf_setrgbcolor_stroke (int pdf document, double red value, double green value, double blue value)
```

The **PDF_setrgbcolor_stroke()** function sets the current drawing color to the given rgb color value.

See also **PDF_setrgbcolor_stroke()**.

PDF_setrgbcolor (PHP 3>= 3.0.6, PHP 4)

Sets drawing and filling color to rgb color value

```
void pdf_setrgbcolor (int pdf document, double red value, double green value, double blue value)
```

The **PDF_setrgbcolor_stroke()** function sets the current drawing and filling color to the given rgb color value.

See also **PDF_setrgbcolor_stroke()**, **PDF_setrgbcolor_fill()**.

PDF_add_outline (PHP 3>= 3.0.6, PHP 4)

Adds bookmark for current page

```
int pdf_add_outline (int pdf document, string text [, int parent [, int open]])
```

The **PDF_add_outline()** function adds a bookmark with text *text* that points to the current page. The bookmark is inserted as a child of *parent* and is by default open if *open* is not 0. The return value is an identifier for the bookmark which can be used as a parent for other bookmarks. Therefore you can build up hierarchies of bookmarks.

Unfortunately pdflib does not make a copy of the string, which forces PHP to allocate the memory. Currently this piece of memory is not been freed by any PDF function but it will be taken care of by the PHP memory manager.

PDF_set_transition (PHP 3>= 3.0.6, PHP 4)

Sets transition between pages

```
void pdf_set_transition (int pdf document, int transition)
```

The **PDF_set_transition()** 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)

See also **PDF_set_duration()**.

PDF_set_duration (PHP 3>= 3.0.6, PHP 4)

Sets duration between pages

```
void pdf_set_duration (int pdf document, double duration)
```

The **PDF_set_duration()** function set the duration between following pages in seconds.

See also **PDF_set_transition()**.

PDF_open_gif (PHP 3>= 3.0.7, PHP 4 >= 4.0b2)

Opens a GIF image

```
int pdf_open_gif (int pdf document, string filename)
```

The **PDF_open_gif()** function opens an image stored in the file with the name *filename*. The format of the image has to be gif. The function returns a pdf image identifier.

Ejemplo 1. Including a gif image

```
<?php
$im = PDF_open_gif($pdf, "test.gif");
pdf_place_image($pdf, $im, 100, 100, 1);
pdf_close_image($pdf, $im);
?>
```

See also **PDF_close_image()**, **PDF_open_jpeg()**, **PDF_open_memory_image()**, **PDF_execute_image()**, **PDF_place_image()**, **PDF_put_image()**.

PDF_open_png (PHP 4 >= 4.0RC2)

Opens a PNG image

```
int pdf_open_png (int pdf, string png_file)
```

The **PDF_open_png()** function opens an image stored in the file with the name *filename*. The format of the image has to be png. The function returns a pdf image identifier.

Ejemplo 1. Including a PNG image

```
<?php
$im = PDF_open_png ($pdf, "test.png");
```

```
pdf_place_image ($pdf, $im, 100, 100, 1);
pdf_close_image ($pdf, $im);
?>
```

See also **PDF_close_image()**, **PDF_open_jpeg()**, **PDF_open_gif()**, **PDF_open_memory_image()**, **PDF_execute_image()**, **PDF_place_image()**, **PDF_put_image()**.

PDF_open_memory_image (PHP 3>= 3.0.10, PHP 4 >= 4.0b2)

Opens an image created with PHP's image functions

```
int pdf_open_memory_image (int pdf document, int image)
```

The **PDF_open_memory_image()** function takes an image created with the PHP's image functions and makes it available for the pdf document. The function returns a pdf image identifier.

Ejemplo 1. Including a memory image

```
<?php
$im = ImageCreate(100, 100);
$col = ImageColorAllocate($im, 80, 45, 190);
ImageFill($im, 10, 10, $col);
$pim = PDF_open_memory_image($pdf, $im);
ImageDestroy($im);
pdf_place_image($pdf, $pim, 100, 100, 1);
pdf_close_image($pdf, $pim);
?>
```

See also **PDF_close_image()**, **PDF_open_jpeg()**, **PDF_open_gif()**, **PDF_open_png()** **PDF_execute_image()**, **PDF_place_image()**, **PDF_put_image()**.

PDF_open_jpeg (PHP 3>= 3.0.7, PHP 4 >= 4.0b2)

Opens a JPEG image

```
int pdf_open_jpeg (int pdf document, string filename)
```

The **PDF_open_jpeg()** function opens an image stored in the file with the name *filename*. The format of the image has to be jpeg. The function returns a pdf image identifier.

See also **PDF_close_image()**, **PDF_open_gif()**, **PDF_open_png()**, **PDF_open_memory_image()**, **PDF_execute_image()**, **PDF_place_image()**, **PDF_put_image()**.

PDF_close_image (PHP 3>= 3.0.7, PHP 4 >= 4.0b2)

Closes an image

```
void pdf_close_image (int image)
```

The **PDF_close_image()** function closes an image which has been opened with any of the **PDF_open_xxx()** functions.
 See also **PDF_open_jpeg()**, **PDF_open_gif()**, **PDF_open_memory_image()**.

PDF_place_image (PHP 3>= 3.0.7, PHP 4 >= 4.0b2)

Places an image on the page

```
void pdf_place_image (int pdf document, int image, double x-coor, double y-coor, double scale)
```

The **PDF_place_image()** function places an image on the page at position (*x-coor*, *y-coor*). The image can be scaled at the same time.

See also **PDF_put_image()**.

PDF_put_image (PHP 3>= 3.0.7, 4.0b2 - 4.0b4 only)

Stores an image in the PDF for later use

```
void pdf_put_image (int pdf document, int image)
```

The **PDF_put_image()** function places an image in the PDF file without showing it. The stored image can be displayed with the **PDF_execute_image()** function as many times as needed. This is useful when using the same image multiple times in order to keep the file size small. Using **PDF_put_image()** and **PDF_execute_image()** is highly recommended for larger images (several kb) if they show up more than once in the document.

Nota: This function has become meaningless with version 2.01 of pdflib. It will just output a warning.

See also **PDF_put_image()**, **PDF_place_image()**, **PDF_execute_image()**.

PDF_execute_image (PHP 3>= 3.0.7, 4.0b2 - 4.0b4 only)

Places a stored image on the page

```
void pdf_execute_image (int pdf document, int image, double x-coor, double y-coor, double scale)
```

The **PDF_execute_image()** function displays an image that has been put in the PDF file with the **PDF_put_image()** function on the current page at the given coordinates.

The image can be scaled while displaying it. A scale of 1.0 will show the image in the original size.

Nota: This function has become meaningless with version 2.01 of pdflib. It will just output a warning.

Ejemplo 1. Multiple show of an image

```
<?php
$im = ImageCreate(100, 100);
$col1 = ImageColorAllocate($im, 80, 45, 190);
ImageFill($im, 10, 10, $col1);
$pim = PDF_open_memory_image($pdf, $im);
pdf_put_image($pdf, $pim);
pdf_execute_image($pdf, $pim, 100, 100, 1);
pdf_execute_image($pdf, $pim, 200, 200, 2);
pdf_close_image($pdf, $pim);
?>
```

pdf_add_annotation (PHP 3>= 3.0.12, PHP 4 >= 4.0b2)

Adds annotation

```
void pdf_add_annotation (int pdf document, double llx, double lly, double urx, double ury,
string title, string content)
```

The **pdf_add_annotation()** adds a note with the lower left corner at (*llx*, *lly*) and the upper right corner at (*urx*, *ury*).

PDF_set_border_style (PHP 3>= 3.0.12, PHP 4 >= 4.0b2)

Sets style of border around links and annotations

```
void pdf_set_border_style (int pdf document, string style, double width)
```

The **PDF_set_border_style()** function sets the style and width of the surrounding box of links and annotations. The parameter *style* can be 'solid' or 'dashed'.

See also **PDF_set_border_color()**, **PDF_set_border_dash()**.

PDF_set_border_color (PHP 3>= 3.0.12, PHP 4 >= 4.0b2)

Sets color of border around links and annotations

```
void pdf_set_border_color (int pdf document, double red, double green, double blue)
```

The **PDF_set_border_color()** function sets the color of the surrounding box of links and annotations. The three color components have to have a value between 0.0 and 1.0.

See also **PDF_set_border_style()**, **PDF_set_border_dash()**.

PDF_set_border_dash (PHP 4 >= 4.0.1)

Sets dash style of border around links and annotations

```
void pdf_set_border_dash (int pdf document, double black, double white)
```

The **PDF_set_border_dash()** function sets the lenght of black and white areas of a dashed line of the suroundig box of links and annotations.

See also **PDF_set_border_style()**, **PDF_set_border_color()**.

LV. Verisign Payflow Pro functions

This extension allows you to process credit cards and other financial transactions using Verisign Payment Services, formerly known as Signio (<http://www.verisign.com/payment/>).

These functions are only available if PHP has been compiled with the `-with-pfpro[=DIR]` option. You will require the appropriate SDK for your platform, which may be downloaded from within the manager interface (https://testmanager.signio.com/Downloads/Downloads_secure.htm) once you have registered.

Once you have downloaded the SDK you should copy the files from the `lib` directory of the distribution. Copy the header file `pfpro.h` to `/usr/local/include` and the library file `libpfpro.so` to `/usr/local/lib`.

When using these functions, you may omit calls to `pfpro_init()` and `pfpro_cleanup()` as this extension will do so automatically if required. However the functions are still available in case you are processing a number of transactions and require fine control over the library. You may perform any number of transactions using `pfpro_process()` between the two.

These functions have been added in PHP 4.0.2.

Nota: These functions only provide a link to Verisign Payment Services. Be sure to read the Payflow Pro Developers Guide for full details of the required parameters.

pfprom_init (PHP 4 >= 4.0.2)

Initialises the Payflow Pro library

```
void pfprom_init(void);
```

pfprom_init() is used to initialise the Payflow Pro library. You may omit this call, in which case this extension will automatically call **pfprom_init()** before the first transaction.

See also **pfprom_cleanup()**.

pfprom_cleanup (PHP 4 >= 4.0.2)

Shuts down the Payflow Pro library

```
void pfprom_cleanup(void);
```

pfprom_cleanup() is used to shutdown the Payflow Pro library cleanly. It should be called after you have processed any transactions and before the end of your script. However you may omit this call, in which case this extension will automatically call **pfprom_cleanup()** after your script terminates.

See also **pfprom_init()**.

pfprom_process (PHP 4 >= 4.0.2)

Process a transaction with Payflow Pro

```
array pfprom_process (array parameters [, string address [, int port [, int timeout [, string proxy address [, int proxy port [, string proxy logon [, string proxy password]]]]]]])
```

Returns: An associative array containing the response

pfprom_process() processes a transaction with Payflow Pro. The first parameter is an associative array containing keys and values that will be encoded and passed to the processor.

The second parameter is optional and specifies the host to connect to. By default this is "test.signio.com", so you will certainly want to change this to "connect.signio.com" in order to process live transactions.

The third parameter specifies the port to connect on. It defaults to 443, the standard SSL port.

The fourth parameter specifies the timeout to be used, in seconds. This defaults to 30 seconds. Note that this timeout appears to only begin once a link to the processor has been established and so your script could potentially continue for a very long time in the event of DNS or network problems.

The fifth parameter, if required, specifies the hostname of your SSL proxy. The sixth parameter specifies the port to use.

The seventh and eighth parameters specify the logon identity and password to use on the proxy.

The function returns an associative array of the keys and values in the response.

Nota: Be sure to read the Payflow Pro Developers Guide for full details of the required parameters.

Ejemplo 1. Payflow Pro example

```
<?php

pfpromo_init();

$transaction = array(USER => 'mylogin',
    PWD => 'mypassword',
    TRXTYPE => 'S',
    TENDER => 'C',
    AMT => 1.50,
    ACCT => '4111111111111111',
    EXPDATE => '0904'
);

$response = pfpromo_process($transaction);

if (!$response) {
    die("Couldn't establish link to Verisign.\n");
}

echo "Verisign response code was ".$response[RESULT];
echo ", which means: ".$response[RESPMSG]."\n";

echo "\nThe transaction request: ";
print_r($transaction);

echo "\nThe response: ";
print_r($response);

pfpromo_cleanup();

?>
```

pfpromo_process_raw (PHP 4 >= 4.0.2)

Process a raw transaction with Payflow Pro

```
string pfpromo_process_raw (string parameters [, string address [, int port [, int timeout [, string proxy address [, int proxy port [, string proxy logon [, string proxy password]]]]]]])
```

Returns: A string containing the response.

pfpromo_process_raw() processes a raw transaction string with Payflow Pro. You should really use **pfpromo_process()** instead, as the encoding rules of these transactions are non-standard.

The first parameter in this case is a string containing the raw transaction request. All other parameters are the same as with **pfpromo_process()**. The return value is a string containing the raw response.

Nota: Be sure to read the Payflow Pro Developers Guide for full details of the required parameters and encoding rules. You would be well advised to use **pfpromo_process()** instead.

Ejemplo 1. Payflow Pro raw example

```
<?php
```

```
pffpro_init();

$response = pffpro_process( "USER=mylogin&PWD[ 5 ]=m&ndy&TRXTYPE=S&TENDER=C&AMT=1.50&ACCT=4111111111111111

if (!$response) {
    die("Couldn't establish link to Verisign.\n");
}

echo "Verisign raw response was ".$response;

pffpro_cleanup();

?>
```

pffpro_version (PHP 4 >= 4.0.2)

Returns the version of the Payflow Pro software

```
string pffpro_version(void);
```

pffpro_version() returns the version string of the Payflow Pro library. At the time of writing, this was L211.

LVI. opciones e información de PHP

extension_loaded (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

averigua si una extensión ha sido cargada

```
bool extension_loaded (string name)
```

Devuelve true si la extensión identificada por *name* (nombre) está cargada. Puede ver el nombre de varias extensiones utilizando **phpinfo()**.

Véase también **phpinfo()**.

Nota: Esta función fue añadida en 3.0.10.

getenv (PHP 3, PHP 4)

Obtiene el valor de una variable de entorno

```
string getenv (string varname)
```

Devuelve el valor de la variable de entorno *varname*, o false en caso de error.

```
$ip = getenv("REMOTE_ADDR"); // get the ip number of the user
```

Puede ver una lista de todas las variables de entorno utilizando **phpinfo()**. Puede encontrar el significado de la mayoría echando un vistazo en CGI specification (especificación CGI) (<http://hoohoo.ncsa.uiuc.edu/cgi/>), especialmente en page on environmental variables (página de variables de entorno) (<http://hoohoo.ncsa.uiuc.edu/cgi/env.html>).

get_cfg_var (PHP 3, PHP 4)

Obtiene el valor de una opción de configuración de PHP.

```
string get_cfg_var (string varname)
```

Devuelve el valor actual de una variable de configuración de PHP especificada en *varname*, o false si ocurre un error.

No devolverá información de la configuracion cuando el PHP fue compilado, o leído desde un fichero de configuración Apache (utilizando las directivas `php3_configuration_option` directives).

Para comprobar si el sistema está utilizando un **fichero de configuración**, intente recuperar el valor de `cfg_file_path`. Si está disponible, se está utilizando un fichero de configuración.

get_current_user (PHP 3, PHP 4)

Obtiene el nombre del propietario del script PHP actual.

```
string get_current_user (void)
```

Devuelve el nombre del propietario del script PHP actual.

Véase también **getmyuid()**, **getmypid()**, **getmyinode()**, y **getlastmod()**.

get_magic_quotes_gpc (PHP 3>= 3.0.6, PHP 4)

Obtiene el valor de la configuración activa actual de las comillas mágicas gpc.

```
long get_magic_quotes_gpc (void)
```

Devuelve el valor de la configuración activa actual de **magic_quotes_gpc**. (0 desactivado, 1 activado)

Véase también **get_magic_quotes_runtime()**, **set_magic_quotes_runtime()**.

get_magic_quotes_runtime (PHP 3>= 3.0.6, PHP 4)

Obtiene el valor de la configuración activa actual de **magic_quotes_runtime**.

```
long get_magic_quotes_runtime (void)
```

Devuelve el valor de la configuración activa actual de **magic_quotes_runtime**. (0 desactivado, 1 activado)

Véase también **get_magic_quotes_gpc()**, **set_magic_quotes_runtime()**.

getlastmod (PHP 3, PHP 4)

Recupera la fecha/hora de la última modificación de la página.

```
int getlastmod (void)
```

Devuelve la fecha/hora de la última modificación de la página actual. El valor devuelto está en formato de fecha/hora Unix, adecuado para que sirva a **date()**. Devuelve false en caso de error.

Ejemplo 1. ejemplo getlastmod()

```
// outputs e.g. 'Last modified: March 04 1998 20:43:59.'
echo "Last modified: ".date( "F d Y H:i:s.", getlastmod() );
```

Véase también **date()**, **getmyuid()**, **get_current_user()**, **getmyinode()**, y **getmypid()**.

getmyinode (PHP 3, PHP 4)

Recupera el inodo del script actual.

```
int getmyinode (void)
```

Devuelve el inodo del script actual, o false en caso de error.

Véase también **getmyuid()**, **get_current_user()**, **getmypid()**, y **getlastmod()**.

getmypid (PHP 3, PHP 4)

Obtiene el ID de proceso de PHP.

```
int getmypid (void)
```

Devuelve el ID del proceso PHP actual, o false en caso de error.

Advierta que cuando se ejecuta como un módulo de servidor, diferentes llamadas del script no garantizan que tengan distintos pids.

Véase también **getmyuid()**, **get_current_user()**, **getmyinode()**, y **getlastmod()**.

getmyuid (PHP 3, PHP 4)

Obtiene el UID del propietario del script PHP.

```
int getmyuid (void)
```

Devuelve el ID de usuario del script actual, o false en caso de error.

Véase también **getmypid()**, **get_current_user()**, **getmyinode()**, y **getlastmod()**.

getrusage (PHP 3>= 3.0.7, PHP 4 >= 4.0b2)

Obtiene el consumo actual de recursos.

```
array getrusage ([int who])
```

Es un interface a getrusage(2). Devuelve un array asociativo que contiene los datos devueltos de la llamada del sistema. Si who (quien) es 1, getrusage debería llamarse con RUSAGE_CHILDREN. Todas las entradas son accesibles utilizando sus nombres de campo documentados.

Ejemplo 1. Ejemplo Getrusage

```
$dat = getrusage();
echo $dat["ru_nswap"];           # number of swaps
echo $dat["ru_majflt"];          # number of page faults
echo $dat["ru_utime.tv_sec"];    # user time used (seconds)
echo $dat["ru_utime.tv_usec"];   # user time used (microseconds)
```

Vea la página man de system para más detalles.

phpinfo (PHP 3, PHP 4)

Recupera gran cantidad de información de PHP.

```
int phpinfo (void)
```

Obtiene gran cantidad de información sobre el estado actual de PHP. Esto incluye información sobre las opciones de compilación y extensiones de PHP, la versión PHP, información y entorno del servidor (si está compilado como un módulo), el entorno PHP, información sobre la versión del SO, rutas, opciones de configuración maestras y locales, cabeceras HTTP, y la Licencia Pública GNU.

Véase también **phpversion()**.

phpversion (PHP 3, PHP 4)

Obtiene la versión actual de PHP.

```
string phpversion (void)
```

Devuelve una cadena de caracteres que contiene la versión del parser PHP que está ejecutándose actualmente.

Ejemplo 1. ejemplo phpversion()

```
// prints e.g. 'Current PHP version: 3.0rel-dev'  
echo "Current PHP version: ".phpversion();
```

Véase también **phpinfo()**.

php_logo_guid (PHP 4 >= 4.0b4)

Obtiene el guid logo

```
string php_logo_guid (void)
```

Nota: Esta funcionalidad fue añadida en PHP4 Beta 4.

putenv (PHP 3, PHP 4)

Establece el valor de una variable de entorno.

```
void putenv (string setting)
```

Añade *setting* (*valor*) al entorno.

Ejemplo 1. Establecer una Variable de Entorno

```
putenv( "UNIQID=$uniqid" );
```

set_magic_quotes_runtime (PHP 3>= 3.0.6, PHP 4)

Establece el valor de la configuración activa actual de magic_quotes_runtime.

```
long set_magic_quotes_runtime (int new_setting)
```

Establece el valor de la configuración activa actual de [magic_quotes_runtime](#). (0 desactivado, 1 activado)

Véase también [get_magic_quotes_gpc\(\)](#), [get_magic_quotes_runtime\(\)](#).

set_time_limit (PHP 3, PHP 4)

limita el tiempo máximo de ejecución

```
void set_time_limit (int seconds)
```

Establece el número de segundos que se le permite a un script ejecutarse. Si éste es alcanzado, el script devuelve un error de tipo fatal. El límite por defecto es 30 segundos o, si existe, el valor max_execution_time definido en el [fichero de configuración](#). Si seconds (segundos) se establece a cero, no se impone ningún límite.

Cuando se llama, **set_time_limit()** reinicia el contador del timeout a cero. En otras palabras, si el timeout es el de por defecto de 30 segundos, y después de 25 segundos de ejecución del script se realiza una llamada **set_time_limit(20)**, el script se ejecutará durante un total de 45 segundos antes de alcanzar su límite.

Advierta que **set_time_limit()** no tiene efecto cuando PHP se ejecuta en modo seguro (safe mode). No hay otra opción que que desactivar el modo seguro o cambiar el límite de tiempo en el [fichero de configuración](#).

zend_logo_guid (PHP 4 >= 4.0b4)

Obtiene el guid zend

```
string zend_logo_guid (void)
```

Nota: Esta funcionalidad fue añadida en PHP4 Beta 4.

LVII. Funciones POSIX

Este módulo contiene una interfaz a aquellas funciones definidas en el documento estandar IEEE 1003.1 (POSIX.1) que no son accesibles de otra manera. POSIX.1 por ejemplo definió las funciones open(),read(), write() y close(), las cuales han sido parte de PHP durante mucho tiempo. Algunas funciones específicas del sistema no habian estado disponibles antes, aunque con este módulo se intenta remediar esto ofreciendo un acceso fácil a esas funciones.

posix_kill (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Manda una señal a un proceso

```
bool posix_kill (int pid, int sig)
```

Manda la señal *sig* al proceso con el identificador de proceso *pid*. Devuelve FALSE, si no puede enviar la señal. Si sí la envia devuelve TRUE .

Vea también la página de manual kill(2) de su sistema POSIX, la cual contiene información adicional sobre los identificadores de proceso negativos, el pid especial 0, el pid especial -1, y la señal numero 0.

posix_getpid (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Devuelve el identificador del proceso actual

```
int posix_getpid (void )
```

Devuelve el identificador de proceso del proceso actual.

posix_getppid (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Devuelve el identificador del proceso padre

```
int posix_getppid (void )
```

Devuelve el identificador de proceso del proceso padre del proceso actual.

posix_getuid (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Devuelve el ID de usuario real del proceso actual

```
int posix_getuid (void )
```

Devuelve el valor numerico ID de usuario real del proceso actual. Vea también **posix_getpwuid()** para información sobre como convertir este ID en un nombre de usuario manejable.

posix_geteuid (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Devuelve el ID de usuario efectivo del proceso actual

```
int posix_geteuid (void )
```

Devuelve el valor numérico ID de usuario efectivo del proceso actual. Vea también **posix_getpwuid()** para información sobre como convertir este número en un nombre de usuario manejable.

posix_getgid (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Devuelve el ID de grupo real del proceso actual

```
int posix_getgid (void )
```

Devuelve el valor numérico ID de grupo real del proceso actual. Vea también **posix_getgrgid()** para información sobre como convertir esto en un nombre de grupo manejable.

posix_getegid (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Devuelve el ID de grupo efectivo del proceso actual

```
int posix_getegid (void )
```

Devuelve el valor numérico ID de grupo efectivo del proceso actual. Vea también **posix_getgrgid()** para información sobre como convertir este número en un nombre de grupo manejable.

posix_setuid (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Asigna el UID efectivo del proceso actual

```
bool posix_setuid (int uid)
```

Asigna el ID de usuario real al proceso actual. Esta es una función privilegiada y necesitas los privilegios apropiados (normalmente root) en tu sistema para realizar esta función.

Devuelve TRUE si tiene éxito, FALSE en caso contrario. Vea también **posix_setgid()**.

posix_setgid (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Asigna el GID efectivo del proceso actual

```
bool posix_setgid (int gid)
```

Asigna el ID de grupo real del proceso actual. Esta es una función privilegiada y necesitas los privilegios apropiados (normalmente root) en tu sistema para realizar esta función. El orden apropiado de llamada es **posix_setgid()** primero, **posix_setuid()** después.

Devuelve TRUE si tiene éxito, FALSE en caso contrario.

posix_getgroups (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Devuelve el conjunto de grupos del proceso actual

```
array posix_getgroups (void )
```

Devuelve un vector de enteros que contiene los ids numéricos de grupo de el conjunto de grupos del proceso actual. Vea también **posix_getgrgid()** para información sobre como convertir esto en nombres de grupo manejables.

posix_getlogin (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Devuelve el nombre de usuario

```
string posix_getlogin (void )
```

Devuelve el nombre de usuario (login) que es dueño del proceso actual. Vea **posix_getpwnam()** para información sobre como conseguir mas datos de este usuario.

posix_getpgrp (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Devuelve el identificador de grupo del proceso actual

```
int posix_getpgrp (void )
```

Devuelve el identificador de grupo de proceso del proceso actual. Vea POSIX.1 y la página de manual getpgrp(2) de su sistema POSIX para más información de grupos de procesos.

posix_setsid (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Convierte el proceso actual en lider de sesión

```
int posix_setsid (void )
```

Convierte el proceso actual en lider de sesión. Vea POSIX.1 y la página de manual setsid(2) en su sistema POSIX para más informacion en grupos de proceso y control de trabajos. Devuelve el id de sesión.

posix_setpgid (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Asigna el id de grupo de procesos para el control de trabajos

```
int posix_setpgid (int pid, int pgid)
```

Inserta el proceso *pid* en el grupo de procesos *pgid*. Vea POSIX.1 y la página de manual setsid(2) de su sistema POSIX para más información sobre grupos de procesos y control de trabajo. Devuelve TRUE si tiene éxito y FALSE en caso contrario.

posix_getpgid (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Recoge el id del grupo de procesos para el control de trabajo

```
int posix_getpgid (int pid)
```

Devuelve el identificador de grupo de procesos del proceso *pid*.

Esta no es una función POSIX, pero es normal en sistemas BSD y System V. Si su sistema no soporta esta función a nivel de sistema, esta función PHP devolverá siempre FALSE.

posix_getsid (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Consigue el sid actual del proceso

```
int posix_getsid (int pid)
```

Devuelve el sid del proceso *pid*. Si *pid* es 0, se devolverá el sid del proceso actual.

Esta no es una función POSIX, pero es normal en sistemas System V. Si su sistema no soporta esta función a nivel de sistema, esta función PHP devolverá siempre FALSE.

posix_uname (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Consigue el nombre del sistema

```
array posix_uname (void )
```

Devuelve un hash de cadenas con información sobre el sistema. Los índices del hash son

- sysname - nombre del sistema operativo (e.g. Linux)
- nodename - nombre del sistema (e.g. valiant)
- release - release del sistema operativo (e.g. 2.2.10)
- version - versión del sistema operativo (e.g. #4 Tue Jul 20 17:01:36 MEST 1999)
- machine - arquitectura del sistema (e.g. i586)

Posix requiere que usted no debe hacer ninguna suposición sobre el formato de los valores, por ejemplo usted no puede confiar en los tres dígitos de la version o cualquier cosa devuelta por esta función.

posix_times (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Recoge el tiempo de los procesos

```
array posix_times (void )
```

Devuelve un hash de cadenas con información sobre el uso de CPU del proceso actual. Los índices del hash son

- ticks - el numero de ticks de reloj que han pasado desde el reinicio.
- utime - tiempo de usuario usado por el proceso actual.
- stime - tiempo de sistema usado por el proceso actual.
- cutime - tiempo de usuario usado por el proceso actual e hijos.
- cstime - tiempo de sistema usado por el proceso actual e hijos.

posix_ctermid (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Recoge el nombre de ruta de la terminal de control

```
string posix_ctermid (void )
```

Necesita ser escrito.

posix_ttyname (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Determina el nombre del dispositivo terminal

```
string posix_ttyname (int fd)
```

Necesita ser escrito.

posix_isatty (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Determina si un descriptor de fichero esta en una terminal interactiva

```
bool posix_isatty (int fd)
```

Necesita ser escrito.

posix_getcwd (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Nombre de ruta del directorio actual

```
string posix_getcwd (void )
```

Necesita ser escrito cuanto antes.

posix_mkfifo (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Crea un fichero especial fifo (los llamados pipe o tuberias)

```
bool posix_getcwd (string pathname, int mode)
```

Necesita ser escrito lo más pronto posible.

posix_getgrnam (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Devuelve información sobre un grupo a través del nombre

```
array posix_getgrnam (string name)
```

Necesita ser escrito.

posix_getgrgid (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Devuelve información sobre un grupo a través del id de grupo

```
array posix_getgrgid (int gid)
```

Necesita ser escrito.

posix_getpwnam (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Devuelve información sobre un usuario a través del nombre de usuario

```
array posix_getpwnam (string username)
```

Devuelve un vector asociativo conteniendo información sobre un usuario referenciado por un nombre alfanumérico, pasado a la función en el parámetro *username*.

Los elementos del vector devuelto son:

Tabla 1. El vector de información de usuario

Elemento	Descripción
name	El elemento name contiene el nombre de usuario del usuario. Este es un nombre, normalmente menor de 16 caracteres, que no es su nombre completo, pero identifica al usuario. Este debe ser el mismo que el parámetro <i>username</i> usado en la llamada a la función y por lo tanto es redundante.
passwd	El elemento passwd contiene la contraseña del usuario en un formato encriptado. Normalmente, por ejemplo en un sistema que este utilizando contraseñas "shadow", devolverá un asterisco.
uid	El ID de usuario del usuario en formato numérico.
gid	El ID de grupo del usuario. Utiliza la función posix_getgrgid() para resolver el nombre del grupo y una lista de sus miembros.

Elemento	Descripción
gecos	GECOS es un término obsoleto que se refiere al campo apuntado de información en un sistema de procesamiento batch Honeywell. El campo y sus contenidos han sido formalizado por POSIX y contiene una lista separada por comas con el nombre completo del usuario, teléfono del trabajo, número de oficina y teléfono de casa. En muchos sistemas solo está disponible el nombre completo del usuario.
dir	Este elemento contiene la ruta absoluta al directorio del usuario (directorio home).
shell	El elemento shell contiene la ruta absoluta al ejecutable del shell por defecto del usuario.

posix_getpwuid (PHP 3>= 3.0.13, PHP 4 >= 4.0b4)

Devuelve información sobre un usuario a través de su id

```
array posix_getpwuid (int uid)
```

Devuelve un vector asociativo que contiene información sobre un usuario referenciado con un ID de usuario, pasado por el parámetro *uid*.

Los elementos del array son:

Tabla 1. El vector de información del usuario

Elemento	Descripción
name	El elemento name contiene el nombre de usuario del usuario. Este es un nombre, normalmente menor de 16 caracteres, que no es su verdadero nombre.
passwd	El elemento passwd contiene la contraseña del usuario en un formato encriptado. Normalmente, por ejemplo en un sistema con contraseñas "shadow", devolverá un asterisco.
uid	ID del usuario, debe ser el mismo que el parámetro <i>uid</i> usado en la llamada a la función, y por lo tanto redundante.
gid	El ID del grupo del usuario. Utiliza la función posix_getgrgid() para resolver el nombre del grupo y una lista de sus miembros.
gecos	GECOS es un término obsoleto que se refiere al campo apuntado de información en un sistema de procesamiento batch Honeywell. El campo y sus contenidos han sido formalizados por POSIX y contiene una lista separada por comas con el nombre completo del usuario, teléfono del trabajo, número de oficina y teléfono de casa. En muchos sistemas solo está disponible el nombre completo del usuario.
dir	Este elemento contiene la ruta absoluta al directorio del usuario (directorio home).

Elemento	Descripción
shell	El elemento shell contiene la ruta absoluta al ejecutable del shell por defecto del usuario.

posix_getrlimit (PHP 3>= 3.0.10, PHP 4 >= 4.0b4)

Devuelve información sobre los límites de recursos del sistema

```
array posix_getrlimit (void )
```

Necesita ser escrita tan pronto como sea posible.

LVIII. Funciones de PostgreSQL

Postgres, desarrollado originalmente en el UC Berkeley Computer Science Department, ha sido pionero en muchos de los conceptos relacionales/orientados a objeto que ahora están empezando a estar disponibles en algunas bases de datos comerciales. Tiene soporte de lenguaje SQL92/SQL3, integridad transaccional, y extensibilidad de tipos. PostgreSQL es un descendiente de dominio público, más concretamente open source, del código original de Berkeley.

PostgreSQL se encuentra disponible sin coste alguno. La versión actual la tienes a tu disposición en www.PostgreSQL.org (<http://www.postgresql.org/>).

Desde la versión 6.3 (02/03/1998) PostgreSQL usa sockets tipo Unix. Abajo se da una tabla con las diferentes posibilidades. El socket se encuentra en el ficheiro /tmp/.s.PGSQL.5432. Esta opción se controla mediante el flag '-i' del **postmaster** y cuando se incluye significa "escuchar sockets TCP/IP además de los de dominio Unix" ya que si no se le dice nada solo escucha sockets tipo Unix.

Tabla 1. Postmaster y PHP

Postmaster	PHP	Estado
postmaster &	pg_connect("", "", "", "", "dbname");	OK
postmaster -i &	pg_connect("", "", "", "", "dbname");	OK
postmaster &	pg_connect("localhost", "", "", "", "dbname");	Unable to connect to PostgreSQL server: connectDB() failed: Is the postmaster running and accepting TCP/IP (with -i) connection at 'localhost' on port '5432'? in /path/to/file.php3 on line 20. (Imposible conectar al servidor PostgreSQL, la llamada connectDB() ha fallado: ¿Está funcionando el postmaster aceptando conexiones TCP/IP (con -i) en 'localhost' en el puerto '5432'? en /path/to/file.php3 en linea 20.)
postmaster -i &	pg_connect("localhost", "", "", "", "dbname");	OK

Uno puede establecer una conexión con el siguiente comando:

Para usar el interface de objetos grandes (large object o lo), es necesario encapsularlo en un bloque de transacción. Un bloque de transacción empieza con un **begin** y si la transacción fue valida termina con **commit** y **end**. Si la transacción falla debe ser cerrada con **abort** y **rollback**.

Ejemplo 1. Usando Objetos Grandes (lo)

```
<?php
$database = pg_Connect ("", "", "", "", "jacarta");
pg_exec ($database, "begin");
$oid = pg_locreate ($database);
echo ("$oid\n");
$handle = pg_loopen ($database, $oid, "w");
echo ("$handle\n");
pg_lowrite ($handle, "gaga");
pg_loclose ($handle);
pg_exec ($database, "commit")
pg_exec ($database, "end")
?>
```


pg_Close (PHP 3, PHP 4)

Cierra una conexión PostgreSQL

```
bool pg_close (int connection)
```

Devuelve false si connection no es un indice de conexión valido y true en cualquier otro caso. Cierra la conexión a la base de datos PostgreSQL asociada con el indice de conexión pasado como parámetro.

pg_CmdTuples (PHP 3, PHP 4)

Devuelve el número de tuplas afectadas

```
int pg_cmdtuples (int result_id)
```

pg_CmdTuples() devuelve el número de tuplas (instancias o filas) afectadas por consultas INSERT, UPDATE y DELETE. Si no hay ninguna tupla afectada la función devolverá 0.

Ejemplo 1. pg_cmdtuples

```
<?php
$result = pg_exec($conn, "INSERT INTO verlag VALUES ('Autor')");
$cmtuples = pg_cmdtuples($result);
echo $cmtuples . " <- cmdtuples affected." ;
?>
```

pg_Connect (PHP 3, PHP 4)

Abre una conexión

```
int pg_connect (string host, string port, string options, string tty, string dbname)
```

Devuelve un índice de conexión en caso de éxito, o falso si la conexión no se puede realizar. Esta función abre una conexión a una base de datos PostgreSQL. Cada uno de los argumentos debe ser una cadena entrecomillada, incluyendo el número de puerto. Los parámetros options y tty son opcionales y pueden ser omitidos. Esta función devuelve un índice de conexión que se necesitará para otras funciones PostgreSQL. Puedes tener multiples conexiones abiertas al mismo tiempo.

Una conexión también se puede establecer con el siguiente comando: **\$conn = pg_connect("dbname=marliese port=5432");** Otros parámetros aparte de *dbname* y *port* son *host*, *tty*, *options*, *user* y *password*.

Ver también **pg_pConnect()**.

pg_Dbname (PHP 3, PHP 4)

Nombre de la base de datos

```
string pg_dbname (int connection)
```

Devuelve el nombre de la base de datos a la cual es el índice de conexión con PostgreSQL está conectado, o false si connection no es un índice de conexión válido.

pg_ErrorMessage (PHP 3, PHP 4)

mensaje de error

```
string pg_errormessage ( int connection )
```

Devuelve una cadena que contiene el mensaje de error, o false en caso de fallo. Probablemente no se podrán obtener los detalles del error a través de la función **pg_errormessage()** si ocurre un error en la última acción de base de datos para la cual existe una conexión válida, esta función retornará una cadena conteniendo el mensaje de error generado por el servidor "backend".

pg_Exec (PHP 3, PHP 4)

Ejecuta una consulta (query)

```
int pg_exec ( int connection, string query )
```

Devuelve un índice de resultado si se pudo ejecutar la consulta, o false en caso de fallo o si connection no es un índice de conexión válido. Se pueden obtener detalles acerca del error mediante la función **pg_ErrorMessage()** siempre que connection sea válido. Envía una sentencia SQL a la base de datos PostgreSQL especificada por el índice de conexión. connection debe ser un índice válido devuelto por **pg_Connect()**. El valor de devuelto por esta función es un índice para ser usado al acceder a los resultados de la consulta desde otras funciones PostgreSQL.

Nota: PHP/FI devolvía 1 si no es una consulta que tenga que devolver datos (inserts o updates, por ejemplo) y un valor mayor que 1 incluso en el caso de selects que no devolvieron nada. En PHP no se puede contar con ninguna de esas suposiciones.

pg_Fetch_Array (PHP 3>= 3.0.1, PHP 4)

obtiene una fila en la forma de un array

```
array pg_fetch_array ( int result, int row [, int result_type] )
```

Devuelve: Un array que se corresponde con la fila obtenida, o false si no hay más filas.

pg_fetch_array() es una versión extendida de **pg_fetch_row()**. Además de almacenar los datos en los índices numéricos del array resultante, también almacena los datos usando índices asociativos, empleando para ello el nombre del campo como la llave o índice.

El tercer parámetro opcional *result_type* en **pg_fetch_array()** es una constante y puede tomar cualquiera de los siguientes valores: PGSQL_ASSOC, PGSQL_NUM, y PGSQL_BOTH.

Nota: *Result_type* se añadio en PHP 4.0.

Una cosa importante a tener en cuenta es que usar **pg_fetch_array()** NO es significativamente más lento que usar **pg_fetch_row()**, y sin embargo el valor añadido que aporta sí lo es.

Para más detalles, ver **pg_fetch_row()**

Ejemplo 1. PostgreSQL fetch array

```
<?php
$conn = pg_pconnect("", "", "", "", "publisher");
if (!$conn) {
    echo "An error occurred.\n";
    exit;
}

$result = pg_Exec ($conn, "SELECT * FROM authors");
if (!$result) {
    echo "An error occurred.\n";
    exit;
}

$arr = pg_fetch_array ($result, 0);
echo $arr[0] . " <- array\n";

$arr = pg_fetch_array ($result, 1);
echo $arr["author"] . " <- array\n";
?>
```

pg_Fetch_Object (PHP 3>= 3.0.1, PHP 4)

obtener una fila en forma de objeto

```
object pg_fetch_object ( int result, int row [, int result_type] )
```

Devuelve: Un objeto cuyas propiedades se corresponden con los campos de la fila obtenida, o false si no hay más filas.

pg_fetch_object() es parecida a **pg_fetch_array()**, con una diferencia - se devuelve un objeto, en vez de un array.

Indirectamente, eso significa que solo puedes acceder a los datos por medio de su nombre de campo, y no a través de sus posiciones (los números son nombres de propiedad invalidos).

El tercer parámetro opcional *result_type* en **pg_fetch_object()** es una constante y puede tomar cualquiera de los siguientes valores: PGSQL_ASSOC, PGSQL_NUM, y PGSQL_BOTH.

Nota: *Result_type* se añadio en PHP 4.0.

Referente a la velocidad, la función es identica a **pg_fetch_array()**, y practicamente tan rápida como **pg_fetch_row()** (la diferencia es insignificante).

Ver también: **pg_fetch_array()** y **pg_fetch_row()**.

Ejemplo 1. Postgres fetch object

```
<?php
$database = "verlag";
$db_conn = pg_connect ("localhost", "5432", "", "", $database);
```

```

if (!$db_conn): ?>
    <H1>Failed connecting to postgres database <? echo $database ?></H1> <?
    exit;
endif;

$qu = pg_exec ($db_conn, "SELECT * FROM verlag ORDER BY autor");
$row = 0; // postgres needs a row counter other dbs might not

while ($data = pg_fetch_object ($qu, $row)):
    echo $data->autor." (" ;
    echo $data->jahr .") : ";
    echo $data->titel."<BR>";
    $row++;
endwhile; ?>

<PRE><?php
$fields[] = Array ("autor", "Author");
$fields[] = Array ("jahr", " Year");
$fields[] = Array ("titel", " Title");

$row= 0; // postgres needs a row counter other dbs might not
while ($data = pg_fetch_object ($qu, $row)):
    echo "-----\n";
    reset ($fields);
    while (list (,$item) = each ($fields)):
        echo $item[1].": ".$data->$item[0]."\n";
    endwhile;
    $row++;
endwhile;
echo "-----\n"; ?>
</PRE> <?php
pg_freeResult ($qu);
pg_close ($db_conn);
?>
```

pg_Fetch_Row (PHP 3>= 3.0.1, PHP 4)

obtiene la fila como un array enumerado

```
array pg_fetch_row (int result, int row)
```

Devuelve: Un array que se corresponde con la fila obtenida, o false en el caso de que no haya más filas.

pg_fetch_row() obtiene una fila de datos a partir del resultado asociado con el identificador de resultado especificado. La fila se devuelve en forma de array. Cada columna del resultado se almacena en una posición del array, empezando a partir de la posición 0.

Las siguientes llamadas a **pg_fetch_row()** devolverán la fila siguiente en el conjunto resultado, o falso en el caso de que no haya más filas que devolver.

Ver también: **pg_fetch_array()**, **pg_fetch_object()**, **pg_result()**.

Ejemplo 1. Postgres fetch row

```
<?php
$conn = pg_pconnect("", "", "", "", "publisher");
if (!$conn) {
```

```

        echo "An error occurred.\n";
        exit;
    }

$result = pg_Exec ($conn, "SELECT * FROM authors");
if (!$result) {
    echo "An error occurred.\n";
    exit;
}

$row = pg_fetch_row ($result, 0);
echo $row[0] . " <- row\n";

$row = pg_fetch_row ($result, 1);
echo $row[0] . " <- row\n";

$row = pg_fetch_row ($result, 2);
echo $row[1] . " <- row\n";
?>

```

pg_FieldIsNull (PHP 3, PHP 4)

Comprueba si un campo es NULO

```
int pg_fieldisnull (int result_id, int row, mixed field)
```

Comprueba si un campo vale NULL o no. Devuelve 0 si el campo en la fila dada no es NULO y uno en el caso de que lo sea. El campo se puede especificar mediante un número o un nombre de campo. La numeración de filas empieza en 0.

pg_FieldName (PHP 3, PHP 4)

Devuelve el nombre de un campo

```
string pg_fieldname (int result_id, int field_number)
```

pg_FieldName() devolverá el nombre del campo que ocupa el número de columna dado en el identificador de resultado de PostgreSQL. La numeración de los campos empieza con 0.

pg_FieldNum (PHP 3, PHP 4)

Devuelve el número de una columna

```
int pg_fieldnum (int result_id, string field_name)
```

pg_FieldNum() devolverá el número de la columna que corresponde al campo cuyo nombre le damos, dentro del identificador de resultado de PostgreSQL. La numeración de campos comienza en 0. Esta función devolverá -1 en caso de error.

pg_FieldPrtLen (PHP 3, PHP 4)

Devuelve la longitud impresa

```
int pg_fieldprtlens (int result_id, int row_number, string field_name)
```

pg_FieldPrtLen() devolverá la longitud impresa real (número de caracteres) de un valor específico dentro del identificador de resultado PostgreSQL. La numeración de filas comienza en 0. Esta función devolverá -1 en caso de error.

pg_FieldSize (PHP 3, PHP 4)

Devuelve el tamaño de almacenamiento interno de un campo en concreto

```
int pg_fieldsize (int result_id, int field_number)
```

pg_FieldSize() devolverá el tamaño de almacenamiento interno (en bytes) de uno de los campos del resultado PostgreSQL que le hemos pasado. La numeración de campos empieza en 0. Un tamaño de campo de -1 indica que se trata de un campo de longitud variable. La función devolverá false en caso de error.

pg_FieldType (PHP 3, PHP 4)

Devuelve el nombre del tipo de dato correspondiente al campo cuyo número pasamos como parámetro

```
int pg_fieldtype (int result_id, int field_number)
```

pg_FieldType() devolverá una cadena con el nombre del tipo de datos de un campo dado dentro del identificador de resultado PostgreSQL result_id. La numeración de campos empieza en 0.

pg_FreeResult (PHP 3, PHP 4)

Libera memoria

```
int pg_freeresult (int result_id)
```

pg_FreeResult() solo necesita ser llamada si estamos preocupados por usar demasiada memoria mientras el script se está ejecutando. La memoria correspondiente a todos los resultados de consulta se libera automáticamente cuando termina el script. Pero, si estás seguro de que no vas a necesitar más los datos del resultado en el script, puedes llamar a **pg_FreeResult()** con el identificador del resultado como parámetro y la memoria asociada al resultado será liberada.

pg_GetLastOid (PHP 3, PHP 4)

Devuelve el identificador del último objeto insertado

```
int pg_getlastoid (int result_id)
```

pg_GetLastOid() se puede usar para conseguir el Oid (identificador de objeto) asignado a una tupla insertada si el identificador de resultado proviene de una llamada a **pg_Exec()** que fuese un INSERT SQL. Esta función devuelve un entero positivo si hay un Oid válido y -1 en caso de que ocurriese un error durante el último comando enviado a través de la función **pg_Exec()** o si esta no fuese un INSERT.

pg_Host (PHP 3, PHP 4)

Devuelve el nombre del host

```
string pg_host (int connection_id)
```

pg_Host() devuelve el nombre del host al que identificador conexión PostgreSQL pasado está conectado.

pg_loclose (PHP 3, PHP 4)

Cierra un objeto grande (large object)

```
void pg_loclose (int fd)
```

pg_loclose() cierra un Large Object. *fd* es el descriptor de fichero del fichero grande obtenido a través de **pg_loopen()**.

pg_locreate (PHP 3, PHP 4)

Crea un objeto grande

```
int pg_locreate (int conn)
```

pg_locreate() Crea un Large Object y devuelve su oid. *conn* determina una conexión de base de datos válida. Los modos de acceso INV_READ, INV_WRITE, y INV_ARCHIVE de PostgreSQL no están soportados, el objeto se crea siempre con acceso tanto de lectura como de escritura. modo El INV_ARCHIVE ha desaparecido incluso de PostgreSQL mismo (a partir de la versión 6.3).

pg_loopen (PHP 3, PHP 4)

Abre un objeto grande

```
int pg_loopen (int conn, int objoid, string mode)
```

pg_loopen() abre un Large Object (objeto grande) y devuelve un descriptor de fichero para el objeto grande. El descriptor de fichero encapsula información acerca de la conexión. No se debe cerrar la conexión antes de cerrar el descriptor de fichero al objeto grande. *objoid* especifica un oid válido para un objeto grande y *mode* puede ser "r", "w", o "rw".

pg_loread (PHP 3, PHP 4)

lee un large object (objeto grande)

```
string pg_loread (int fd, int len)
```

pg_loread() lee como mucho *len* bytes a partir de un objeto grande y lo devuelve como una cadena. *fd* especifica un descriptor de fichero de objeto grande válido y *len* especifica máximo número de bytes que se deben leer del objeto grande.

pg_loreadall (PHP 3, PHP 4)

Lee un objeto grande entero

```
void pg_loreadall (int fd)
```

pg_loreadall() lee un objeto grande y lo pasa tal cual al browser después de enviar todas las cabeceras pendientes. Principalmente dirigido a mandar datos binarios como imágenes o sonido.

pg_lounlink (PHP 3, PHP 4)

borra un large object

```
void pg_lounlink (int conn, int lobjid)
```

pg_lounlink() borra el objeto grande con identificador *lobjid*.

pg_lowrite (PHP 3, PHP 4)

escribe en un objeto grande

```
int pg_lowrite (int fd, string buf)
```

pg_lowrite() escribe todo lo que puede en un objeto grande a partir de la variable *buf* y devuelve el número de bytes realmente escritos, o falso si ocurre algún error. *fd* es un descriptor de fichero para el objeto grande obtenido a través de **pg_loopen()**.

pg_NumFields (PHP 3, PHP 4)

Devuelve el número de campos

```
int pg_numfields (int result_id)
```

pg_NumFields() devuelve el número de campos (columnas) en un resultado PostgreSQL. El parámetro es un identificador de resultado válido devuelto por **pg_Exec()**. La función devuelve -1 en caso de error.

pg_NumRows (PHP 3, PHP 4)

Devuelve el número de filas

```
int pg_numrows (int result_id)
```

pg_NumRows() devuelve el número de filas en un resultado PostgreSQL. El parámetro es un identificador de resultado PostgreSQL válido devuelto por **pg_Exec()**. En caso de error se devuelve -1.

pg_Options (PHP 3, PHP 4)

Devuelve opciones

```
string pg_options (int connection_id)
```

pg_Options() devuelve una cadena que contiene las opciones especificadas en el identificador de conexión con PostgreSQL dado.

pg_pConnect (PHP 3, PHP 4)

Crea una conexión persistente con una base de datos

```
int pg_pconnect (string host, string port, string options, string tty, string dbname)
```

Devuelve un índice de conexión en caso de éxito, o false si no es posible realizar la conexión. Abre una conexión persistente hacia una base de datos de PostgreSQL. Cada uno de los parámetros puede ser una cadena entrecomillada (quoted), incluyendo el número de puerto. Los parámetros options y tty son opcionales y pueden omitirse. Esta función devuelve un índice de conexión que luego será empleado al llamar a otras funciones PostgreSQL. Puedes tener multiples conexiones persistentes abiertas al mismo tiempo. Ver también **pg_Connect()**.

Una conexión también se puede establecer con el comando siguiente: `$conn = pg_pconnect('dbname=marliese port=5432');` Otros parámetros además de `dbname` y `port` son `host`, `tty`, `options`, `user` y `password`.

pg_Port (PHP 3, PHP 4)

Devuelve el número de puerto

```
int pg_port (int connection_id)
```

pg_Port() devuelve el número del puerto al que el identificador de conexión con PostgreSQL está conectado.

pg_Result (PHP 3, PHP 4)

Devuelve valores a partir de un identificador de resultado

```
mixed pg_result (int result_id, int row_number, mixed fieldname)
```

pg_Result() devuelve valores a partir de un identificador de resultado generado en la función **pg_Exec()**. Los parámetros *row_number* y *fieldname* especifican que celda en la tabla queremos obtener. La numeración de filas comienza en 0. En vez de usar el nombre del campo también puedes usar el índice del campo como un número sin entrecomillar. Los índices de campo comienzan también en 0.

PostgreSQL tiene muchos tipos y solo los básicos están soportados directamente aquí. Todas las formas de enteros, booleanos y oids se devuelven como valores enteros. Todas las formas de los tipos float y real se devuelven como valores double. Todos los demás tipos, incluyendo los arrays se devuelven como cadenas formateadas de la misma manera en que PostgreSQL usa por defecto. De la misma forma en que lo verías en el programa **psql**.

pg_tty (PHP 3, PHP 4)

Devuelve el nombre del tty

```
string pg_tty (int connection_id)
```

pg_tty() devuelve el nombre del tty hacia el que se dirige la salida de depuración del lado del servidor en el identificador de conexión de PostgreSQL dado.

LIX. Funciones de ejecución de programas

escapeshellcmd (PHP 3, PHP 4)

enmascara los metacaracteres del intérprete de ordenes

```
string escapeshellcmd (string command)
```

EscapeShellCmd() enmascara cualquier carácter en una cadena de caracteres que pueda usarse para introducir fraudulentamente una orden al intérprete de órdenes para que éste ejecute instrucciones arbitrarias. Esta función se debería usar para asegurarse que cualquier dato que venga del usuario se enmascare antes de que éste se le pase a las funciones **exec()** o **system()**, o al operador ‘(apóstrofe invertido)’. Un uso habitual podría ser:

```
system(EscapeShellCmd($cmd))
```

Véase también **exec()**, **popen()**, **system()**, y el operador ‘(apóstrofe invertido).

exec (PHP 3, PHP 4)

Ejecuta un programa externo

```
string exec (string command [, string array [, int return_var]])
```

exec() ejecuta la orden indicada en *command*, sin embargo no produce ninguna salida. Simplemente devuelve la última línea de la salida resultado de la orden. Si necesita ejecutar una orden y obtener directamente todos los datos devueltos por la orden sin ninguna interferencia, use la función **PassThru()**.

Si el parámetro *array* existe, entonces el array especificado se llenará con cada una de las líneas de la salida producida por la orden. Notar que si el array ya contiene algunos elementos, **exec()** los añadirá al final del array. Si no quiere que la función añada dichos elementos, haga un **unset()** sobre el array antes de pasárselo a **exec()**.

Si el parámetro *return_var* existe a la vez que el parámetro *array*, entonces el valor de retorno de la orden ejecutada se guardará en dicha variable.

Destacar que si usted va a permitir que se pasen datos provenientes de usuarios a esta función, entonces debería usar **EscapeShellCmd()** para asegurarse de que los usuarios no pueden engañar al sistema para ejecutar instrucciones arbitrarias.

Véase también **system()**, **PassThru()**, **popen()**, **EscapeShellCmd()**, y el operador ‘(apóstrofe invertido).

passthru (PHP 3, PHP 4)

Ejecuta un programa externo y muestra su salida literal

```
string passthru (string command [, int return_var])
```

La función **passthru()** es similar a la función **Exec()** en que ejecuta una orden (*command*). Si existe el parámetro *return_var*, el valor de estado devuelto por la orden Unix se guardará ahí. Esta función debería usarse en lugar de **Exec()** o **System()** cuando la salida de la orden Unix sean datos binarios que deban ser pasados directamente al navegador. Un uso típico de ello es ejecutar algo como las utilidades pbmplus las cuales pueden dar como resultado directamente el flujo de datos de una imagen. Poniendo el content-type a *image/gif* y llamando al programa pbmplus para mostrar un gif, usted puede crear archivos de órdenes PHP que generen directamente imágenes.

Véase también **exec()**, **system()**, **popen()**, **EscapeShellCmd()**, y el [operador ‘ \(apóstrofe invertido\)](#).

system (PHP 3, PHP 4)

Ejecuta un programa externo y muestra su salida

```
string system (string command [, int return_var])
```

System() se parece a la versión C de la función de mismo nombre en que ejecuta la orden indicada en *command* y muestra el resultado. Si se indica una variable como segundo parámetro, el código de estado devuelto por la orden ejecutada se guardará en esta variable.

Destacar que si usted va a permitir que se pasen datos provenientes de usuarios a esta función, entonces debería usar **EscapeShellCmd()** para asegurarse de que los usuarios no pueden engañar al sistema para ejecutar instrucciones arbitrarias.

La llamada a **System()** también intenta vaciar automáticamente el buffer de salida del servidor web después de cada línea de salida si PHP está funcionando como un módulo del servidor.

Devuelve la última línea de la orden en caso de éxito, y falso en caso de fallo.

Si necesita ejecutar una orden y obtener de vuelta todo los datos del mismo sin interferencias, use la función **PassThru()**.

Véase también **exec()**, **PassThru()**, **popen()**, **EscapeShellCmd()**, y el [operador ‘ \(apóstrofe invertido\)](#).

LX. Pspell Functions

The **pspell()** functions allow you to check the spelling of a word and offer suggestions.

You need the aspell and pspell libraries, available from <http://aspell.sourceforge.net/> and <http://pspell.sourceforge.net/> respectively, and add the `-with-pspell[=dir]` option when compiling php.

pspell_new (PHP 4 >= 4.0.2)

Load a new dictionary

```
int pspell_new (string language [, string spelling [, string jargon [, string encoding [, int mode]]]])
```

Pspell_new() opens up a new dictionary and returns the dictionary link identifier for use in other pspell functions.

The language parameter is the language code which consists of the two letter ISO 639 language code and an optional two letter ISO 3166 country code after a dash or underscore.

The spelling parameter is the requested spelling for languages with more than one spelling such as English. Known values are 'american', 'british', and 'canadian'.

The jargon parameter contains extra information to distinguish two different words lists that have the same language and spelling parameters.

The encoding parameter is the encoding that words are expected to be in. Valid values are 'utf-8', 'iso8859-*', 'koi8-r', 'viscii', 'cp1252', 'machine unsigned 16', 'machine unsigned 32'. This parameter is largely untested, so be careful when using.

The mode parameter is the mode in which spellchecker will work. There are several modes available:

- PSPELL_FAST - Fast mode (least number of suggestions)
- PSPELL_NORMAL - Normal mode (more suggestions)
- PSPELL_BAD_SPELLERS - Slow mode (a lot of suggestions)
- PSPELL_RUN_TOGETHER - Consider run-together words as legal compounds. That is, "thecat" will be a legal compound, although there should be a space between the two words. Changing this setting only affects the results returned by **pspell_check()**; **pspell_suggest()** will still return suggestions.

Mode is a bitmask constructed from different constants listed above. However, PSPELL_FAST, PSPELL_NORMAL and PSPELL_BAD_SPELLERS are mutually exclusive, so you should select only one of them.

For more information and examples, check out inline manual pspell website:<http://pspell.sourceforge.net/>.

Ejemplo 1. Pspell_new()

```
$pspell_link = pspell_new ("en", "", "", "",  
                           (PSPELL_FAST|PSPELL_RUN_TOGETHER));
```

pspell_check (PHP 4 >= 4.0.2)

Check a word

```
boolean pspell_check (int dictionary_link, string word)
```

Pspell_check() checks the spelling of a word and returns true if the spelling is correct, false if not.

Ejemplo 1. Pspell_check()

```
$pspell_link = pspell_new ("en");
```

```
if (pspell_check ($pspell_link, "testt")) {
    echo "This is a valid spelling";
} else {
    echo "Sorry, wrong spelling";
}
```

pspell_suggest (PHP 4 >= 4.0.2)

Suggest spellings of a word

```
array pspell_suggest (int dictionary_link, string word)
```

Pspell_suggest() returns an array of possible spellings for the given word.

Ejemplo 1. Pspell_suggest()

```
$pspell_link = pspell_new ("en");

if (!pspell_check ($pspell_link, "testt")) {
    $suggestions = pspell_suggest ($pspell_link, "testt");

    for ($i=0; $i < count ($suggestions); $i++) {
        echo "Possible spelling: " . $suggestions[$i] . "<br>";
    }
}
```

LXI. GNU Readline

Las funciones **readline()** implementan una interfaz con la librería GNU Readline. Un ejemplo de la manera de funcionar podría ser la forma en que el Bash permite usar las flechas de dirección para insertar caracteres o desplazarse a través del historial de comandos. Debido a la naturaleza interactiva de esta librería, tendrá un uso muy reducido en la escritura de aplicaciones Web, aunque puede ser útil para scripts que han de ser ejecutados desde la consola.

La página principal del proyecto GNU Readline es <http://cnswww.cns.cwru.edu/~chet/readline/rlist.html>. Está actualizada por Chet Ramey, quien además es el autor de Bash.

readline (PHP 4 >= 4.0b4)

Lee una línea

```
string readline ([string prompt])
```

Esta función devuelve una única cadena del usuario. Puede especificar una cadena que se mostrará al usuario. La línea devuelta tiene el indicador final de nueva línea eliminado. Necesita añadir esta línea al historial usando la función **readline_add_history()**.

Ejemplo 1. Readline()

```
//obtiene 3 comandos del usuario
for ($i=0; $i < 3; $i++) {
    $line = readline ("Comando: ");
    readline_add_history ($line);
}

//Vuelca el historial
print_r (readline_list_history());

//Vuelca las variables
print_r (readline_info());
```

readline_add_history (PHP 4 >= 4.0b4)

Añade una línea al historial

```
void readline_add_history (string line)
```

Esta función añade una línea al historial de líneas de comandos.

readline_clear_history (PHP 4 >= 4.0b4)

Borra el historial

```
boolean readline_clear_history (void )
```

Esta función borra por completo el historial de la línea de comandos.

readline_completion_function (PHP 4 >= 4.0b4)

Registra una función de completitud

```
boolean readline_completion_function (string line)
```

Esta función registra una función de completitud. Debe proporcionar el nombre de una función existente que acepte una línea de comandos parcial y devuelva un array con posibles coincidencias. Es el mismo tipo de funcionalidad que se obtiene al pulsar la tecla de tabulación cuando se está usando el Bash.

readline_info (PHP 4 >= 4.0b4)

Establece/Obtiene diversas variables internas de readline

```
mixed readline_info ([string varname [, string newvalue]])
```

Si es llamada sin parámetros, esta función devuelve un array con los valores de todas las opciones que readline usa. Los elementos vendrán indexados por los siguientes valores: done, end, erase_empty_line, library_version, line_buffer, mark, pending_input, point, prompt, readline_name, y terminal_name.

Si es llamada con un parámetro, devuelve el valor de esa opción. Si es llamada con dos parámetros, el valor de la opción será cambiado al parámetro dado.

readline_list_history (PHP 4 >= 4.0b4)

Lista el historial

```
array readline_list_history (void )
```

Esta función devuelve un array con el historial de líneas de comandos completo. Los elementos están indexados por enteros comenzando por el cero.

readline_read_history (PHP 4 >= 4.0b4)

Lee un historial

```
boolean readline_read_history (string filename)
```

Esta función lee un historial de comandos desde un fichero.

readline_write_history (PHP 4 >= 4.0b4)

Escribe el historial

```
boolean readline_write_history (string filename)
```

Esta función escribe el historial de comandos en un archivo.

LXII. Funciones GNU Recode

Este modulo contiene un interfaz para la biblioteca GNU Recode version 3.5. Para poder usar las funciones definidas en este modulo, debereis de compilar el interprete PHP con la opcion –with-recode. Para poder hacer esto debereis tener instalado en vuestro sistema GNU Recode 3.5 o superior.

La biblioteca GNU Recode convierte entre ficheros con diferentes codigos de caracteres y codificacion. Cuando esto no puede realizarse exactamente, puede desahacerse de los caracteres problematicos o crear una aproximacion. La biblioteca reconoce o produce alrededor de 150 codigos de caracteres y puede convertir ficheros entre casi todos los pares posibles. La gran mayoria de los codigos de caracteres RFC 1345 estan soportados.

recode_string (PHP 3>= 3.0.13, PHP 4 >= 4.0RC1)

Recodifica una cadena literal segun una peticion de recodificacion.

```
string recode_string (string request, string string)
```

Recodifica la cadena *string* segun una peticion de recodificacion *request*. Devuelve FALSE si no puede realizar la recodificacion, TRUE si todo va bien.

Una simple peticion "recode" podria ser "lat1..iso646-de". Ver tambien la documentacion de GNU Recode de tu instalacion para obtener instrucciones detalladas sobre peticiones de recodificacion.

recode_file (PHP 3>= 3.0.13, PHP 4 >= 4.0RC1)

Recodifica de fichero a fichero segun una peticion de recodificacion.

```
bool recode_file (int input, int output)
```

Recodifica el fichero definido por *input* a el fichero definido por *output*, segun la peticion de recodificacion *request*. Devuelve FALSE si no puede realizar la recodificacion, TRUE si todo va bien.

Esta funcion no procesa ficheros remotos (URLs). Los dos ficheros deben de ser locales en el sistema.

LXIII. Funciones de expresiones regulares compatibles con Perl

La sintaxis, para los patrones usados en estas funciones, es muy semejante al Perl. Las expresiones estarán encerradas por delimitadores, por ejemplo una barra de dividir (/). Cualquier carácter puede ser usado para delimitar incluso los que no son caracteres alfanuméricos o la barra invertida (\). Si el carácter delimitador ha sido usado en la propia expresión, es necesario que sea precedido por una barra inversa.

El delimitador de fin puede ser seguido por varios modificadores que afectarán al resultado. Examina [Modificadores de Patrones](#).

Ejemplo 1. Ejemplos de patrones válidos

- /<|\w+>/
- |(\d{3})-\d+|Sm
- /^(?i)php[34]/

Ejemplo 2. Ejemplos de patrones no válidos

- /href='.*' - falta el delimitador de fin
- \w+\s*\w+/J - el modificador 'J' es desconocido
- 1-\d3-\d3-\d4| - falta el delimitador de inicio

Nota: Para las funciones de expresiones compatibles con Perl se necesita PHP 4 o PHP 3.0.9 o superior.

preg_match (PHP 3>= 3.0.9, PHP 4)

Realiza un emparejamiento dada una expresión

```
int preg_match (string pattern, string subject [, array matches])
```

Busca en *subject* para un emparejamiento, dada la expresión *pattern*.

Si *matches* es dado, entonces será definido con el resultado de la búsqueda. \$matches[0] contendrá el texto que empareja con el patrón en su totalidad. \$matches[1] tendrá la cadena que empareje con el primer subpatrón que esté entre paréntesis y así sucesivamente.

Devuelve true si se encontró en la cadena un emparejamiento dado el patrón *pattern*, false si no se produjo o hubo un error.

Ejemplo 1. Obtener el número de la siguiente página dada una cadena

```
if (preg_match("/page\s+\#(\d+)/i", "Go to page #9.", $parts))
    print "Next page is $parts[1]";                                // La siguiente página es $parts[1]
else
    print "Page not found. ";                                     // Página no encontrada
```

Examinar también **preg_match_all()**, **preg_replace()**, y **preg_split()**.

preg_match_all (PHP 3>= 3.0.9, PHP 4)

Realiza un completo emparejamiento de expresiones

```
int preg_match_all (string pattern, string subject, array matches [, int order])
```

Busca en *subject* todos los emparejamientos de la expresión *pattern* y los pone en *matches* de la forma indicada por *order*.

Después de encontrar el primer emparejamiento, las subsiguientes búsquedas empiezan desde el punto del último casamiento.

order puede tener los siguientes valores:

PREG_PATTERN_ORDER

Los resultados serán devueltos de manera que \$matches[0] es un array con el patrón de búsqueda completo, \$matches[1] es una array de las cadenas casadas por el primer subpatrón que esté entre paréntesis y así sucesivamente.

```
preg_match_all("|[>]+(.*)<[>]+|U", "<b>example: </b><div align=left>this is a test</div>",
print $out[0][0].", ".$out[0][1]."\n";
print $out[1][0].", ".$out[1][1]."\n"
```

Esta ejemplo dará como resultado:

```
<b>example: </b>, <div align=left>this is a test</div>
example: , this is a test
```

Así, \$out[0] contiene el array con las cadena que casan completamente con el patrón y \$out[1] con las cadenas que se encuentran entre los tags.

PREG_SET_ORDER

Los resultados son dados de manera que \$matches[0] es una array del primer conjunto de emparejamientos, \$matches[1] es un array de los segundos conjuntos de casamientos y así sucesivamente.

```
preg_match_all(" |<[^>]+>(.*)</[^>]+>|U", "example: <b><div align=left>this is a test</div></b>, print $out[0][0].", ".$out[0][1]."\n";
print $out[1][0].", ".$out[1][1]."\n"
```

Este ejemplo dará como resultado:

```
<b>example: </b>, example:
<div align=left>this is a test</div>, this is a test
```

En este caso, \$matches[0] es el primer conjunto de emparejamientos y \$matches[0][0] tiene el casamiento completo, \$matches[0][1] el del primer subpatrón y así sucesivamente. Similarmente, \$matches[1] es el segundo conjunto de emparejamientos, etc.

Si *order* no es dado, se asume PREG_PATTERN_ORDER.

Devuelve el número de casamientos completos, false si no hubo o se produjo error.

Ejemplo 1. Obtener los número de teléfonos de un texto.

```
preg_match_all("/\(? (\d{3})? \)? (?(1) [\-\s] ) \d{3}-\d{4}/x",
    "Call 555-1212 or 1-800-555-1212", $phones);
```

Examina también **preg_match()**, **preg_replace()** y **preg_split()**.

preg_replace (PHP 3>= 3.0.9, PHP 4)

Lleva a cabo la búsqueda de una expresión y su sustitución

```
mixed preg_replace (mixed pattern, mixed replacement, mixed subject)
```

Busca en *subject* los emparejamientos con *pattern* y los sustituye por *replacement*.

replacement puede contener referencias de la forma `\n`. Éstas serán sustituidas por el texto obtenido por el patrón del paréntesis *n*ésimo. *n* puede tener un valor de cero a noventa y nueve, y `\0` se refiere al texto casado por el patrón completo. Para obtener el número del subpatrón de búsqueda, los paréntesis abiertos son contados de izquierda derecha tomando el primero como uno.

Si el patrón no es encontrado en *subject*, entonces no se realizarán cambios.

Todos los parámetros de la función **preg_replace()** pueden ser un array.

Si *subject* es un array, entonces la búsqueda y sustitución es realizada para todos los elementos de *subject*, y el valor devuelto es también un array.

Si *pattern* y *replacement* son arrays, entonces **preg_replace()** toma un valor desde cada array y los usas para buscar y sustituir sobre *subject*. Si *replacement* tiene menos valores que *pattern*, entonces la cadena vacía es usada como valor para el resto de sustituciones. Si *pattern* es una array y *replacement* es una cadena, entonces esta cadena de sustitución es usada para todos los valores de *pattern*. Sin embargo, lo contrario no tiene sentido.

El modificador `/e` hace que la función **preg_replace()** trate el parámetro *replacement* como código PHP después de que la apropiada sustitución sea hecha. Atención, asegúrate que *replacement* es un código PHP correcto, de otro modo PHP dará un error de parse en la línea que contenga **preg_replace()**.

Nota: Este modificador fue añadido en PHP 4.0.

Ejemplo 1. Sustituir varios valores

```
$patterns = array("/(19|20\d{2})-(\d{1,2})-(\d{1,2})/", "/^\s*\{(\w+)\}\s*/");
$replace = array("\3/\4/\1", "$\1=");
print preg_replace($patterns, $replace, "{startDate} = 1999-5-27");
```

Esta ejemplo dará como resultado:

```
$startDate = 5/27/1999
```

Ejemplo 2. Usar el modificador /e

```
preg_replace("/(<\/?)(\w+)([^>]*>)/e", "'\\1'.strtoupper('\\2').'\\3'", $html_body);
```

Pondrá en mayúscula todos los tags HTML del texto de entrada.

Examina también **preg_match()**, **preg_match_all()**, y **preg_split()**.

preg_split (PHP 3>= 3.0.9, PHP 4)

Divide una cadena dada una expresión

```
array preg_split (string pattern, string subject [, int limit [, int flags]])
```

Nota: El parámetro *flags* fue añadido en la Beta 3 de PHP

Devuelve un array contenido las subcadenas de *subject* divididas mediante los emparejamientos limitados por *pattern*.

Si *limit* es proporcionado, entonces sólo *limit* subcadenas son devueltas.

Si el flags es PREG_SPLIT_NO_EMPTY entonces las cadenas vacías no serán devueltas por **preg_split()**.

Ejemplo 1. Obtener las partes de una cadena de búsqueda

```
$keywords = preg_split("/[\s,]+/", "hypertext language, programming");
```

Examinar también **preg_match()**, **preg_match_all()**, y **preg_replace()**.

preg_quote (PHP 3>= 3.0.9, PHP 4)

Prepara los caracteres de expresiones

```
string preg_quote (string str)
```

preg_quote() toma *str* y pone una barra invertida (\) delante de todo carácter que sea parte de la sintaxis de las expresiones. Es útil si tienes una cadena en tiempo de ejecución y puede contener caracteres especiales.

Los caracteres especiales de las expresiones son:

```
. \ \ + * ? [ ^ ] $ ( ) { } = ! < > | :
```

Nota: Esta función fue añadida en PHP 3.0.9.

preg_grep (PHP 4)

Devuelve un array con los elementos que casen con el patrón

```
array preg_grep (string pattern, array input)
```

preg_grep() devuelve un array conteniendo los elementos del array *input* que emparejen con el patrón (*pattern*) dado.

Ejemplo 1. Ejemplo de la función preg_grep()

```
preg_grep("/^(\d+)?\.\d+$/", $array); // encuentra todos los números reales en el array
```

Nota: Esta función fue añadida en PHP 4.0.

Modificadores de Patrones (unknown)

describe los modificadores posibles en los patrones de expresiones regulares (regex)

Los posibles modificadores PCRE (Funciones de Expresiones Compatibles con Perl), en este momento, son mostrados a continuación. Los nombres entre paréntesis se refieren a nombres internos PCRE para dichos modificadores.

i (PCRE_CASELESS)

Si es usado, no se distinguirá entre mayúsculas y minúsculas.

m (PCRE_MULTILINE)

Por defecto, PCRE trata la cadena de entrada como si fuera una sola línea de caracteres (aun cuando tenga varias). El carácter especial de "inicio de línea" (^) empareja sólo al principio de la cadena, mientras el carácter especial de "fin de línea" (\$) casa sólo el fin de la entrada, o antes un carácter de nueva línea (a menos que el modificador *E* sea definido). Esto es lo mismo que en Perl.

Cuando este modificador es utilizado, los constructores de "inicio de línea" y "fin de línea" son emparejados con el carácter de nueva línea. Esto es equivalente al modificador /m del Perl. Si no hay caracteres "\n" en la cadena de entrada, o no existen ^ o \$ en el patrón, entonces este modificador no alterará el resultado.

s (PCRE_DOTALL)

Si se usa, el carácter especial de un punto en el patrón emparejará todos los caracteres, incluyendo el de nueva línea. Sin él, el carácter de nueva línea es excluido. Este modificador equivale a /s en Perl. Una cláusula como [^a] siempre casa con un carácter de nueva línea, independientemente de la utilización de este modificador.

x (PCRE_EXTENDED)

Si es definido, los caracteres de información con espacios en blanco en el patrón son ignorados excepto cuando son precedidos por una barra invertida o dentro de una clase carácter, y los caracteres entre un # fuera de una clase carácter y los siguientes caracteres de nueva línea, incluidos, son ignorados también. Esto es equivalente al /x en Perl y hace posible incluir comentarios dentro de patrones complejos. Sin embargo, esto es sólo aplicable a caracteres de información. Los caracteres de espacio en blanco nunca pueden aparecer en la secuencia de caracteres especiales de un patrón, por ejemplo en la secuencia (?(la cual introduce un subpatrón condicional.

e

Si es usado, **preg_replace()** hace las sustituciones \\ de forma habitual, evalúa el código PHP y usa el resultado para realizar una sustitución en la cadena de búsqueda.

Sólo **preg_replace()** hace uso de este modificador y es ignorado por las otras funciones PCRE.

Nota: Este modificador fue añadido en PHP 4.0.

A (PCRE_ANCHORED)

Si es definido, el patrón es forzado a ser "anclado", esto es, es obligado a emparejar sólo desde el inicio de la cadena (el "subject string"). Esta característica también puede realizarse con el apropiado patrón, y esta es la única manera de hacerlo en Perl.

E (PCRE_DOLLAR_ENDONLY)

Si es usado, el carácter del dólar en el patrón casará sólo con fin de la cadena de entrada (subject). Sin este modificador, un dólar es también emparejado con el carácter inmediatamente antes del de una nueva línea (pero no antes de cualquier otra nueva línea). Este modificador es ignorado si m es definido. No hay equivalente en Perl para este modificador.

S

Cuando un patrón va a ser usado varias veces, es mejor dedicar más tiempo a analizarlo para acelerar el proceso de casamientos. Si es definido entonces se realizar un análisis adicional. Estudiar a un patrón es sólo útil para los no anclados, esto es, no tienen un carácter de inicio fijo.

U (PCRE_UNGREEDY)

Este modificador invierte la "codicia" de los cuantificadores aunque no son ansiosos por defecto, se vuelven codiciosos si son seguidos por un "?". No es compatible con Perl. También puede usarse dentro del patrón.

X (PCRE_EXTRA)

Este modificador activa características adicionales del PCRE que no son compatibles con Perl. Cualquier barra invertida en el patrón que sea seguida por una letra que no tenga una interpretación especial provocará un error, estas combinaciones están reservadas para futuras ampliaciones. Por defecto, como en Perl, una barra invertida seguida por una letra sin un significado especial es tratada literalmente. No hay otras características controladas por este modificador a la fecha de hoy.

Sintaxis de los Patrones (unknown)

describe la sintaxis de PCRE regex

La librería PCRE es un conjunto de funciones que implementan emparejamientos dados patrones de expresiones regulares usando la misma sintaxis y semántica que Perl 5, con unas pocas diferencias (ver más adelante). La actual versión corresponde a Perl 5.005.

Las diferencias descritas aquí son con respecto a Perl 5.005.

1. Por defecto, un carácter de espacio en blanco es cualquier carácter que la función `isspace()` de la librería C reconozca, así es posible compilar PCRE con tablas alternativas de tipos de caracteres. Normalmente `isspace()` casa con el espacio, salto de pagina, nueva línea, retorno de carro, tabulador horizontal y vertical. Perl 5 ya no incluye el tabulador vertical en su conjunto de caracteres de espacio en blanco. La secuencia de escape `\n` que estuvo durante mucho tiempo en la documentación de Perl nunca fue reconocida. Sin embargo, el carácter fue tratado como espacio en blanco hasta la 5.002. En 5.004 y 5.005 no casa `\s`.
2. PCRE no permite repetir cuantificadores sobre sentencias hacia adelante. Perl las permite, pero no de la forma que puedes pensar. Por ejemplo, `(?!a){3}` no dice que los próximos tres caracteres no son "a". En realidad significa que los siguientes caracteres no son "a" tres veces.
3. Los subpatrones encontrados dentro de sentencias de más adelante negativas son contados, pero sus entradas en el vector de desplazamientos no son definidas. Perl define sus variables numéricas desde cualquiera de tales patrones que son casados antes de que la sentencia falle emparejar algo, pero solo si las sentencias de más adelante negativas contienen una opción sola.
4. Aunque los caracteres de cero binario son soportados en la cadena de entrada, no son permitidos en un patrón porque son pasados como un cadena típica de C, terminada por cero. La secuencia de escape "`\0`" puede ser usada en el patrón para representar el cero binario.
5. Las siguientes secuencias de Perl no son soportadas:
`\l, \u, \L, \U, \E, \Q`. En efecto, estas son implementadas por manipuladores de cadenas típicos de Perl y no son parte de los patrones del motor de búsqueda.
6. La secuencia `\G` de Perl no es soportada ya que no es relevante para emparejamientos de patrones sencillos.
7. Obviamente, PCRE no soporta el constructor `(?{code})`
8. Hay algunas diferencias en Perl 5.005_02 respecto a las definiciones de las cadenas de captura cuando parte de un patrón es repetido. Por ejemplo, casando "aba" con el patrón `/(a(b)?)+$/` define \$2 al valor "b", pero emparejando "aabbaa" con `/(aa(bb)?)+$/` deja \$2 sin definir. Sin embargo, si el patrón es cambiado a `/(aa(b(b))?)+$/` entonces \$2 (y \$3) son definidos.

En Perl 5.004 \$2 es definido en ambos casos, y también es cierto en PCRE. Si en el futuro Perl cambia a una regla diferente, PCRE puede cambiar para seguirla.

9. Otra discrepancia aún no resuelta es que en Perl 5.005_02 el patrón `/^(a)?(?(1)a|b)+$/` casa la cadena "a", pero en PCRE eso no es así. Sin embargo, en ambos Perl y PCRE `/^(a)?a/` empareja "a" dejando \$1 sin definir.

10. PCRE da algunas extensiones para facilitar las expresiones de PERL:

- (a) Aunque las sentencias de más adelante deben emparejar cadenas de longitud fija, cada opción de una sentencia de punto actual puede casar con una cadena de longitud diferente. Perl 5.005 requiere que todas ellas tengan la misma longitud.
- (b) Si es definido PCRE_DOLLAR_ENDONLY y PCRE_MULTILINE no lo es, el carácter especial \$ sólo casa con el final de la cadena.
- (c) Si se define PCRE_EXTRA, una barra invertida seguida de una letra sin un significado especial provoca un error.
- (d) Si defines PCRE_UNGREEDY, la voracidad de los cuantificadores de repetición es invertida, esto es, por defecto son no codiciosos, pero seguidos por una interrogación si lo son.

La sintaxis y la semántica de las expresiones soportadas por PCRE es descrita a continuación. Las expresiones son descritas en la documentación del Perl y en numerosos libros, algunos de los cuales tienen mucho ejemplos, Jeffrey Friedl's "Mastering Regular Expressions", publicado por O'Reilly (ISBN 1-56592-257-3), las cubre con gran detalle. La presente descripción es propuesta como documentación de referencia.

Una expresión es un patrón que es emparejada repetidamente, dada una cadena de entrada, de izquierda a derecha. Muchos caracteres se representan a ellos mismos en el patrón. Como un ejemplo trivial, el patrón

The quick brown fox

casa una parte de una cadena de entrada que es idéntica a ella. El poder de las expresiones proviene de la posibilidad de incluir alternativas y repeticiones en el patrón. Éstos son codificados en el patrón usando *meta-characters* (caracteres especiales también llamados meta caracteres), los cuales no se representan a ellos mismos, en vez de eso, son interpretados de una manera especial.

Hay dos diferentes conjuntos de caracteres especiales: aquellos que son reconocidos en cualquier parte en el patrón excepto dentro corchetes ('[' y ']'), y aquellos que son reconocidos dentro. Fuera de los corchetes, los caracteres especiales son:

- \ carácter de escape genérico con diferentes usos
- ^ secuencia de inicio de la cadena de entrada (o línea, en modo multilínea)
- \$ secuencia de fin de la cadena de entrada (o línea, en modo multilínea)
- . empareja cualquier carácter excepto el de nueva línea (por defecto)
- [inicia definición de clase de caracteres
- | inicio de opción alternativa

- (inicio de subpatrón
-) fin de subpatrón
- ? amplia el significado de (
 - también es el cuantificador 0 ó 1
 - también es el cuantificador minimizado
- * cero o más cuantificadores
- + uno o más cuantificadores
- { inicia el cuantificador min/max

Parte de un patrón dentro de corchetes ([]) es llamado un "character class" (clase de caracteres). En una clase de caracteres los únicos caracteres especiales son:

- \ carácter de escape genérico
- ^ niega la clase, pero sólo si el primer carácter
- indica un rango de caracteres
-] finaliza la clase de caracteres

Las secciones siguientes describen el uso de cada uno de los caracteres especiales (meta caracteres).

BARRA INVERTIDA

El carácter de barra invertida tiene varios usos. Primero, si es seguido por un carácter que no sea alfanumérico, toma el significado que el carácter pueda tener. Este uso de la barra invertida, como un carácter de escape, se aplica tanto dentro como fuera de las clases de caracteres.

Por ejemplo, si quieras casar un carácter "*", debes escribir "*" en el patrón. Esto es aplicable ya sea o no el carácter siguiente interpretado como un carácter especial, por eso siempre es aconsejable preceder un carácter no alfanumérico con "\" para especificar que se representa a él mismo. En particular, si quieras casar una barra invertida, escribe "\\".

Si el patrón es compilado con la opción PCRE_EXTENDED , los espacios en blanco en el patrón (fuera de una clase de caracteres) y los caracteres entre un "#" fuera de una clase de caracteres y el carácter de nueva línea son ignorados. Una barra invertida de escape puede usarse para incluir un espacio en blanco o el carácter "#" como parte del patrón.

Un segundo uso de la barra invertida sirve para codificar caracteres no imprimibles en los patrones de una manera visible. No hay restricciones sobre la apariencia de los caracteres no imprimibles, quitando el cero binario de terminación de un patrón, pero cuando un patrón es preparado con un editor de texto, normalmente es fácil utilizar una de las siguientes secuencias de escape que representan sus caracteres binarios:

- \a alarma, esto es, el carácter BEL (07 en hexadecimal)
- \cx "control-x", donde x es cualquier carácter
- \e escape (1B en hexadecimal)
- \f nueva página (0C en hexadecimal)
- \n nueva línea (0A en hexadecimal)
- \r retorno de carro (0D en hexadecimal)
- \t tabulador (09 en hexadecimal)
- \xhh carácter con código hh en hexadecimal

\ddd carácter con código ddd en octal

El efecto de "\cx" es como sigue: si "x" es una letra minúscula, es convertida a mayúscula. Entonces el sexto bit del carácter (40 en hexadecimal) es invertido. Esto es, "\cz" es 1A en hexadecimal, pero "\c{" es 3B en hexadecimal, mientras "\c;" es 7B en hexadecimal.

Después de "\x", hasta dos dígitos hexadecimales son leídos (las letras pueden ser mayúsculas o minúsculas).

Después de "\0" son leídos dos dígitos octales más. En ambos casos, si hay menos de dos dígitos, se usará lo que haya. Esto es, la secuencia "\0\x\07" indica dos ceros binarios seguidos por un carácter BEL. Asegúrate dar dos dígitos después del inicial cero si el carácter que sigue es un dígito octal.

El uso de una barra invertida seguido por otro dígito que no sea el cero es complejo. Fuera de una clase carácter, PCRE interpreta cualquier dígito como un número decimal. Si el número es menor que diez, o si ha habido al menos tantos paréntesis capturados a la izquierda en la expresión, entonces la secuencia entera es tomada como una *back reference* (referencia atrás). Una descripción de como trabaja esto es dada después, siguiendo la discusión de subpatrones con paréntesis.

Dentro de una clase carácter, o si el número decimal es mayor que nueve y no ha habido tantos subpatrones capturados PCRE relee los tres dígitos octales siguientes a la barra invertida y genera un byte desde los ocho bits menos significativos del valor. Cualquier dígito a continuación se representa a él mismo. Por ejemplo:

- \040 es otro modo de escribir un espacio
- \40 es lo mismo, siempre que haya menos de cuarenta subpatrones abiertos
- \7 siempre es una referencia atrás
- \11 puede ser una referencia atrás o un tabulador
- \011 siempre es un tabulador
- \0113 es el carácter con código octal 113 (ya que no puede haber más de noventa y nueve referencias atrás)
- \377 es un byte con todos sus bits a uno
- \81 puede ser una referencia atrás o un cero binario seguido por dos caracteres "8" y "1"

Ten en cuenta que el valor octal de un número mayor o igual a cien no debe ser precedido por un cero ya que no son leídos más de tres dígitos octales.

Todas las secuencias que definen el valor de un byte pueden ser usadas tanto dentro como fuera de la clase carácter. Además, la secuencia "\b" es interpretada como el carácter backspace (hex 08) dentro. Fuera es definido de otra manera (ver más adelante).

El tercer uso de la barra invertida es para especificar los tipos de caracteres genéricos:

- \d cualquier un dígito decimal
- \D cualquier carácter que no sea un dígito decimal
- \s cualquier carácter de espacio en blanco (whitespace)
- \S cualquier carácter que no sea un espacio en blanco

- \w cualquier carácter de "palabra"
- \W cualquier carácter que no se de "palabra"

Cada pareja de secuencia de escape divide el conjunto global de caracteres en dos. Cualquier carácter dado empareja en uno y sólo uno de cada pareja.

Un carácter de "palabra" es cualquier letra o dígito o el carácter subrayado, esto es, cualquier carácter puede ser parte de una "palabra" en Perl. La definición de letras y dígitos es controlada por la tabla de caracteres de PERL, y puede ser variada si las especificaciones regionales son tomadas en cuenta (ver "Soporte regional más adelante"). Por ejemplo, en Francia algunos caracteres tienen un código superior a 128, para representar las letras acentuadas, y son emparejados por \w.

Estas secuencias de tipos de caracteres pueden aparecer tanto dentro como fuera de las clases carácter. Cada una casa un carácter del tipo apropiado. Si el punto de casamiento actual es el final de la cadena, todo ello falla, ya que no hay más caracteres que casar.

El cuarto uso de la barra invertida es para ciertas sentencias (assertions). Una sentencia especifica una condición que tiene que ser encontrada en un punto particular de un emparejamiento, sin utilizar ningún carácter de la cadena de entrada. El uso de subpatrones para sentencias más complicadas es descrito después. Las sentencias de barra invertida son

- \b límites de palabra
- \B no sean límites de palabra
- \A inicio de la cadena de entrada (independiente del modo multilínea)
- \Z fin de la cadena de entrada o de una nueva línea delante del final
(independiente del modo multilínea)
- \z fin de la cadena de entrada (independiente de modo multilínea)

Estas sentencias no pueden aparecer dentro de una clase carácter (pero ten en cuenta que "\b" tiene un significado diferente, quiere decir el carácter backspace dentro de una clase carácter)

Un límite de palabra es una posición en la cadena de entrada donde un carácter y el anterior no emparejan con \w o \W (por ejemplo, uno casa con \w y el otro con \W), o el principio o el final de la cadena si el primero o el último carácter emparejan con \w, respectivamente.

Las sentencias \A, \Z y \z se diferencian de los tradicionales circunflejo y dólar (ver más adelante) en que sólo emparejan el inicio y fin de la cadena de entrada sin tener en cuenta las opciones definidas. No les afectan las opciones PCRE_NOTBOL o PCRE_NOTEOL. La diferencia entre \Z y \z es que \Z casa antes una nueva línea que es el último carácter de la cadena como también el final de la cadena, sin embargo \z sólo casa el final.

CIRCUNFLEJO Y DOLAR

Fuera de una clase carácter, en el modo de emparejamiento por defecto, el carácter circunflejo es una sentencia la cual es verdadera sólo si el punto de casamiento actual es el inicio de la cadena de entrada. Dentro de una clase carácter, el circunflejo tiene significado completamente distinto (ver más adelante).

El circunflejo no necesita ser el primer carácter del patrón si son posibles un número de alternativas, pero será la primera cosa en cada alternativa en la cual aparezca si el patrón casa esa opción.

Si todas las alternativas posibles empiezan con un circunflejo, esto es, si el patrón es obligado a casar sólo con en el inicio de la cadena de entrada, se dice que es un patrón "anclado". También hay otros constructores que pueden hacer que un patrón sea anclado.

Un carácter de dólar es una sentencia que es verdadera sólo si el punto de emparejamiento actual es el final de la cadena de entrada, o inmediatamente antes de un carácter de nueva línea, el cual es el último carácter en la cadena, por defecto. El dólar no necesita ser el último carácter del patrón si hay varias alternativas, pero será el último elemento en cualquier alternativa en el que aparezca. El dólar no tiene un significado especial en una clase carácter.

El significado del dólar puede ser cambiado para que sólo empareje el final de la cadena de entrada definiendo la opción PCRE_DOLLAR_ENDONLY a la hora de compilar o tiempo de ejecución. Esto no afecta a la sentencia \Z.

El significado de los caracteres circunflejo y dólar cambia si la opción PCRE_MULTILINE es definida. Cuando éste es el caso, casan, respectivamente, inmediatamente antes y después de un carácter "\n" interno, además de emparejar con el inicio y el final de la cadena. Por ejemplo, el patrón /^abc\$/ casa con la cadena de entrada "def\nabc" en modo multilínea, pero en otro modo no. Consecuentemente, los patrones anclados son en modo línea ya que todas las opciones que empiezan con "^" no son ancladas en modo multilínea. La opción PCRE_DOLLAR_ENDONLY es ignorada si PCRE_MULTILINE es definido.

Ten en cuenta que las secuencias \A, \Z y \z pueden ser usadas para casar el inicio y el final de la cadena en ambos modos, y si todas las opciones de un patrón empiezan con \A siempre es anclado, independientemente de si PCRE_MULTILINE es definido o no.

FINAL (PUNTO)

Fuera de una clase carácter, un punto en el patrón casa con un carácter cualquiera en la cadena de entrada, incluyendo un carácter no imprimible, exceptuando el de nueva línea (por defecto). Si la opción PCRE_DOTALL es definida, entonces los puntos casan con los de nueva línea también. El manejo de puntos es completamente independiente del uso del circunflejo y el dólar, la única relación entre ellos son los caracteres de nueva línea. Los puntos no tienen un significado especial dentro de una clase carácter.

CORCHETES

Un corchete de apertura crea una clase carácter, terminada por uno de cierre. Un corchete de cierre no tiene un significado especial. Si un corchete de cierre es necesitado como un miembro de la clase, será el primer carácter de datos en la clase (después de un circunflejo inicial, si está presente) o con una barra invertida antes.

Si una clase carácter casa con un carácter único en la cadena; el carácter debe estar en el conjunto de los caracteres definidos por la clase, a menos que el primero sea un circunflejo, en cuyo caso el carácter de la cadena de entrada no debe estar en el conjunto definido por la clase. Si un circunflejo

es necesitado como un miembro de la clase, asegúrate que no es el primero o es precedido por una barra invertida.

Por ejemplo, la clase carácter [aeiou] empareja cualquier vocal minúscula, mientras [^aeiou] casa cualquier carácter que no sea una vocal minúscula. Ten en cuenta que un circunflejo es una notación convenida para especificar los caracteres que están en la clase enumerando los que no lo están. No es una sentencia: consume un carácter de la cadena de entrada y falla si el punto actual es final.

Cuando se define el emparejamiento sin tener en cuenta mayúsculas y minúsculas (caseless), cualquier letra en una clase representa ambas, por ejemplo, un patrón caseless [aeiou] empareja tanto "A" como "a" y un caseless [^aeiou] no casa con "A"

El carácter de nueva línea nunca es tratado de un modo especial en una clase carácter, aunque se hallan definido cualquiera de las opciones PCRE_DOTALL o PCRE_MULTILINE. Una clase como [^a] siempre casa con una nueva línea.

El carácter de menos puede ser usado para especificar un rango de caracteres en una clase miembro. Por ejemplo, [d-m] casa con cualquier letra entre d y m ambas incluidas. Si un carácter de menos es necesario en una clase, debe ser precedido por una barra invertida o aparecer en una posición donde no pueda ser interpretado como indicador de una rango, normalmente al inicio o al final de la clase.

No es posible tener el carácter literal "]" como el de final de un rango. Un patrón como [W-]46] es interpretado como una clase de dos caracteres ("W" y "-") seguido por la cadena literal "46]", por lo que emparejaría con "W46]" o "-46]". Sin embargo, si el carácter "]" es precedido con una barra invertida es tomado por el final del rango, así [W-]46] es interpretado como una clase conteniendo un rango seguido por dos caracteres. La representación octal o hexadecimal de "]" puede ser usada para finalizar un rango.

Los rangos trabajan en la secuencia ASCII. Se pueden especificar mediante la representación numérica de los mismos, por ejemplo [\000-\037]. Si un rango que incluye letras es usado cuando es definida la opción de no tener en cuenta mayúsculas y minúsculas casan ambas. Por ejemplo, [W-c] es equivalente a [][^_wxyzabc], teniendo en cuenta mayúsculas y minúsculas, y si la tabla de caracteres para la región "fr" es usada, entonces [\xc8-\xcb] empareja los caracteres E acentuados en ambos casos.

Los tipos de caracteres \d, \D, \s, \S, \w, y \W también pueden aparecer en una clase carácter y añaden los caracteres que ellos casen para la clase. Por ejemplo, [\dABCDEF] casa cualquier dígito hexadecimal. Un circunflejo puede ser usado convenientemente con el tipo de carácter mayúsculo para especificar un conjunto más restrictivo de caracteres que el de un casamiento con tipo de carácter minúsculo. Por ejemplo, la clase [^\W_] empareja cualquier letra o dígito pero no el subrayado.

Todos los caracteres no alfanuméricos y los diferentes a \, -, ^ (al principio) y] no tienen un significado especial en una clase, y éstos tampoco si son definidos convenientemente.

BARRA VERTICAL

Los caracteres de barra vertical son usados para separar patrones alternativos. Por ejemplo, el patrón

```
gilbert|sullivan
```

casa con "gilbert" o "sullivan". Cualquier cantidad de opciones pueden ser implementadas, y una alternativa vacía se permite (emparejando la cadena vacía). El proceso de casamiento intenta cada una de izquierda a derecha, y la primera que valga es usada. Si las alternativas están dentro de un subpatrón, "valga" significa que casa el resto del patrón principal como también la alternativa en el subpatrón.

DEFINIENDO LAS OPCIONES INTERNAS

Las definiciones de PCRE_CASELESS, PCRE_MULTILINE, PCRE_DOTALL, y PCRE_EXTENDED pueden ser cambiadas desde dentro del patrón mediante una secuencia de letras de opciones de Perl encerradas entre "(?" y ")".

Las letras de opciones son

- i para PCRE_CASELESS
- m para PCRE_MULTILINE
- s para PCRE_DOTALL
- x para PCRE_EXTENDED

Por ejemplo, (?im) define sin tener en cuenta mayúsculas y minúsculas y modo multilínea. También es posible eliminar estas opciones precediendo las letras con un menos y una combinación de definiciones y eliminaciones tal como (?im-sx), la cual define PCRE_CASELESS y PCRE_MULTILINE mientras elimina PCRE_DOTALL y PCRE_EXTENDED, también se permite. Si una letra aparece antes y después del menos, la opción es eliminada.

El ámbito de estas opciones cambia dependiendo dónde ocurra la definición. Las definiciones que son hechas fuera de subpatrones (como antes), el efecto es el mismo que si la opción se define o elimina al inicio del casamiento. Los siguientes patrones se comportan todos de la misma manera:

```
(?i)abc
a(?i)bc
ab(?i)c
abc(?i)
```

el cual tiene el mismo efecto que compilar el patrón abc con la opción PCRE_CASELESS. En otras palabras, tales definiciones de "nivel superior" se aplican a todo el patrón (a menos que haya otro cambio dentro del subpatrón). Si hay más de una definición de la misma opción en el mismo nivel superior, la definición más a la derecha se usa.

Si un cambio de opción sucede dentro de un subpatrón, el efecto es diferente. Esto es un cambio respecto de la conducta de Perl 5.005. Un cambio de opción dentro de un subpatrón afecta sólo a la parte del subpatrón que lo sigue, por eso

```
(a(?i)b)c
```

empareja abc y aBc y ninguna otra cadena (asumiendo que no es usado PCRE_CASELESS). De este modo, las opciones pueden ser hechas para tener diferente significado en diferente partes del patrón. Cualquier cambio realizado en una alternativa provoca que todo el subpatrón la use. Por ejemplo,

```
(a(?:i)b|c)
```

empareja "ab", "aB", "c", y "C", siempre y cuando case "C" la primera opción es abandonada antes de definir la opción. Esto es porque los efectos de definiciones de opción ocurren en tiempo de compilación. De otro modo, éstos sería una conducta muy rara.

Las opciones específicas PCRE PCRE_UNGREEDY y PCRE_EXTRA pueden ser cambiadas del mismo modo que las opciones compatibles con Perl usando los caracteres U y X respectivamente. La bandera (?X) es especial ya que siempre debe aparecer antes que cualquier otra en el patrón, incluso cuando es definida a nivel superior. Es mejor ponerla en el inicio.

SUBPATRONES

Los subpatrones son delimitados por paréntesis y pueden estar anidados. Marcando parte de un patrón como un subpatrón permite dos cosas:

1. Define un conjunto de opciones. Por ejemplo, el patrón

```
cat(aract|erpillar|)
```

empareja con "cat", "cataract", or "caterpillar". Sin los paréntesis, casaría "cataract", "erpillar" o la cadena vacía.

2. Define el subpatrón como un subpatrón capturado. Cuando el patrón sea emparejado por completo, esa porción de la cadena de entrada que casa con el subpatrón es devuelta mediante el argumento *ovector* de **pcre_exec()**. Los paréntesis abiertos son contados de izquierda a derecha (empezando por uno) para definir los números de subpatrones capturados.

Por ejemplo, si la cadena "the red king" es casada con el patrón

```
the ((red|white) (king|queen))
```

las subcadenas capturadas son "red king", "red", y "king" y los números son 1,2 y 3

El hecho de que los paréntesis realicen dos funciones no siempre es útil. A menudo, hay veces que un subpatrón agrupado es necesario sin una querer una captura. Si un paréntesis abierto le sigue "?:", el subpatrón no hace ninguna captura, y no es contado cuando compute el número de subpatrones capturados. Por ejemplo, si la cadena "the white queen" es casada con el patrón

```
the ((?:red|white) (king|queen))
```

las subcadenas capturadas son "white queen" y "queen" y son numeradas como

1 y 2. El número máximo de subcadenas es de 99 y el número máximo de subpatrones, capturados o no, es de 200.

Como un atajo, si cualquier definición de opción es necesitada al inicio de un subpatrón no capturado, las letras de opciones pueden aparecer entre "?" y ":". Así los dos patrones

```
(?i:saturday|sunday)
(?:(?i)saturday|sunday)
```

emparejan como el mismo conjunto de cadena de entrada exactamente. Ya que las alternativas son intentadas de izquierda a derecha, y las opciones no son dejadas de tener en cuenta hasta que el final de subpatrón se alcanza, una definición de opción en una alternativa afecta al resto, por eso el patrón anterior empareja tanto con "SUNDAY" como con "Saturday".

REPETICION

La repetición es especificada por cuantificadores, la cual puede utilizarla cualquiera de los siguientes elementos:

- un carácter, posiblemente precedido por el meta carácter .
- una clase carácter
- una referencia atrás (ver la próxima sección)
- un subpatrón con paréntesis (a menos que sea una sentencia, ver más adelante)

El cuantificador de repetición general indica un número mínimo y un máximo de casamientos permitidos, dando los dos números entre llaves, separados por coma. El número debe ser menor que 65536, y el primero debe ser menor o igual que el segundo. Por ejemplo:

`z{2,4}`

casa con "zz", "zzz", o "zzzz". Una llave de cierre por si misma no es un carácter especial. Si el segundo número es omitido, pero aparece la coma, entonces no hay límite superior; si el segundo número y la coma son omitidos, el cuantificador indica el número exacto de repeticiones. Así

`[aeiou]{3,}`

empareja al menos tres vocales seguidas, pero pueden ser muchas más, mientras

`\d{8}`

casa exactamente ocho dígitos. Una llave abierta en una posición donde un cuantificador no es permitido o una que no empareje con la sintaxis de un cuantificador es tomada como un carácter literal. Por ejemplo, `{,6}` no es un cuantificador, pero sí una cadena literal de cuatro caracteres.

Se permite el cuantificado `{0}`, provocando que la expresión se comporte como si el elemento anterior y el cuantificador no estuvieran presentes.

Por conveniencia (y compatibilidad histórica) los cuantificadores más comunes tienen abbreviaciones de un solo carácter.

* es equivalente a `{0,}`

- + es equivalente a {1,}
- ? es equivalente a {0,1}

Es posible construir bucles infinitos mediante un subpatrón que pueda casar ningún carácter con un cuantificador que no tenga límite superior, por ejemplo:

(a?)^{*}

Las primeras versiones de Perl y PCRE dan un error en tiempo de compilación para tales patrones. Sin embargo, ya que existen casos donde esto puede ser útil, estos patrones son aceptados ahora, pero si cualquier repetición del subpatrón no casa ningún carácter, el bucle es roto.

Por defecto, los cuantificadores son "codiciosos", esto es, casan tantas veces como sea posible (hasta el número máximo de veces permitido), sin provocar que el resto del patrón falle. El ejemplo clásico de donde viene este problema es en intentar casar comentarios en los programas en C. Estos aparecen entre las secuencias /* y */ y dentro de la secuencia los caracteres * y / pueden aparecer individualmente. Un modo de casar comentarios en C es aplicando el patrón

/*.**/

para la cadena

/* first command */ not comment /* second comment */

falla, porque casa la cadena entera debido a la voracidad del elemento .*

Sin embargo, si un cuantificador le sigue un signo de interrogación entonces cesa la voracidad y empareja el mínimo número de veces posibles, así el patrón

/*.*?*/

hace las cosas correctamente con los comentarios en C. El significado de los cuantificadores variables no es cambiado en otro modo, justo el número preferido de casamientos. No confundas el uso de las interrogaciones con su uso como un cuantificador mas. Ya que tiene dos usos, a veces puede parecer doble, como en

\d??\d

el cual empareja un dígito normalmente, pero puede casar dos si ese es el único modo de casar el resto del patrón.

Si se define la opción PCRE_UNGREEDY (la cual no es posible en Perl) entonces los cuantificadores no son voraces por defecto, pero uno puede serlo seguido por una interrogación. En otras palabras, invierte la conducta por defecto.

Cuando un subpatrón entre paréntesis es cuantificado con un número mínimo de repeticiones superior a uno o con un límite máximo, se necesita más almacenamiento para compilar el patrón, en proporción al tamaño del mínimo o del máximo.

Si un patrón empieza con `.{0,}` y la opción PCRE_DOTALL (equivalente a `/s` del Perl) es definida, esta permitiendo el `.` para casar nuevas líneas, entonces el patrón es anclado implícitamente. PCRE trata tales patrones como si estuvieran precedidos por `\A`. En los casos donde se conoce que la cadena de entrada no contiene nuevas líneas, es conveniente definir PCRE_DOTALL cuando el patrón empieza con `.*` para obtener esta optimización o usar `^` para indicar explícitamente anclamiento.

Cuando un subpatrón capturado es repetido, el valor capturado es la subcadena que empareja la iteración final. Por ejemplo, el patrón

```
(tweedle[dume]{3}\s*)+
```

con la cadena de entrada "tweedledum tweedledee" el valor de la subcadena capturada es "tweedledee". Sin embargo, si hay subpatrones capturados anidadamente, los valores capturados correspondientes pueden haber sido definidos en las iteraciones anteriores. Por ejemplo, después de casar "aba" con

```
/(a|(b))+/
```

el valor de la segunda subcadena capturada es "b".

REFERENCIAS ATRAS

Fuera de una clase carácter, una barra invertida seguida por un dígito mayor que cero (y posiblemente más dígitos) es una referencia atrás a un subpatrón capturado antes (a su izquierda) en el patrón, siempre que haya habido tantos paréntesis a la izquierda capturados.

Sin embargo, si el número decimal seguido por la barra invertida es menor que diez, siempre es tomado como una referencia atrás, y da error sólo si no hay los suficientes subpatrones capturados en todo el patrón. En otras palabras, los paréntesis que son referidos no necesitan estar a la izquierda de la referencia para un número menor de diez. Examina la sección anterior titulada "Barra invertida" para más detalles del manejo de los dígitos con la barra invertida.

Una referencia atrás empareja si casa el subpatrón capturado en el actual punto de la cadena de entrada, mejor que casar cualquier subpatrón de la misma. Así el patrón

```
(sens|respons)e and \1ibility
```

casa con "sense and sensibility" y "response and responsibility", pero no "sense and responsibility". Si el casamiento con la distinción entre minúsculas y mayúsculas está activado en el momento de la referencia atrás, entonces la distinción de las letras es relevante. Por ejemplo,

```
((?i)rah)\s+\1
```

casa con "rah rah" y "RAH RAH", pero no "RAH rah", pero el subpatrón capturado originalmente es emparejado sin la distinción.

Puede haber más de una referencia atrás en el mismo subpatrón. Si un

subpatrón no ha sido usado en un emparejamiento particular, entonces cualquier referencia atrás siempre fallara. Por ejemplo, el patrón

`(a|(bc))\2`

fallará siempre si inicia a casar con "a" mejor que con "bc". Ya que puede haber hasta 99 referencias atrás, todos los dígitos seguidos por una barra invertida son tomados como parte de número potencial de referencias atrás. Si el patrón continua con un carácter de dígito, entonces algún delimitador debe ser usado para terminar la referencia atrás. Si la opción PCRE_EXTENDED es definida, este puede ser el espacio en blanco. De otro modo un comentario vacío puede ser usado.

Una referencia atrás ocurre dentro del paréntesis al cual refiere, falla cuando el subpatrón es usado por primera vez, así por ejemplo, `(a\b{1})` nunca emparejará. Sin embargo, tal referencia puede ser útil dentro de los subpatrones repetidos. Por ejemplo, el patrón

`(a|b\b{1})+`

casa con cualquier número de "a"s y también con "aba", "ababaa" etc. Para cada iteración del subpatrón, la referencia atrás casa la cadena de caracteres correspondiente a la iteración anterior. Para que esto trabaje, el patrón debe ser tal que la primera iteración no necesite casar la referencia atrás. Esto puede hacerse usando alternativas, como en el ejemplo anterior, o por medio de cuantificadores con un número mínimo de cero.

SENTENCIAS

Una sentencia es un test sobre los caracteres siguiendo o precediendo el punto actual de emparejamiento que no consume caracteres. Las sentencias codificadas como `\b`, `\B`, `\A`, `\Z`, `\z`, `^` y `$` son descritas después. Las sentencias más complejas son codificadas como subpatrones. Hay dos clases: aquellas que condicionan más adelante de la posición actual en la cadena de entrada (lookahead) y las que lo hacen en este punto (lookbehind).

Un subpatrón de sentencia es emparejado del modo típico, excepto que no hace que el punto actual de emparejamiento cambie. Sentencias que condicionan más adelante empiezan con `(?=` para sentencias afirmativas y `(?!?` para las negativas

`\w+(?=;)`

empareja una palabra seguida por un punto y coma. pero no incluye el punto y coma en el casamiento, y

`foo(?!bar)`

casa cualquier ocurrencia de "foo" que no es seguida por "bar". Ten en cuenta que el patrón similar

`(?!foo)bar`

no encuentra una ocurrencia de "bar" que es precedida por algo que no sea "foo"; encuentra cualquier ocurrencia de "bar", ya que la sentencia `(?!foo)` es siempre verdadera cuando los tres primeros caracteres son "bar". Una sentencia en el

punto actual es necesaria para realizar este efecto. Las sentencias de punto actual empiezan con (?=< para sentencias afirmativas y (?<! para las negativas. Por ejemplo,

```
(?<!foo)bar
```

encuentra una ocurrencia de "bar" que no es precedida por "foo". Los contenidos de un sentencia de punto actual están limitados para que todas las cadenas que emparejen deban tener una longitud fijada. Sin embargo, si hay varias alternativas, no todas tienen que tener la misma longitud. Así

```
(?<=bullock|donkey)
```

es permitido, pero

```
(?<!dogs?|cats?)
```

da error en tiempo de compilación. Opciones que emparejen diferentes longitudes de cadenas son permitidas sólo a nivel superior de la sentencia de punto actual. Ésta es una extensión comparada con Perl 5.005, la cual requiere que todas las opciones a casar tengan la misma longitud. Una sentencia como

```
(?<=ab(c|de))
```

no es permitida, ya que sus opciones a nivel superior pueden casar dos longitudes diferentes, pero es aceptable si se rescribe para usar dos opciones a nivel superior:

```
(?<=abc|abde)
```

La implementación de sentencias de punto actual es, para cada alternativa, mover temporalmente la posición actual hacia atrás por la longitud fijada e intentar casar. Si no hay suficientes caracteres antes de la posición actual, fallará. Las sentencias de punto actual en unión con subpatrones de sólo una vez pueden ser particularmente útiles para emparejamientos de finales de cadenas; un ejemplo es dado al final de la sección sobre subpatrones de una sola vez.

Varias sentencias (de cualquier tipo) pueden suceder consecutivamente. Por ejemplo,

```
(?<=|d{3})(?<!999)foo
```

empareja "foo" precedido por tres dígitos que no sean "999". Además, las sentencias puede ser anidadas en cualquier combinación. Por ejemplo,

```
(?<=(?<!foo)bar)baz
```

empareja una ocurrencia de "baz" que es precedida por "bar" la cual no sea precedida por "foo".

Los subpatrones de sentencias no son subpatrones capturados, y no pueden ser repetidos, ya que no tiene sentido la misma cosa varias veces. Si una sentencia contiene subpatrones capturados dentro de ella, éstos son siempre

contados para el propósito de la numeración de los subpatrones capturados en todo el patrón. Las subcadenas capturadas son tenidas en cuenta para las sentencias afirmativas, pero no para las negativas (no tiene sentido).

El contador de sentencias llega hasta un máximo de doscientos subpatrones con paréntesis.

SUBPATRONES DE UNA SOLA VEZ

Maximizando y minimizando las repeticiones para ver si un número diferente de éstas permite al resto del patrón emparejar, causa múltiples evaluaciones de la cadena de entrada. A veces es útil prevenir esto, cambiando el patrón o provocando que la repetición falle pronto, cuando el creador del patrón conoce que no hay puntos en común.

Considera, por ejemplo, el patrón `\d+foo` cuando se aplica a esta cadena de entrada

```
123456bar
```

Después de emparejar los seis dígitos falla al emparejar "foo", la acción normal del casamiento es intentar otra vez con sólo cinco dígitos que emparejen con el elemento `\d+`, y entonces con cuatro, y así, antes de fallar. Subpatrones de una sola vez dan el modo de especificar que una parte del patrón tiene que emparejar, no es re-evaluado de esta manera, así el casamiento fallará al emparejar "foo" la primera vez. La notación es otra clase de paréntesis especial, iniciado con `(?>`; como en este ejemplo:

```
(?>\d+)bar
```

Esta clase de paréntesis "bloquean" la parte del patrón que tiene que ser emparejada una vez y un fallo impide que la re-evalue.

Una descripción alternativa es que un subpatrón de este tipo case los caracteres de la cadena que un patrón fijo emparejaría, si estuviera anclado en el punto actual de la cadena de entrada.

Subpatrones de una sola vez no son subpatrones capturados. Estos casos tal como el ejemplo anterior pueden ser interpretado como de una repetición maximizada que debe tragar todo lo que pueda. Por esto, mientras ambos `\d+` y `\d?` están preparados para ajustar el número de dígitos que emparejan para hacer que el resto del patrón case, `(?>\d+)` sólo puede emparejar una secuencia de dígitos entera.

Esta construcción, por supuesto, puede contener subpatrones arbitrariamente complicados y pueden estar anidados.

Subpatrones de una sola vez pueden usarse con sentencias de punto actual para especificar eficientes emparejamientos al final de la cadena de entrada. Consideremos un patrón sencillo como este

```
abcd$
```

cuando se aplica a una cadena larga con la cual no empareja. Ya que el casamiento va de izquierda a derecha, PCRE buscará cada "a" en la cadena y entonces verá si lo que sigue casa con el resto del patrón. Si el patrón

se escribe así

`^.*abcd$`

entonces el `.*` inicial casará primero la cadena entera, pero cuando esto falle, volverá atrás para emparejar todo menos el último carácter, entonces los dos últimos y así sucesivamente. Otra vez la búsqueda de "a" cubre la cadena completa, de derecha a izquierda, de esta manera no se mejora. Sin embargo, si el patrón fuese este

`^(?>.*)(?=<abcd)`

entonces no hay vuelta atrás para el elemento `.*`; sólo puede casar la cadena entera. La sentencia de punto actual subsiguiente hace un test sencillo sobre los últimos cuatro caracteres. Si falla, el casamiento inmediatamente da un resultado negativo. Para cadenas largas, este acercamiento da una diferencia significativa en tiempo de ejecución.

SUBPATRONES CONDICIONALES

Es posible hacer que el casamiento procese un subpatrón condicionalmente o elegir entre dos subpatrones alternativos, dependiendo del resultado de una sentencia o si un subpatrón capturado previamente casó o no.

Las dos formas posibles de subpatrones condicionales son

`(?(condition)yes-pattern)`
`(?(condition)yes-pattern|no-pattern)`

Si la condición es satisfecha, el yes-pattern es usado; sino el no-pattern es utilizado si existe. Si hay más de dos alternativas en el subpatrón, se produce un error en tiempo de compilación.

Hay dos clases de condiciones. Si el texto entre los paréntesis consiste de una secuencia de dígitos, entonces la condición es verdadera si el subpatrón capturado de ese número ha sido casado previamente. Consideremos el siguiente patrón, contiene espacios en blanco para hacerlo más leíble (asumimos la opción PCRE_EXTENDED) y lo dividimos en tres partes para facilitar la discusión:

`(\()? [^()]+ (?1 \))`

La primera parte empareja un paréntesis opcional abierto, y si el carácter está presente, lo define como la primera subcadena capturada. La segunda parte casa uno o más caracteres que no están entre paréntesis. La tercera parte es un subpatrón condicional que examina si el primer conjunto de paréntesis casa o no. Si fuera así, esto es, si la cadena de entrada empieza por un paréntesis abierto, la condición es cierta, y el yes-pattern es ejecutado y un paréntesis de cierre es requerido. De otro modo, ya que no-pattern no está presente, el subpatrón no casa con nada. En otras palabras, este patrón casa una secuencia de datos sin paréntesis opcionalmente limitada por ellos.

Si la condición no es una secuencia de dígitos, debe ser una sentencia. Esto puede ser una sentencia de más adelante positiva o negativa o una de punto actual. Consideremos este patrón, otra vez conteniendo espacios

en blanco sin significado y con la segunda alternativa en la siguiente línea:

```
(?=(?=^[a-z]*[a-z])
 \d{2}[a-z]{3}-\d{2} | \d{2}-\d{2}-\d{2} )
```

La condición es una sentencia de más adelante positiva que empareja una secuencia opcional de cualquier cosas menos letras seguido por una letra. En otras palabras, examina la presencia de al menos una letra en la cadena de entrada. Si una letra es encontrada, la cadena es casada con la primera alternativa; sino lo es con la segunda. Este patrón casa cadenas de una de estas dos formas dd-aaa-dd o dd-dd-dd, donde aaa son letra y dd son dígitos.

COMENTARIOS

La secuencia (# marca el inicio de un comentario el cual continua hasta el primer paréntesis. Los paréntesis anidados no son permitidos. Los caracteres que forman un comentario no forman parte del patrón de emparejamiento.

Si la opción PCRE_EXTENDED es definida, un carácter # fuera de una clase carácter crea un comentario que continua hasta la próxima línea del patrón.

RENDIMIENTO

Ciertos elementos que pueden aparecer en los patrones son más eficientes que otros. Es más eficiente usar una clase carácter como [aeiou] que un conjunto de alternativas tal como (a|e|i|o|u). En general, los constructores más sencillos que dan la conducta requerida son, normalmente, más eficientes. El libro de Jeffrey Friedl contiene un montón de discusiones sobre la optimización de expresiones regulares para un rendimiento eficiente.

Cuando un patrón empieza con .* y la opción PCRE_DOTALL está definida, el patrón es anclado implícitamente por PCRE, ya que sólo puede casar el inicio de la cadena de entrada. Sin embargo, si PCRE_DOTALL no es definido, PCRE no puede hacer esta optimización, ya que el meta carácter . no tiene porque casar con una nueva línea y si la cadena de entrada contiene varias nuevas líneas, el patrón puede emparejar desde el carácter inmediatamente siguiente a uno de ellos en vez del inicio. Por ejemplo, el patrón

(.*) second

casa la cadena de entrada "first\nand second" (donde \n representa un carácter de nueva línea) con la primera subcadena capturada empezando con "and". En otras palabras, PCRE tiene que intentar los casamientos iniciándolos después de cada nueva línea en la cadena de entrada.

Si estás usando un patrón con cadenas de entrada que no contienen nuevas líneas, el mejor rendimiento se obtiene definiendo PCRE_DOTALL o iniciando el patrón con ^.* para indicar anclamiento explícito. Esto previene a PCRE tener que examinar toda la cadena de entrada buscando nuevas líneas para empezar de nuevo.

LXIV. Funciones para expresiones regulares

Las expresiones regulares se usan en PHP para manipular cadenas complejas. Las funciones que soportan expresiones regulares son:

- **ereg()**
- **ereg_replace()**
- **eregi()**
- **eregi_replace()**
- **split()**

En todas estas funciones, el primer argumento es una expresión regular. PHP utiliza las expresiones regulares extendidas de POSIX, definidas en POSIX 1003.2. Para una descripción completa de las expresiones regulares POSIX, ver las páginas de manual de regex incluidas en el directorio regex de la distribución de PHP. Están en formato de página de manual, por lo que se deben leer con una orden como **man /usr/local/src/regex/regex.7**.

Ejemplo 1. Ejemplos de expresiones regulares

```
ereg("abc",$string);
/* Devuelve true si "abc"
   se encuentra en $string. */

ereg("^abc",$string);
/* Devuelve true si "abc"
   se encuentra al comienzo de $string. */

ereg("abc$",$string);
/* Devuelve true si "abc"
   se encuentra al final de $string. */

eregi("(ozilla.[23]|MSIE.3)",$HTTP_USER_AGENT);
/* Devuelve true si el navegador cliente
   es Netscape 2, 3 o MSIE 3. */

ereg("[[:alnum:]]+ ([[:alnum:]]+) ([[:alnum:]]+)",
     $string,$regs);
/* Pone tres palabras separadas por espacios
   en $regs[1], $regs[2] y $regs[3]. */

$string = ereg_replace("^","<BR>",$string);
/* Coloca la etiqueta <BR> al comienzo de $string. */

$string = ereg_replace("$","<BR>",$string);
/* Coloca la etiqueta <BR> al final de $string. */

$string = ereg_replace("\n","","$string");
/* Elimina los caracteres fin-de-línea de $string. */
```


ereg (PHP 3, PHP 4)

Coincidencia de expresiones regulares

```
int ereg (string pattern, string string [, array regs])
```

Busca en *string* las coincidencias con la expresión regular *pattern*.

Si se encuentran coincidencias con subcadenas entre paréntesis de *pattern* y la función se ha llamado con el tercer argumento *regs*, las coincidencias se almacenarán en los elementos de *regs*. \$regs[1] contendrá la subcadena que empieza en el primer paréntesis izquierdo; \$regs[2] la que comienza en el segundo, etc. \$regs[0] contendrá una copia de *string*.

La búsqueda diferencia mayúsculas y minúsculas.

Devuelve un valor verdadero si se encontró alguna coincidencia, o falso si no se encontraron coincidencias u ocurrió algún error.

El siguiente fragmento de código toma una fecha en formato ISO (AAAA-MM-DD) y la imprime en formato DD.MM.AAAA:

Ejemplo 1. **ereg()** example

```
if ( ereg( "([0-9]{4})-([0-9]{1,2})-([0-9]{1,2})", $date, $regs ) ) {
    echo "$regs[3].$regs[2].$regs[1]";
} else {
    echo "Invalid date format: $date";
}
```

Ver también **eregi()**, **ereg_replace()**, y **eregi_replace()**.

ereg_replace (PHP 3, PHP 4)

reemplaza expresiones regulares

```
string ereg_replace (string pattern, string replacement, string string)
```

Esta función examina *string* buscando coincidencias de *pattern*, y reemplaza el texto encontrado con *replacement*.

Devuelve la cadena modificada. Si no hay coincidencias que reemplazar, devuelve la cadena original.

Si *pattern* contiene subcadenas entre paréntesis, *replacement* puede contener subcadenas de la forma `\cifra`, que serán reemplazadas por el texto que coincide con la subcadena entre paréntesis que ocupa el lugar indicado por la cifra; `\0` produce el contenido total de la cadena. Se pueden usar hasta nueve subcadenas. Los paréntesis pueden anidarse; en este caso se cuentan los paréntesis de apertura.

Si no se encuentran coincidencias en *string*, se devuelve *string* sin cambios.

Por ejemplo, el siguiente fragmento de código imprime "This was a test"tres veces:

Ejemplo 1. **ereg_replace()** example

```
$string = "This is a test";
echo ereg_replace( " is", " was", $string );
echo ereg_replace( "( )is", "\1was", $string );
echo ereg_replace( "(( )is)", "\2was", $string );
```

Ver también **ereg()**, **eregi()**, y **eregi_replace()**.

eregi (PHP 3, PHP 4)

coincidencia de expresiones regulares sin diferenciar mayúsculas y minúsculas

```
int eregi (string pattern, string string [, array regs])
```

Esta función es idéntica a **ereg()**, excepto en que ignora la distinción entre mayúsculas y minúsculas.

Ver también **ereg()**, **ereg_replace()**, y **eregi_replace()**.

eregi_replace (PHP 3, PHP 4)

reemplaza expresiones regularse sin diferenciar mayúsculas y minúsculas

```
string eregi_replace (string pattern, string replacement, string string)
```

Esta función es idéntica a **ereg_replace()**, excepto en que ignora la distinción entre mayúsculas y minúsculas.

Ver también **ereg()**, **eregi()**, y **ereg_replace()**.

split (PHP 3, PHP 4)

divide la cadena en elementos de un array según una expresión regular

```
array split (string pattern, string string [, int limit])
```

Devuelve un array de cadenas, cada una de las cuales es una subcadena de *string* formada al dividir esta en los límites formados por la expresión regular *pattern*. Si ocurre un error, devuelve un valor falso.

Para obtener los cinco primeros campos de una línea de /etc/passwd:

Ejemplo 1. split() example

```
$passwd_list = split( ":", $passwd_line, 5 );
```

Para examinar una fecha que puede estar delimitada por barras, puntos o guiones:

Ejemplo 2. split() example

```
$date = "04/30/1973"; // Los delimitadores pueden ser barras, puntos o guiones
list( $month, $day, $year ) = split( '[/.-]', $date );
echo "Month: $month; Day: $day; Year: $year<br>\n";
```

Observar que *pattern* distingue entre mayúsculas y minúsculas.

Observar que si no se necesita la potencia de las expresiones regulares, es más rápido utilizar **explode()**, que no carga el motor de expresiones regulares.

Por favor, observar que *pattern* es una expresión regular. Si se quiere dividir con alguno de los caracteres especiales de las expresiones regulares, se necesita protegerlo antes. Si pareciera que **split()** (o cualquier otra función de regex) está haciendo algo irregular, léase el archivo *regex.7*, incluido en el subdirectorio *regex* de la distribución de PHP. Está en formato de página de manual, por lo que para leerlo es necesaria una orden como **man /usr/local/src/regex/regex.7**.

Ver también: **explode()** e **implode()**.

sql_regcase (PHP 3, PHP 4)

construye una expresión regular para buscar coincidencias sin diferenciar mayúsculas y minúsculas

```
string sql_regcase (string string)
```

Devuelve una expresión regular válida que coincide con *string* sin distinguir mayúsculas y minúsculas. Esta expresión es *string* con cada carácter convertido a una expresión entre corchetes que contiene el carácter en mayúscula y minúscula, si es posible; en caso contrario, contiene el carácter original dos veces.

Ejemplo 1. **sql_regcase()** example

```
echo sql_regcase( "Foo bar" );
imprime
[FF][Oo][Oo][ ][Bb][Aa][Rr]
```

Se puede utilizar para lograr coincidencias que no diferencien mayúsculas de minúsculas en productos que sólo soportan expresiones regulares que sí distinguen.

LXV. Satellite CORBA client extension

The Satellite extension is used for accessing CORBA objects. You will need to set the idl_directory= entry in php.ini to a path where you store all IDL files you use.

OrbitObject (unknown)

Access CORBA objects

```
new OrbitObject (string ior)
```

This class provides access to a CORBA object. The *ior* parameter should be a string containing the IOR (Interoperable Object Reference) that identifies the remote object.

Ejemplo 1. Sample IDL file

```
interface MyInterface {
    void SetInfo (string info);
    string GetInfo();

    attribute int value;
}
```

Ejemplo 2. PHP code for accessing MyInterface

```
<?php
$obj = new OrbitObject ($ior);

$obj->SetInfo ("A 2GooD object");

echo $obj->GetInfo();

$obj->value = 42;

echo $obj->value;
?>
```

OrbitEnum (unknown)

Use CORBA enums

```
new OrbitEnum (string id)
```

This class represents the enumeration identified with the *id* parameter. The *id* can be either the name of the enumeration (e.g "MyEnum"), or the full repository id (e.g. "IDL:MyEnum:1.0").

Ejemplo 1. Sample IDL file

```
enum MyEnum {
    a,b,c,d,e
};
```

Ejemplo 2. PHP code for accessing MyEnum

```
<?php
$enum = new OrbitEnum ("MyEnum");

echo $enum->a; /* write 0 */
echo $enum->c; /* write 2 */
echo $enum->e; /* write 4 */
?>
```

OrbitStruct (unknown)

Use CORBA structs

```
new OrbitStruct (string id)
```

This class represents the structure identified with the *id* parameter. The *id* can be either the name of the struct (e.g "MyStruct"), or the full repository id (e.g. "IDL:MyStruct:1.0").

Ejemplo 1. Sample IDL file

```
struct MyStruct {
    short shortvalue;
    string stringvalue;
};

interface SomeInterface {
    void SetValues (MyStruct values);
    MyStruct GetValues();
}
```

Ejemplo 2. PHP code for accessing MyStruct

```
<?php
$obj = new OrbitObject ($ior);

$initial_values = new OrbitStruct ("IDL:MyStruct:1.0");
$initial_values->shortvalue = 42;
$initial_values->stringvalue = "HGTIG";

$obj->SetValues ($initial_values);

$values = $obj->GetValues();

echo $values->shortvalue;
echo $values->stringvalue;
?>
```

satellite_caught_exception (PHP 4 >= 4.0.3)

See if an exception was caught from the previous function

```
bool satellite_caught_exception ()
```

This function returns true if an exception has been caught.

Ejemplo 1. Sample IDL file

```
/* +++++++ Out of Cheese Error. Redo From Start. */
exception OutOfCheeseError {
    int parameter;
}

interface AnotherInterface {
    void AskWhy() raises (OutOfCheeseError);
}
```

Ejemplo 2. PHP code for handling CORBA exceptions

```
<?php
$obj = new OrbitObject ($ior);

$obj->AskWhy();

if (satellite_caught_exception()) {
    if ("IDL:OutOfCheeseError:1.0" == satellite_exception_id()) {
        $exception = satellite_exception_value();
        echo $exception->parameter;
    }
}
?>
```

satellite_exception_id (PHP 4 >= 4.0.3)

Get the repository id for the latest exception.

```
string satellite_exception_id ()
```

Return a repository id string. (E.g. "IDL:MyException:1.0".) For example usage see **satellite_caught_exception()**.

satellite_exception_value (PHP 4 >= 4.0.3)

Get the exception struct for the latest exception

```
OrbitStruct satellite_exception_value ()
```

Return an exception struct. For example usage see **satellite_caught_exception()**.

LXVI. Funciones Semáforo y de memoria compartida

Este módulo provee funciones semáforo utilizando los semáforos de System V. Los semáforos pueden usarse para obtener acceso exclusivo a algún recurso del ordenador en cuestión, o para limitar el número de procesos que pueden usar un recurso simultáneamente.

Este módulo provee tambien funciones de memoria compartida, usando el compartimiento de memoria de System V. La memoria compartida puede usarse para proveer acceso a variables globales. Los diferentes demonios http e incluso otros programas, (como Perl, C, ...) son capaces de utilizar estos datos, para intercambiarlos de modo global. Recuerde que, la memoria compartida NO es segura para los accesos simultáneos. Use los semáforos para obtener sincronismo.

Tabla 1. Limites de la memoria compartida del SO Unix

SHMMAX	máximo tamaño de memoria compartida, normalmente 131072 bytes
SHMMIN	minimo tamaño de memoria compartida, por lo general 1 byte
SHMMNI	máxima cantidad de segmentos de memoria compartida, normalmente 100
SHMSEG	máximo de memoria compartida por proceso, normalmente 6

sem_get (PHP 3>= 3.0.6, PHP 4)

obtiene la identificacion de un semáforo (semaphore id)

```
int sem_get (int key [, int max_acquire [, int perm]])
```

Devuelve: Un identificador positivo de semáforo en caso de éxito, o falso en caso de error.

sem_get() Devuelve un id que puede usarse para acceder al semáforo de System V con la llave dada. El semáforo se usa si es necesario usando los bits de permisos especificados en perm (por defecto 0666). El número de procesos que pueden adquirir el semáforo simultáneamente, se define en max_acquire (por defecto es 1). Actualmente este valor se fija sólo si el proceso determina que es el único relacionado actualmente al semáforo.

Una segunda llamada a **sem_get()** para la misma llave, devolverá un id de semáforo diferente, pero con ambos identificados, se accederá al mismo semáforo.

Véase también: **sem_acquire()** y **sem_release()**.

sem_acquire (PHP 3>= 3.0.6, PHP 4)

adquiere un semáforo, lo toma para sí

```
int sem_acquire (int sem_identifier)
```

Devuelve: Verdadero si hay éxito, falso si hay errores

sem_acquire() Produce un bloqueo (de ser necesario) hasta que el semáforo puede adquirirse. Un proceso intentando adquirir un semáforo ya adquirido, se bloqueará permanentemente si adquiriendo el semáforo, excede su valor de max_acquire.

Despues de procesar una petición, cualquier semáforo adquirido por un proceso, que no sea explícitamente liberado, será liberado automáticamente, generando un mensaje de alerta.

Véase tambien: **sem_get()** and **sem_release()**.

sem_release (PHP 3>= 3.0.6, PHP 4)

release a semaphore

```
int sem_release (int sem_identifier)
```

Returns: true on success, false on error

sem_release() releases the semaphore if it is currently acquired by the calling process, otherwise a warning is generated.

After releasing the semaphore, **sem_acquire()** may be called to re-acquire it.

Véase tambien: **sem_get()** y **sem_acquire()**.

shm_attach (PHP 3>= 3.0.6, PHP 4)

Crea o abre un segmento de memoria compartida

```
int shm_attach (int key [, int memsize [, int perm]])
```

shm_attach() devuelve un identificador que puede usarse para acceder a la memoria compartida de SystemV, con la llave dada, la primera llamada creará el segmento de memoria compartida con `mem_size` de tamaño (por defecto: `sysvshm.init_mem` en el [archivo de configuración](#), o bien de 10000 bytes) y los bits de permiso mas apropiados (por defecto: 0666).

Una segunda llamada a **shm_attach()** con la misma `llave` devolverá un id diferente de memoria compartida, pero ambos identificadores acceden a la misma porción de memoria compartida. `memsize` y `perm` serán ignorados.

shm_detach (PHP 3>= 3.0.6, PHP 4)

Finaliza conexión con un segmento de memoria compartida

```
int shm_detach (int shm_identifier)
```

shm_detach() finaliza la conexión con la memoria compartida, especificada por el identificador `shm_identifier` creado con **shm_attach()**. Hay que recordar que la memoria compartida aún existe en el sistema Unix, y los datos aún están presentes.

shm_remove (PHP 3>= 3.0.6, PHP 4)

Elimina memoria compartida del sistema Unix

```
int shm_remove (int shm_identifier)
```

Elimina la memoria compartida de un sistema Unix, Todos los datos serán destruidos.

shm_put_var (PHP 3>= 3.0.6, PHP 4)

Inserta o actualiza una variable en la memoria compartida

```
int shm_put_var (int shm_identifier, int variable_key, mixed variable)
```

Inserta o actualiza una `variable` con una llave `variable_key`. Son válidos todos los tipos de variable (doble, entera, cadena, arreglo).

shm_get_var (PHP 3>= 3.0.6, PHP 4)

Devuelve una variable de la memoria compartida

```
mixed shm_get_var (int id, int variable_key)
```

shm_get_var() devuelve la variable con la llave `variable_key` dada. La variable, queda presente en la memoria compartida.

shm_remove_var (PHP 3>= 3.0.6, PHP 4)

Elimina una variable de la memoria compartida

```
int shm_remove_var (int id, int variable_key)
```

Elimina la variable que se corresponde con la llave *variable_key* dada, liberando la memoria que ocupaba aquella.

LXVII. SESAM database functions

SESAM/SQL-Server is a mainframe database system, developed by Fujitsu Siemens Computers, Germany. It runs on high-end mainframe servers using the operating system BS2000/OSD.

In numerous productive BS2000 installations, SESAM/SQL-Server has proven ...

- the ease of use of Java-, Web- and client/server connectivity,
- the capability to work with an availability of more than 99.99%,
- the ability to manage tens and even hundreds of thousands of users.

Now there is a PHP3 SESAM interface available which allows database operations via PHP-scripts.

Configuration notes: There is no standalone support for the PHP SESAM interface, it works only as an integrated Apache module. In the Apache PHP module, this [SESAM interface is configured](#) using Apache directives.

Tabla 1. SESAM Configuration directives

Directive	Meaning
php3_sesam_oml	Name of BS2000 PLAM library containing the loadable SESAM driver modules. Required for using SESAM functions. Example: <code>php3_sesam_oml \$.SYSLNK.SESAM-SQL.030</code>
php3_sesam_configfile	Name of SESAM application configuration file. Required for using SESAM functions. Example: <code>php3_sesam_configfile \$SESAM.SESAM.CONF.AW</code> It will usually contain a configuration like (see SESAM reference manual): <code>CNF=B NAM=K NOTYPE</code>
php3_sesam_messagecatalog	Name of SESAM message catalog file. In most cases, this directive is not necessary. Only if the SESAM message file is not installed in the system's BS2000 message file table, it can be set with this directive. Example: <code>php3_sesam_messagecatalog \$.SYSMES.SESAM-SQL.030</code>

In addition to the configuration of the PHP/SESAM interface, you have to configure the SESAM-Database server itself on your mainframe as usual. That means:

- starting the SESAM database handler (DBH), and
- connecting the databases with the SESAM database handler

To get a connection between a PHP script and the database handler, the `CNF` and `NAM` parameters of the selected SESAM configuration file must match the id of the started database handler.

In case of distributed databases you have to start a SESAM/SQL-DCN agent with the distribution table including the host and database names.

The communication between PHP (running in the POSIX subsystem) and the database handler (running outside the POSIX subsystem) is realized by a special driver module called SQLSCI and SESAM connection modules using

common memory. Because of the common memory access, and because PHP is a static part of the web server, database accesses are very fast, as they do not require remote accesses via ODBC, JDBC or UTM.

Only a small stub loader (SESMOD) is linked with PHP, and the SESAM connection modules are pulled in from SESAM's OML PLAM library. In the [configuration](#), you must tell PHP the name of this PLAM library, and the file link to use for the SESAM configuration file (As of SESAM V3.0, SQLSCI is available in the SESAM Tool Library, which is part of the standard distribution).

Because the SQL command quoting for single quotes uses duplicated single quotes (as opposed to a single quote preceded by a backslash, used in some other databases), it is advisable to set the PHP configuration directives `php3_magic_quotes_gpc` and `php3_magic_quotes_sybase` to `on` for all PHP scripts using the SESAM interface.

Runtime considerations: Because of limitations of the BS2000 process model, the driver can be loaded only after the Apache server has forked off its server child processes. This will slightly slow down the initial SESAM request of each child, but subsequent accesses will respond at full speed.

When explicitly defining a Message Catalog for SESAM, that catalog will be loaded each time the driver is loaded (i.e., at the initial SESAM request). The BS2000 operating system prints a message after successful load of the message catalog, which will be sent to Apache's `error_log` file. BS2000 currently does not allow suppression of this message, it will slowly fill up the log.

Make sure that the SESAM OML PLAM library and SESAM configuration file are readable by the user id running the web server. Otherwise, the server will be unable to load the driver, and will not allow to call any SESAM functions. Also, access to the database must be granted to the user id under which the Apache server is running. Otherwise, connections to the SESAM database handler will fail.

Cursor Types: The result cursors which are allocated for SQL "select type" queries can be either "sequential" or "scrollable". Because of the larger memory overhead needed by "scrollable" cursors, the default is "sequential".

When using "scrollable" cursors, the cursor can be freely positioned on the result set. For each "scrollable" query, there are global default values for the scrolling type (initialized to: `SESAM_SEEK_NEXT`) and the scrolling offset which can either be set once by `sesam_seek_row()` or each time when fetching a row using `sesam_fetch_row()`. When fetching a row using a "scrollable" cursor, the following post-processing is done for the global default values for the scrolling type and scrolling offset:

Tabla 2. Scrolled Cursor Post-Processing

Scroll Type	Action
<code>SESAM_SEEK_NEXT</code>	none
<code>SESAM_SEEK_PRIOR</code>	none
<code>SESAM_SEEK_FIRST</code>	set scroll type to <code>SESAM_SEEK_NEXT</code>
<code>SESAM_SEEK_LAST</code>	set scroll type to <code>SESAM_SEEK_PRIOR</code>
<code>SESAM_SEEK_ABSOLUTE</code>	Auto-Increment internal offset value
<code>SESAM_SEEK_RELATIVE</code>	none. (maintain global default <code>offset</code> value, which allows for, e.g., fetching each 10th row backwards)

Porting note: Because in the PHP world it is natural to start indexes at zero (rather than 1), some adaptions have been made to the SESAM interface: whenever an indexed array is starting with index 1 in the native SESAM interface, the PHP interface uses index 0 as a starting point. E.g., when retrieving columns with `sesam_fetch_row()`, the first column has the index 0, and the subsequent columns have indexes up to (but not including) the column count (`$array["count"]`). When porting SESAM applications from other high level languages to PHP, be aware of this changed interface. Where appropriate, the description of the respective php sesam functions include a note that the index is zero based.

Security concerns: When allowing access to the SESAM databases, the web server user should only have as little

privileges as possible. For most databases, only read access privilege should be granted. Depending on your usage scenario, add more access rights as you see fit. Never allow full control to any database for any user from the 'net! Restrict access to php scripts which must administer the database by using password control and/or SSL security.

Migration from other SQL databases: No two SQL dialects are ever 100% compatible. When porting SQL applications from other database interfaces to SESAM, some adaption may be required. The following typical differences should be noted:

- Vendor specific data types

Some vendor specific data types may have to be replaced by standard SQL data types (e.g., TEXT could be replaced by VARCHAR(max. size)).

- Keywords as SQL identifiers

In SESAM (as in standard SQL), such identifiers must be enclosed in double quotes (or renamed).

- Display length in data types

SESAM data types have a precision, not a display length. Instead of int(4) (intended use: integers up to '9999'), SESAM requires simply int for an implied size of 31 bits. Also, the only datetime data types available in SESAM are: DATE, TIME(3) and TIMESTAMP(3).

- SQL types with vendor-specific unsigned, zerofill, or auto_increment attributes

Unsigned and zerofill are not supported. Auto_increment is automatic (use "INSERT ... VALUES(*, ...)" instead of "... VALUES(0, ...)" to take advantage of SESAM-implied auto-increment.

- int ... DEFAULT '0000'

Numeric variables must not be initialized with string constants. Use DEFAULT 0 instead. To initialize variables of the datetime SQL data types, the initialization string must be prefixed with the respective type keyword, as in: CREATE TABLE exmpl (xtime timestamp(3) DEFAULT TIMESTAMP '1970-01-01 00:00:00.000' NOT NULL);

- \$count = xxxx_num_rows();

Some databases promise to guess/estimate the number of the rows in a query result, even though the returned value is grossly incorrect. SESAM does not know the number of rows in a query result before actually fetching them. If you REALLY need the count, try SELECT COUNT(...) WHERE ..., it will tell you the number of hits. A second query will (hopefully) return the results.

- DROP TABLE thename;

In SESAM, in the **DROP TABLE** command, the table name must be either followed by the keyword RESTRICT or CASCADE. When specifying RESTRICT, an error is returned if there are dependent objects (e.g., VIEWS), while with CASCADE, dependent objects will be deleted along with the specified table.

Notes on the use of various SQL types: SESAM does not currently support the BLOB type. A future version of SESAM will have support for BLOB.

At the PHP interface, the following type conversions are automatically applied when retrieving SQL fields:

Tabla 3. SQL to PHP Type Conversions

SQL Type	PHP Type
SMALLINT, INTEGER	"integer"
NUMERIC, DECIMAL, FLOAT, REAL, DOUBLE	"double"
DATE, TIME, TIMESTAMP	"string"
VARCHAR, CHARACTER	"string"

When retrieving a complete row, the result is returned as an array. Empty fields are not filled in, so you will have to check for the existence of the individual fields yourself (use `isset()` or `empty()` to test for empty fields). That allows more user control over the appearance of empty fields (than in the case of an empty string as the representation of an empty field).

Support of SESAM's "multiple fields" feature: The special "multiple fields" feature of SESAM allows a column to consist of an array of fields. Such a "multiple field" column can be created like this:

Ejemplo 1. Creating a "multiple field" column

```
CREATE TABLE multi_field_test
(
    pkey CHAR(20) PRIMARY KEY,
    multi(3) CHAR(12)
)
```

and can be filled in using:

Ejemplo 2. Filling a "multiple field" column

```
INSERT INTO multi_field_test ( pkey, multi(2..3) )
VALUES ( 'Second', <'first_val','second_val'>)
```

Note that (like in this case) leading empty sub-fields are ignored, and the filled-in values are collapsed, so that in the above example the result will appear as multi(1..2) instead of multi(2..3).

When retrieving a result row, "multiple columns" are accessed like "inlined" additional columns. In the example above, "pkey" will have the index 0, and the three "multi(1..3)" columns will be accessible as indices 1 through 3.

For specific SESAM details, please refer to the SESAM/SQL-Server documentation (english) (http://its.siemens.de/lobs/its/techinf/oltp/sesam/manuals/index_en.htm) or the SESAM/SQL-Server documentation (german) (http://its.siemens.de/lobs/its/techinf/oltp/sesam/manuals/index_gr.htm), both available online, or use the respective manuals.

sesam_connect (PHP 3 CVS only)

Open SESAM database connection

```
boolean sesam_connect (string catalog, string schema, string user)
```

Returns TRUE if a connection to the SESAM database was made, or FALSE on error.

sesam_connect() establishes a connection to an SESAM database handler task. The connection is always "persistant" in the sense that only the very first invocation will actually load the driver from the configured SESAM OML PLAM library. Subsequent calls will reuse the driver and will immediately use the given catalog, schema, and user.

When creating a database, the "*catalog*" name is specified in the SESAM configuration directive
//ADD-SQL-DATABASE-CATALOG-LIST ENTRY-1 = *CATALOG(CATALOG-NAME = catalogname,...)

The "*schema*" references the desired database schema (see SESAM handbook).

The "*user*" argument references one of the users which are allowed to access this "*catalog*" / "*schema*" combination. Note that "*user*" is completely independent from both the system's user id's and from HTTP user/password protection. It appears in the SESAM configuration only.

See also **sesam_disconnect()**.

Ejemplo 1. Connect to a SESAM database

```
<?php
if (! sesam_connect ("mycatalog", "myschema", "otto"))
    die("Unable to connect to SESAM";
?>
```

sesam_disconnect (PHP 3 CVS only)

Detach from SESAM connection

```
boolean sesam_disconnect(void);
```

Returns: always TRUE.

sesam_disconnect() closes the logical link to a SESAM database (without actually disconnecting and unloading the driver).

Note that this isn't usually necessary, as the open connection is automatically closed at the end of the script's execution. Uncommitted data will be discarded, because an implicit **sesam_rollback()** is executed.

sesam_disconnect() will not close the persistent link, it will only invalidate the currently defined "*catalog*", "*schema*" and "*user*" triple, so that any sesam function called after **sesam_disconnect()** will fail.

See also: **sesam_connect()**.

Ejemplo 1. Closing a SESAM connection

```
if (sesam_connect ("mycatalog", "myschema", "otto")) {
... some queries and stuff ...
sesam_disconnect(); }
```

sesam_settransaction (PHP 3 CVS only)

Set SESAM transaction parameters

```
boolean sesam_settransaction (int isolation_level, int read_only)
```

Returns: TRUE if the values are valid, and the **settransaction()** operation was successful, FALSE otherwise.

sesam_settransaction() overrides the default values for the "isolation level" and "read-only" transaction parameters (which are set in the SESAM configuration file), in order to optimize subsequent queries and guarantee database consistency. The overridden values are used for the next transaction only.

sesam_settransaction() can only be called before starting a transaction, not after the transaction has been started already.

To simplify the use in php scripts, the following constants have been predefined in php (see SESAM handbook for detailed explanation of the semantics):

Tabla 1. Valid values for "*Isolation_Level*" parameter

Value	Constant	Meaning
1	SESAM_TXISOL_READ_UNCOMMITTED	Read Uncommitted
2	SESAM_TXISOL_READ_COMMITTED	Read Committed
3	SESAM_TXISOL_REPEATABLE_READ	Repeatable Read
4	SESAM_TXISOL_SERIALIZABLE	Serializable

Tabla 2. Valid values for "*Read_Only*" parameter

Value	Constant	Meaning
0	SESAM_TXREAD_READWRITE	Read/Write
1	SESAM_TXREAD_READONLY	Read-Only

The values set by **sesam_settransaction()** will override the default setting specified in the [SESAM configuration file](#).

Ejemplo 1. Setting SESAM transaction parameters

```
<?php
sesam_settransaction(SESAM_TXISOL_REPEATABLE_READ,
                      SESAM_TXREAD_READONLY);
?>
```

sesam_commit (PHP 3 CVS only)

Commit pending updates to the SESAM database

```
boolean sesam_commit(void);
```

Returns: TRUE on success, FALSE on errors

sesam_commit() commits any pending updates to the database.

Note that there is no "auto-commit" feature as in other databases, as it could lead to accidental data loss. Uncommitted data at the end of the current script (or when calling **sesam_disconnect()**) will be discarded by an implied **sesam_rollback()** call.

See also: **sesam_rollback()**.

Ejemplo 1. Committing an update to the SESAM database

```
<?php
if (sesam_connect ("mycatalog", "myschema", "otto")) {
    if (!sesam_execimm("INSERT INTO mytable VALUES (*, 'Small Test', <0, 8, 15>)"))
        die("insert failed");
    if (!sesam_commit())
        die("commit failed");
}
?>
```

sesam_rollback (PHP 3 CVS only)

Discard any pending updates to the SESAM database

```
boolean sesam_rollback(void);
```

Returns: TRUE on success, FALSE on errors

sesam_rollback() discards any pending updates to the database. Also affected are result cursors and result descriptors.

At the end of each script, and as part of the **sesam_disconnect()** function, an implied **sesam_rollback()** is executed, discarding any pending changes to the database.

See also: **sesam_commit()**.

Ejemplo 1. Discarding an update to the SESAM database

```
<?php
if (sesam_connect ("mycatalog", "myschema", "otto")) {
    if (sesam_execimm("INSERT INTO mytable VALUES (*, 'Small Test', <0, 8, 15>)")
        && sesam_execimm("INSERT INTO othertable VALUES (*, 'Another Test', 1)"))
        sesam_commit();
    else
        sesam_rollback();
}
?>
```

sesam_execimm (PHP 3 CVS only)

Execute an "immediate" SQL-statement

```
string sesam_execimm (string query)
```

Returns: A SESAM "result identifier" on success, or FALSE on error.

sesam_execimm() executes an "immediate" statement (i.e., a statement like UPDATE, INSERT or DELETE which returns no result, and has no INPUT or OUTPUT variables). "select type" queries can not be used with **sesam_execimm()**. Sets the *affected_rows* value for retrieval by the **sesam_affected_rows()** function.

Note that **sesam_query()** can handle both "immediate" and "select-type" queries. Use **sesam_execimm()** only if you know beforehand what type of statement will be executed. An attempt to use SELECT type queries with **sesam_execimm()** will return \$err["sqlstate"] == "42SBW".

The returned "result identifier" can not be used for retrieving anything but the **sesam_affected_rows()**; it is only returned for symmetry with the **sesam_query()** function.

```
$stmt = "INSERT INTO mytable VALUES('one', 'two')";
$result = sesam_execimm ($stmt);
$err = sesam_diagnostic();
print("sqlstate = ".$err["sqlstate"]."\n".
      "Affected rows = ".$err["rowcount"]." == ".
      sesam_affected_rows($result)."\n");
```

See also: **sesam_query()** and **sesam_affected_rows()**.

sesam_query (PHP 3 CVS only)

Perform a SESAM SQL query and prepare the result

```
string sesam_query (string query [, boolean scrollable])
```

Returns: A SESAM "result identifier" on success, or FALSE on error.

A "result_id" resource is used by other functions to retrieve the query results.

sesam_query() sends a query to the currently active database on the server. It can execute both "immediate" SQL statements and "select type" queries. If an "immediate" statement is executed, then no cursor is allocated, and any subsequent **sesam_fetch_row()** or **sesam_fetch_result()** call will return an empty result (zero columns, indicating end-of-result). For "select type" statements, a result descriptor and a (scrollable or sequential, depending on the optional boolean *scrollable* parameter) cursor will be allocated. If *scrollable* is omitted, the cursor will be sequential.

When using "scrollable" cursors, the cursor can be freely positioned on the result set. For each "scrollable" query, there are global default values for the scrolling type (initialized to: SESAM_SEEK_NEXT) and the scrolling offset which can either be set once by **sesam_seek_row()** or each time when fetching a row using **sesam_fetch_row()**.

For "immediate" statements, the number of affected rows is saved for retrieval by the **sesam_affected_rows()** function.

See also: **sesam_fetch_row()** and **sesam_fetch_result()**.

Ejemplo 1. Show all rows of the "phone" table as a html table

```
<?php
if (!sesam_connect("phonedb", "demo", "otto"))
    die("cannot connect");
$result = sesam_query("select * from phone");
if (!$result) {
    $err = sesam_diagnostic();
    die($err["errmsg"]);
}
echo "<TABLE BORDER>\n";
// Add title header with column names above the result:
if ($cols = sesam_field_array($result)) {
```

```

echo " <TR><TH COLSPAN=". $cols["count"] .">Result:</TH></TR>\n";
echo " <TR>\n";
for ($col = 0; $col < $cols["count"]; ++$col) {
    $colattr = $cols[$col];
    /* Span the table head over SESAM's "Multiple Fields": */
    if ($colattr["count"] > 1) {
        echo " <TH COLSPAN=". $colattr["count"] .">". $colattr["name"] .
            "(1...". $colattr["count"] .")</TH>\n";
        $col += $colattr["count"] - 1;
    }
    else
        echo " <TH> ". $colattr["name"] . "</TH>\n";
}
echo " </TR>\n";
}

do {
    // Fetch the result in chunks of 100 rows max.
    $ok = sesam_fetch_result($result,100);
    for ($row=0; $row < $ok["rows"]; ++$row) {
        echo " <TR>\n";
        for ($col = 0; $col < $ok["cols"]; ++$col) {
            if (isset($ok[$col][$row]))
                echo " <TD>". $ok[$col][$row] . "</TD>\n";
            else
                echo " <TD>-empty-</TD>\n";
        }
        echo " </TR>\n";
    }
} while ($ok["truncated"]); // while there may be more data
echo "</TABLE>\n";
// free result id
sesam_free_result($result);
?>

```

sesam_num_fields (PHP 3 CVS only)

Return the number of fields/columns in a result set

```
int sesam_num_fields (string result_id)
```

After calling **sesam_query()** with a "select type" query, this function gives you the number of columns in the result. Returns an integer describing the total number of columns (aka. fields) in the current *result_id* result set or FALSE on error.

For "immediate" statements, the value zero is returned. The SESAM "multiple field" columns count as their respective dimension, i.e., a three-column "multiple field" counts as three columns.

See also: **sesam_query()** and **sesam_field_array()** for a way to distinguish between "multiple field" columns and regular columns.

sesam_field_name (PHP 3 CVS only)

Return one column name of the result set

```
int sesam_field_name (string result_id, int index)
```

Returns the name of a field (i.e., the column name) in the result set, or FALSE on error.

For "immediate"queries, or for dynamic columns, an empty string is returned.

Nota: The column index is zero-based, not one-based as in SESAM.

See also: **sesam_field_array()**. It provides an easier interface to access the column names and types, and allows for detection of "multiple fields".

sesam_diagnostic (PHP 3 CVS only)

Return status information for last SESAM call

```
array sesam_diagnostic(void);
```

Returns an associative array of status and return codes for the last SQL query/statement/command. Elements of the array are:

Tabla 1. Status information returned by sesam_diagnostic()

Element	Contents
\$array["sqlstate"]	5 digit SQL return code (see the SESAM manual for the description of the possible values of SQLSTATE)
\$array["rowcount"]	number of affected rows in last update/insert/delete (set after "immediate"statements only)
\$array["errmsg"]	"human readable"error message string (set after errors only)
\$array["errcol"]	error column number of previous error (0-based; or -1 if undefined. Set after errors only)
\$array["errlin"]	error line number of previous error (0-based; or -1 if undefined. Set after errors only)

In the following example, a syntax error (E SEW42AE ILLEGAL CHARACTER) is displayed by including the offending SQL statement and pointing to the error location:

Ejemplo 1. Displaying SESAM error messages with error position

```
<?php
// Function which prints a formatted error message,
// displaying a pointer to the syntax error in the
// SQL statement
function PrintReturncode($exec_str)
{
    $err = Sesam_Diagnostic();
    $colspan=4; // 4 cols for: sqlstate, errlin, errcol, rowcount
    if ($err["errlin"] == -1)
        -$colspan;
    if ($err["errcol"] == -1)
        -$colspan;
    if ($err["rowcount"] == 0)
```

```

-$colspan;
echo "<TABLE BORDER>\n";
echo "<TR><TH COLSPAN=". $colspan ."><FONT COLOR=red>ERROR:</FONT> ".
htmlspecialchars($err["errormsg"]) . "</TH></TR>\n";
if ($err["errcol"] >= 0) {
    echo "<TR><TD COLSPAN=". $colspan ."><PRE>\n";
    $errstmt = $exec_str . "\n";
    for ($lin=0; $errstmt != ""; ++$lin) {
        if ($lin != $err["errlin"]) { // $lin is less or greater than errlin
            if (! ($i = strchr($errstmt, "\n")))
                $i = "";
            $line = substr($errstmt, 0, strlen($errstmt)-strlen($i)+1);
            $errstmt = substr($i, 1);
            if ($line != "\n")
                print htmlspecialchars($line);
        }
    }
    else {
        if (! ($i = strchr($errstmt, "\n")))
            $i = "";
        $line = substr($errstmt, 0, strlen($errstmt)-strlen($i)+1);
        $errstmt = substr($i, 1);
        for ($col=0; $col < $err["errcol"]; ++$col)
            echo (substr($line, $col, 1) == "\t") ? "\t" : ".";
        echo "<FONT COLOR=RED><BLINK>\\"</BLINK></FONT>\n";
        print "<FONT COLOR=\">#880000\>".htmlspecialchars($line). "</FONT> ";
        for ($col=0; $col < $err["errcol"]; ++$col)
            echo (substr($line, $col, 1) == "\t") ? "\t" : ".";
        echo "<FONT COLOR=RED><BLINK>/</BLINK></FONT>\n";
    }
}
echo "</PRE></TD></TR>\n";
}
echo "<TR>\n";
echo " <TD>sqlstate=" . $err["sqlstate"] . "</TD>\n";
if ($err["errlin"] != -1)
    echo " <TD>errlin=" . $err["errlin"] . "</TD>\n";
if ($err["errcol"] != -1)
    echo " <TD>errcol=" . $err["errcol"] . "</TD>\n";
if ($err["rowcount"] != 0)
    echo " <TD>rowcount=" . $err["rowcount"] . "</TD>\n";
echo "</TR>\n";
echo "</TABLE>\n";
}

if (!sesam_connect("mycatalog", "phoneno", "otto"))
    die("cannot connect");

$stmt = "SELECT * FROM phone\n".
    " WHERE@ LASTNAME='KRAEMER'\n".
    " ORDER BY FIRSTNAME";
if (!$result = sesam_query($stmt))
    PrintReturncode($stmt);
?>

```

See also: **sesam_errormsg()** for simple access to the error string only

sesam_fetch_result (PHP 3 CVS only)

Return all or part of a query result

```
mixed sesam_fetch_result (string result_id [, int max_rows])
```

Returns a mixed array with the query result entries, optionally limited to a maximum of *max_rows* rows. Note that both row and column indexes are zero-based.

Tabla 1. Mixed result set returned by sesam_fetch_result()

Array Element	Contents
int \$arr["count"]	number of columns in result set (or zero if this was an "immediate"query)
int \$arr["rows"]	number of rows in result set (between zero and <i>max_rows</i>)
boolean \$arr["truncated"]	TRUE if the number of rows was at least <i>max_rows</i> , FALSE otherwise. Note that even when this is TRUE, the next sesam_fetch_result() call may return zero rows because there are no more result entries.
mixed \$arr[col][row]	result data for all the fields at row(<i>row</i>) and column(<i>col</i>), (where the integer index <i>row</i> is between 0 and \$arr["rows"]-1, and <i>col</i> is between 0 and \$arr["count"]-1). Fields may be empty, so you must check for the existence of a field by using the php isset() function. The type of the returned fields depend on the respective SQL type declared for its column (see SESAM overview for the conversions applied). SESAM "multiple fields"are "inlined"and treated like a sequence of columns.

Note that the amount of memory used up by a large query may be gigantic. Use the *max_rows* parameter to limit the maximum number of rows returned, unless you are absolutely sure that your result will not use up all available memory.

See also: **sesam_fetch_row()**, and **sesam_field_array()** to check for "multiple fields". See the description of the **sesam_query()** function for a complete example using **sesam_fetch_result()**.

sesam_affected_rows (PHP 3 CVS only)

Get number of rows affected by an immediate query

```
int sesam_affected_rows (string result_id)
```

result_id is a valid result id returned by **sesam_query()**.

Returns the number of rows affected by a query associated with *result_id*.

The **sesam_affected_rows()** function can only return useful values when used in combination with "immediate"SQL statements (updating operations like **INSERT**, **UPDATE** and **DELETE**) because SESAM does not deliver any "affected rows"information for "select type"queries. The number returned is the number of affected rows.

See also: **sesam_query()** and **sesam_execimm()**

```
$result = sesam_execimm ("DELETE FROM PHONE WHERE LASTNAME = '".strtoupper($name)."'");
if (! $result) {
    ... error ...
}
print sesam_affected_rows($result).
    " entries with last name ".$name." deleted.\n"
```

sesam_errormsg (PHP 3 CVS only)

Returns error message of last SESAM call

```
string sesam_errormsg(void);
```

Returns the SESAM error message associated with the most recent SESAM error.

```
if (!sesam_execimm($stmt))
    printf("%s<br>\n", sesam_errormsg());
```

See also: **sesam_diagnostic()** for the full set of SESAM SQL status information

sesam_field_array (PHP 3 CVS only)

Return meta information about individual columns in a result

```
array sesam_field_array (string result_id)
```

result_id is a valid result id returned by **sesam_query()**.

Returns a mixed associative/indexed array with meta information (column name, type, precision, ...) about individual columns of the result after the query associated with *result_id*.

Tabla 1. Mixed result set returned by sesam_field_array()

Array Element	Contents
int \$arr["count"]	Total number of columns in result set (or zero if this was an "immediate"query). SESAM "multiple fields"are "inlined"and treated like the respective number of columns.
string \$arr[col]["name"]	column name for column(col), where col is between 0 and \$arr["count"]-1. The returned value can be the empty string (for dynamically computed columns). SESAM "multiple fields"are "inlined"and treated like the respective number of columns, each with the same column name.

Array Element	Contents
string \$arr[col]["count"]	The "count"attribute describes the repetition factor when the column has been declared as a "multiple field". Usually, the "count"attribute is 1. The first column of a "multiple field"column however contains the number of repetitions (the second and following column of the "multiple field"contain a "count"attribute of 1). This can be used to detect "multiple fields" in the result set. See the example shown in the sesam_query() description for a sample use of the "count"attribute.
string \$arr[col]["type"]	php variable type of the data for column(col), where col is between 0 and \$arr["count"]-1. The returned value can be one of <ul style="list-style-type: none"> • "integer" • "double" • "string" depending on the SQL type of the result. SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same php type.
string \$arr[col]["sqltype"]	SQL variable type of the column data for column(col), where col is between 0 and \$arr["count"]-1. The returned value can be one of <ul style="list-style-type: none"> • "CHARACTER" <ul style="list-style-type: none"> • "VARCHAR" • "NUMERIC" • "DECIMAL" • "INTEGER" • "SMALLINT" • "FLOAT" • "REAL" • "DOUBLE" • "DATE" • "TIME" • "TIMESTAMP" describing the SQL type of the result. SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same SQL type.

Array Element	Contents
string \$arr[col]["length"]	The SQL "length"attribute of the SQL variable in column(col), where col is between 0 and \$arr["count"]-1. The "length"attribute is used with "CHARACTER"and "VARCHAR"SQL types to specify the (maximum) length of the string variable. SESAM "multiple fields"are "inlined"and treated like the respective number of columns, each with the same length attribute.
string \$arr[col]["precision"]	The "precision"attribute of the SQL variable in column(col), where col is between 0 and \$arr["count"]-1. The "precision"attribute is used with numeric and time data types. SESAM "multiple fields"are "inlined"and treated like the respective number of columns, each with the same precision attribute.
string \$arr[col]["scale"]	The "scale"attribute of the SQL variable in column(col), where col is between 0 and \$arr["count"]-1. The "scale"attribute is used with numeric data types. SESAM "multiple fields"are "inlined"and treated like the respective number of columns, each with the same scale attribute.

See the **sesam_query()** function for an example of the **sesam_field_array()** use.

sesam_fetch_row (PHP 3 CVS only)

Fetch one row as an array

```
array sesam_fetch_row (string result_id [, int whence [, int offset]])
```

Returns an array that corresponds to the fetched row, or FALSE if there are no more rows.

The number of columns in the result set is returned in an associative array element \$array["count"]. Because some of the result columns may be empty, the **count()** function can not be used on the result row returned by **sesam_fetch_row()**.

result_id is a valid result id returned by **sesam_query()** (select type queries only!).

whence is an optional parameter for a fetch operation on "scrollable" cursors, which can be set to the following predefined constants:

Tabla 1. Valid values for "whence" parameter

Value	Constant	Meaning
0	SESAM_SEEK_NEXT	read sequentially (after fetch, the internal default is set to SESAM_SEEK_NEXT)
1	SESAM_SEEK_PRIOR	read sequentially backwards (after fetch, the internal default is set to SESAM_SEEK_PRIOR)

Value	Constant	Meaning
2	SESAM_SEEK_FIRST	rewind to first row (after fetch, the default is set to SESAM_SEEK_NEXT)
3	SESAM_SEEK_LAST	seek to last row (after fetch, the default is set to SESAM_SEEK_PRIOR)
4	SESAM_SEEK_ABSOLUTE	seek to absolute row number given as <i>offset</i> (<i>Zero-based. After fetch, the internal default is set to SESAM_SEEK_ABSOLUTE, and the internal offset value is auto-incremented</i>)
5	SESAM_SEEK_RELATIVE	seek relative to current scroll position, where <i>offset</i> can be a positive or negative offset value.

This parameter is only valid for "scrollable" cursors.

When using "scrollable" cursors, the cursor can be freely positioned on the result set. If the *whence* parameter is omitted, the global default values for the scrolling type (initialized to: SESAM_SEEK_NEXT, and settable by **sesam_seek_row()**) are used. If *whence* is supplied, its value replaces the global default.

offset is an optional parameter which is only evaluated (and required) if *whence* is either SESAM_SEEK_RELATIVE or SESAM_SEEK_ABSOLUTE. This parameter is only valid for "scrollable" cursors.

sesam_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array (indexed by values between 0 and \$array["count"]-1). Fields may be empty, so you must check for the existence of a field by using the php **isSet()** function. The type of the returned fields depend on the respective SQL type declared for its column (see [SESAM overview](#) for the conversions applied). SESAM "multiple fields" are "inlined" and treated like a sequence of columns.

Subsequent calls to **sesam_fetch_row()** would return the next (or prior, or n'th next/prior, depending on the scroll attributes) row in the result set, or FALSE if there are no more rows.

Ejemplo 1. SESAM fetch rows

```
<?php
$result = sesam_query ("SELECT * FROM phone\n".
                      " WHERE LASTNAME='".strtoupper($name)."' \n".
                      " ORDER BY FIRSTNAME", 1);
if (! $result) {
    ... error ...
}
// print the table in backward order
print "<TABLE BORDER>\n";
$row = sesam_fetch_row ($result, SESAM_SEEK_LAST);
while (is_array($row)) {
    print " <TR>\n";
    for($col = 0; $col < $row["count"]; ++$col) {
        print "   <TD>".htmlspecialchars($row[$col])."</TD>\n";
    }
    print " </TR>\n";
    // use implied SESAM_SEEK_PRIOR
    $row = sesam_fetch_row ($result);
}
print "</TABLE>\n";
sesam_free_result ($result);
?>
```

See also: **sesam_fetch_array()** which returns an associative array, and **sesam_fetch_result()** which returns many rows per invocation.

sesam_fetch_array (PHP 3 CVS only)

Fetch one row as an associative array

```
array sesam_fetch_array (string result_id [, int whence [, int offset]])
```

Returns an array that corresponds to the fetched row, or FALSE if there are no more rows.

sesam_fetch_array() is an alternative version of **sesam_fetch_row()**. Instead of storing the data in the numeric indices of the result array, it stores the data in associative indices, using the field names as keys.

result_id is a valid result id returned by **sesam_query()** (select type queries only!).

For the valid values of the optional *whence* and *offset* parameters, see the **sesam_fetch_row()** function for details.

sesam_fetch_array() fetches one row of data from the result associated with the specified result identifier. The row is returned as an associative array. Each result column is stored with an associative index equal to its column (aka. field) name. The column names are converted to lower case.

Columns without a field name (e.g., results of arithmetic operations) and empty fields are not stored in the array. Also, if two or more columns of the result have the same column names, the later column will take precedence. In this situation, either call **sesam_fetch_row()** or make an alias for the column.

```
SELECT TBL1.COL AS FOO, TBL2.COL AS BAR FROM TBL1, TBL2
```

A special handling allows fetching "multiple field" columns (which would otherwise all have the same column names). For each column of a "multiple field", the index name is constructed by appending the string "(n)" where n is the sub-index of the multiple field column, ranging from 1 to its declared repetition factor. The indices are NOT zero based, in order to match the nomenclature used in the respective query syntax. For a column declared as:

```
CREATE TABLE ... ( ... MULTI(3) INT )
```

the associative indices used for the individual "multiple field" columns would be "multi(1)", "multi(2)", and "multi(3)" respectively.

Subsequent calls to **sesam_fetch_array()** would return the next (or prior, or n'th next/prior, depending on the scroll attributes) row in the result set, or FALSE if there are no more rows.

Ejemplo 1. SESAM fetch array

```
<?php
$result = sesam_query ("SELECT * FROM phone\n".
                      " WHERE LASTNAME='".strtoupper($name)."'\\n".
                      " ORDER BY FIRSTNAME", 1);
if (! $result) {
    ... error ...
}
// print the table:
print "<TABLE BORDER>\\n";
while (($row = sesam_fetch_array ($result)) && count($row) > 0) {
    print " <TR>\\n";
    print "   <TD>".htmlspecialchars($row["firstname"])."</TD>\\n";
    print "   <TD>".htmlspecialchars($row["lastname"])."</TD>\\n";
    print "   <TD>".htmlspecialchars($row["phoneno"])."</TD>\\n";
    print " </TR>\\n";
}
```

```
print "</TABLE>\n";
sesam_free_result ($result);
?>
```

See also: **sesam_fetch_row()** which returns an indexed array.

sesam_seek_row (PHP 3 CVS only)

Set scrollable cursor mode for subsequent fetches

```
boolean sesam_seek_row (string result_id, int whence [, int offset])
```

result_id is a valid result id (select type queries only, and only if a "scrollable"cursor was requested when calling **sesam_query()**).

whence sets the global default value for the scrolling type, it specifies the scroll type to use in subsequent fetch operations on "scrollable" cursors, which can be set to the following predefined constants:

Tabla 1. Valid values for "whence" parameter

Value	Constant	Meaning
0	SESAM_SEEK_NEXT	read sequentially
1	SESAM_SEEK_PRIOR	read sequentially backwards
2	SESAM_SEEK_FIRST	fetch first row (after fetch, the default is set to SESAM_SEEK_NEXT)
3	SESAM_SEEK_LAST	fetch last row (after fetch, the default is set to SESAM_SEEK_PRIOR)
4	SESAM_SEEK_ABSOLUTE	fetch absolute row number given as <i>offset</i> (<i>Zero-based. After fetch, the default is set to SESAM_SEEK_ABSOLUTE, and the offset value is auto-incremented</i>)
5	SESAM_SEEK_RELATIVE	fetch relative to current scroll position, where <i>offset</i> can be a positive or negative offset value (<i>this also sets the default "offset" value for subsequent fetches</i>).

offset is an optional parameter which is only evaluated (and required) if *whence* is either SESAM_SEEK_RELATIVE or SESAM_SEEK_ABSOLUTE.

sesam_free_result (PHP 3 CVS only)

Releases resources for the query

```
int sesam_free_result (string result_id)
```

Releases resources for the query associated with *result_id*. Returns FALSE on error.

LXVIII. Funciones para la Gestión de Sesiones

El soporte de sesiones en PHP es básicamente un sistema que preserva ciertos datos en una serie de accesos. Esto permite construir aplicaciones más personalizadas e incrementar el atractivo de un website.

Si ya está familiarizado con la gestión de sesión de PHPLIB, apreciará que algunos conceptos son similares al soporte de sesiones de PHP.

A cada visitante que acceda a su web se le asignan un único id, el conocido id de sesión. Éste se almacena en una cookie del usuario o bien se propaga con la URL.

El soporte de sesión le permite transportar tantas variables como desee a través de las solicitudes del cliente. Cuando un visitante accede a su web, PHP chequea automáticamente (si session.auto_start está puesta a 1), manualmente (si añade el comando `session_start()`) o implicitamente (al añadir `session_register()`) si se ha establecido una sesión concreta con la llamada. Si es así, el entorno grabado es reproducido.

Todas las variables registradas son serializadas tras finalizar la solicitud. Las variables registradas que no estén definidas se marcan como no definidas. En los accesos posteriores, estas variables no las define el módulo de sesión, a no ser que las defina el usuario después.

Los parámetros de configuración `track_vars` y `gpc_globals` influyen en como se recuperan las variables de sesión. Si `track_vars` está activado, las variables recuperadas estarán guardadas en la matriz asociativa `$HTTP_STATE_VARS`. Si `gpc_globals` está activado, las variables de sesión se recuperarán como variables globales. Si ambos parámetros están activados, las variables globales y los registros de `$HTTP_STATE_VARS` valdrán lo mismo.

Hay dos formas de propagar un id de sesión:

- Cookies
- parámetros URL

El módulo de sesión soporta ambos métodos. Las Cookies son ideales, pero como su permanencia no la controla el servidor, sino el cliente no nos podemos fiar. El segundo método integra el id de sesión en las URLs.

PHP es capaz de hacer esto de forma transparente cuando se ha compilado con `-enable-trans-sid`. Si activa esta opción las URIs relativas se cambiarán automáticamente para contener el id de sesión. Otra posibilidad es usar la constante `SID` siempre definida, si el cliente no manda la cookie apropiada. `SID` es de la forma `session_name=session_id` o bien es una cadena vacía.

El siguiente ejemplo muestra como registrar una variable y como enlazarla correctamente a otra página usando SID.

Ejemplo 1. Contando el número de hits en un usuario sencillo

```
<?php  
  
session_register("count");  
  
$count++;  
  
?>  
  
Hola, visitante, has visto esta página <? echo $count; ?> veces.<p>  
  
<?  
# el <?=SID?> es necesario para preservar el id de la sesión  
# para el caso en que el usuario haya desactivado las Cookies  
?>  
  
Para continuar, <A HREF="nextpage.php?<?=SID?>">pulse aquí</A>
```

Para implementar el almacenamiento en base de datos necesita código PHP y la función a nivel de usuario **session_set_save_handler()**. Tendría que extender las siguientes funciones para soportar MySQL y otras bases de datos.

Ejemplo 2. Manejo de session_set_save_handler()

```
<?php

function open ($save_path, $session_name) {
    echo "open ($save_path, $session_name)\n";
    return true;
}

function close () {
    echo "close\n";
    return true;
}

function read ($key) {
    echo "write ($key, $val)\n";
    return "foo|i:1;";
}

function write ($key, $val) {
    echo "write ($key, $val)\n";
    return true;
}

function destroy ($key)
    return true;
}

function gc ($maxlifetime) {
    return true;
}

session_set_save_handler ("open", "close", "read", "write", "destroy", "gc");
session_start ();
$foo++;

?>
```

Producirá los siguientes resultados:

```
$ ./php save_handler.php
Content-Type: text/html
Set-cookie: PHPSESSID=f08b925af0ecb52bdd2de97d95cdbe6b

open (/tmp, PHPSESSID)
read (f08b925af0ecb52bdd2de97d95cdbe6b)
write (f08b925af0ecb52bdd2de97d95cdbe6b, foo|i:2; )
close
```

El <?=SID?> no es necesario, si -enable-trans-sid se usó para compilar PHP.

El sistema gestor de sesiones soporta una serie de opciones de configuración que puede poner en su fichero php.ini. Las vemos a continuación.

- `session.save_handler` define el nombre del handler que se usa para almacenar y recuperar los datos asociados con la sesión. Por defecto es `files`.
- `session.save_path` define el argumento que se pasa al handler de grabación. Si escoge el handler por defecto, éste es el path donde los ficheros son creados. Por defecto es `/tmp`.
- `session.name` especifica el nombre de la sesión el cual se usa como nombre de la cookie. Debe contener sólo caracteres alfanuméricos. Por defecto es `PHPSESSID`.
- `session.auto_start` especifica si el módulo de sesión inicia automáticamente una sesión en la solicitud inicial. Por defecto es 0 (disabled).
- `session.lifetime` especifica el tiempo de vida de una cookie en segundos que se le envía al navegador. El valor 0 significa "hasta que se cierre el navegador". Por defecto es 0.
- `session.serialize_handler` define el nombre del handler que se usa para serializar/deserializar datos. Actualmente, están soportados un formato interno de PHP (con nombre `php`) y WDDX (con nombre `wddx`). WDDX sólo está disponible si se compila PHP con [WDDX support](#). Por defecto es `php`.
- `session.gc_probability` especifica la probabilidad en tanto por cien de que la rutina de gc (garbage collection) se ejecute en cada solicitud. Por defecto es 1.
- `session.gc_maxlifetime` especifica el número de segundos después de los cuales los datos son considerados como 'basura' y por tanto limpiados mediante gc.
- `session.referer_check` determina si los ids de sesión referidos a sitios externos son eliminados. Si los ids de sesión se propagan usando el método de URL los usuarios que no conocen sus implicaciones podrían exponer sin querer los ids de sesión. Esto podría generar problemas de seguridad, que este chequeo trata de eliminar. Por defecto es 0.
- `session.entropy_file` da una ruta a un recurso externo (un fichero) que se usará como fuente adicional de entropía en la creación del id de sesión. Un ejemplo sería `/dev/random` o bien `/dev/urandom` que están disponibles en muchos sistemas Unix.
- `session.entropy_length` especifica el número de bytes que se leerán del fichero arriba indicado. Por defecto es 0 (desactivado).
- `session.use_cookies` especifica si el módulo usará cookies para almacenar el id de sesión en el lado del cliente. Por defecto es 1 (activado).

Nota: La gestión de sesiones se añadió en PHP 4.0.

session_start (PHP 4)

Inicializa las variables de sesión

```
bool session_start(void);
```

session_start() crea una sesión (o continua con la actual basada en el id de sesión pasado mediante una variable GET (o una cookie).

Esta función siempre devuelve true.

Nota: Esta función se añadió en PHP 4.0.

session_destroy (PHP 4)

Elimina toda la información registrada en una sesión

```
bool session_destroy(void);
```

session_destroy() elimina todos los datos asociados a la sesión en curso.

Esta función siempre devuelve true.

Nota: Esta función se añadió en PHP 4.0.

session_name (PHP 4)

Obtiene y/o establece el nombre de la sesión en curso

```
string session_name ([string name])
```

session_name() devuelve el nombre de la sesión en curso. Si *name* está especificado, el nombre de la sesión en curso es cambiado por éste.

El nombre de la sesión referencia el id de sesión en las cookies y en las URLs. Debería contener sólo caracteres alfanuméricos; debe ser corto y descriptivo (para que lo vean los usuarios que tengan activados los warnings de cookies en su navegador). El nombre de la sesión es inicializado a su valor por defecto guardado en `session.name` en el momento de inicio de la sesión. Así, se hace necesario llamar a **session_name()** en cada solicitud (y antes de que **session_start()** o **session_register()** sean llamados).

Ejemplo 1. Ejemplos de session_name()

```
<?php
# Define el nombre de sesión como WebsiteID
$previous_name = session_name("WebsiteID");
echo "El nombre anterior era $previous_name<p>";
```

Nota: Esta función se añadió en PHP 4.0.

session_module_name (PHP 4)

Obtiene y/o establece el módulo de sesión en curso

```
string session_module_name ([string module])
```

session_module_name() devuelve el nombre del módulo de sesión en curso. Si *module* está especificado, dicho módulo será usado en su lugar.

Nota: Esta función se añadió en PHP 4.0.

session_save_path (PHP 4)

Obtiene o establece el path de grabación de la sesión en curso

```
string session_save_path ([string path])
```

session_save_path() devuelve la ruta del directorio que se está usando para grabar la sesión en curso. Si el parámetro *path* se indica, el path donde se graben los datos cambiará.

Nota: En algunos sistemas operativos, puede que le interese especificar un path en el sistema de ficheros que maneje eficazmente grandes cantidades de pequeños ficheros. Por ejemplo, bajo Linux, reiserfs puede dar mejor rendimiento que ext2fs.

Nota: Esta función se añadió en PHP 4.0.

session_id (PHP 4)

Obtiene o establece el id de sesión en curso

```
string session_id ([string id])
```

session_id() devuelve el id de la sesión en curso. Si se especifica *id*, se reemplazará el id de la actual sesión.

directorio usado para grabar los datos de sesión. Si *path* se especifica, el path donde se graben los datos cambiará.

La constante SID se puede usar también para obtener el nombre e id de la sesión en curso como una cadena preparada para ser añadida a las URLs.

Nota: Esta función se añadió en PHP 4.0.

session_register (PHP 4)

Registra una o más variables en la sesión en curso

```
bool session_register (mixed name [, mixed ...])
```

session_register() tiene un número variable de argumentos, cada uno de los cuales puede ser una cadena que contiene el nombre de la variable, o un vector que contenga esos nombres de variables u otros vectores. Por cada variable que se encuentre, **session_register()** registra el nombre de la variable como variable global en la sesión en curso.

Esta función devuelve true cuando la variable se registra con éxito en la sesión.

Nota: Esta función se añadió en PHP 4.0.

session_unregister (PHP 4)

Desliga una variable de la sesión en curso

```
bool session_unregister (string name)
```

session_unregister() desliga (olvida) la variable global llamada *name* de la sesión en curso.

Esta función devuelve true cuando la variable se ha desligado de la sesión.

Nota: Esta función se añadió en PHP 4.0.

session_is_registered (PHP 4)

Aveigua si una variable está registrada en una sesión

```
bool session_is_registered (string name)
```

session_is_registered() devuelve true si hay una variable que se llame *name* registrada en la sesión en curso.

Nota: Esta función se añadió en PHP 4.0.

session_decode (PHP 4)

Extrae los datos de sesión a partir de una cadena

```
bool session_decode (string data)
```

session_decode() extrae los datos de sesión en *data*, dando valores a las variables almacenadas en la sesión.

Nota: Esta función se añadió en PHP 4.0.

session_encode (PHP 4)

Codifica los datos de la sesión en curso en una cadena

```
bool session_encode(void);
```

session_encode() devuelve una cadena con los contenidos de la sesión en curso dentro.

Nota: Esta función se añadió en PHP 4.0.

LXIX. Shared Memory Functions

Shmop is an easy to use set of functions that allows php to read, write, create and delete UNIX shared memory segments. The functions will not work on windows, as it does not support shared memory. To use shmop you will need to compile php with the `--enable-shmop` parameter in your configure line.

Nota: The functions explained in the chapter begin all with `shm_()` in PHP 4.0.3, but in PHP 4.0.4 and later versions these names are changed to begin with `shmop_()`.

Ejemplo 1. Shared Memory Operations Overview

```
<?php

// Create 100 byte shared memory block with system id if 0xff3
$shm_id = shmop_open(0xff3, "c", 0644, 100);
if(!$shm_id) {
echo "Couldn't create shared memory segment\n";
}

// Get shared memory block's size
$shm_size = shmop_size($shm_id);
echo "SHM Block Size: ".$shm_size. " has been created.\n";

// Lets write a test string into shared memory
$shm_bytes_written = shmop_write($shm_id, "my shared memory block", 0);
if($shm_bytes_written != strlen("my shared memory block")) {
echo "Couldn't write the entire length of data\n";
}

// Now lets read the string back
$my_string = shmop_read($shm_id, 0, $shm_size);
if(!$my_string) {
echo "Couldn't read from shared memory block\n";
}
echo "The data inside shared memory was: ".$my_string."\n";

//Now lets delete the block and close the shared memory segment
if(!shmop_delete($shm_id)) {
echo "Couldn't mark shared memory block for deletion.
}
shmop_close($shm_id);

?>
```


shmop_open (PHP 4 >= 4.0.4)

Create or open shared memory block

```
int shmop_open (int key, string flags, int mode, int size)
```

shmop_open() can create or open a shared memory block.

shmop_open() takes 4 parameters: key, which is the system's id for the shared memory block, this parameter can be passed as a decimal or hex. The second parameter are the flags that you can use:

- "a"for access (sets IPC_EXCL) use this flag when you need to open an existing shared memory segment
- "c"for create (sets IPC_CREATE) use this flag when you need to create a new shared memory segment.

The third parameter is the mode, which are the permissions that you wish to assign to your memory segment, those are the same as permission for a file. Permissions need to be passed in octal form ex. 0644. The last parameter is size of the shared memory block you wish to create in bytes.

Nota: Note: the 3rd and 4th should be entered as 0 if you are opening an existing memory segment. On success **shmop_open()** will return an id that you can use to access the shared memory segment you've created.

Ejemplo 1. Create a new shared memory block

```
<?php
$shm_id = shmop_open(0x0fff, "c", 0644, 100);
?>
```

This example opened a shared memory block with a system id of 0x0fff.

shmop_read (PHP 4 >= 4.0.4)

Read data from shared memory block

```
string shmop_read (int shmid, int start, int count)
```

shmop_read() will read a string from shared memory block.

shmop_read() takes 3 parameters: shmid, which is the shared memory block identifier created by **shmop_open()**, offset from which to start reading and count on the number of bytes to read.

Ejemplo 1. Reading shared memory block

```
<?php
$shm_data = shmop_read($shm_id, 0, 50);
?>
```

This example will read 50 bytes from shared memory block and place the data inside \$shm_data.

shmop_write (PHP 4 >= 4.0.4)

Write data into shared memory block

```
int shmop_write (int shmid, string data, int offset)
```

shmop_write() will write a string into shared memory block.

shmop_write() takes 3 parameters: shmid, which is the shared memory block identifier created by **shmop_open()**, data, a string that you want to write into shared memory block and offset, which specifies where to start writing data inside the shared memory segment.

Ejemplo 1. Writing to shared memory block

```
<?php
$shm_bytes_written = shmop_write($shm_id, $my_string, 0);
?>
```

This example will write data inside \$my_string into shared memory block, \$shm_bytes_written will contain the number of bytes written.

shmop_size (PHP 4 >= 4.0.4)

Get size of shared memory block

```
int shmop_size (int shmid)
```

shmop_size() is used to get the size, in bytes of the shared memory block.

shmop_size() takes the shmid, which is the shared memory block identifier created by **shmop_open()**, the function will return and int, which represents the number of bytes the shared memory block occupies.

Ejemplo 1. Getting the size of the shared memory block

```
<?php
$shm_size = shmop_size($shm_id);
?>
```

This example will put the size of shared memory block identified by \$shm_id into \$shm_size.

shmop_delete (PHP 4 >= 4.0.4)

Delete shared memory block

```
int shmop_delete (int shmid)
```

shmop_delete() is used to delete a shared memory block.

shmop_delete() takes the shmid, which is the shared memory block identifier created by **shmop_open()**. On success 1 is returned, on failure 0 is returned.

Ejemplo 1. Deleting shared memory block

```
<?php  
shmop_delete($shm_id);  
?>
```

This example will delete shared memory block identified by \$shm_id.

shmop_close (PHP 4 >= 4.0.4)

Close shared memory block

```
int shmop_close (int shmid)
```

shmop_close() is used to close a shared memory block.

shmop_close() takes the shmid, which is the shared memory block identifier created by **shmop_open()**.

Ejemplo 1. Closing shared memory block

```
<?php  
shmop_close($shm_id);  
?>
```

This example will close shared memory block identified by \$shm_id.

LXX. Shockwave Flash functions

PHP offers the ability to create Shockwave Flash files via Paul Haeberli's libswf module. You can download libswf at <http://reality.sgi.com/grafica/flash/>. Once you have libswf all you need to do is to configure `-with-swf[=DIR]` where DIR is a location containing the directories include and lib. The include directory has to contain the swf.h file and the lib directory has to contain the libswf.a file. If you unpack the libswf distribution the two files will be in one directory. Consequently you will have to copy the files to the proper location manually.

Once you've successfully installed PHP with Shockwave Flash support you can then go about creating Shockwave files from PHP. You would be surprised at what you can do, take the following code:

Ejemplo 1. SWF example

```
<?php
swf_openfile ("test.swf", 256, 256, 30, 1, 1, 1);
swf_ortho2 (-100, 100, -100, 100);
swf_defineline (1, -70, 0, 70, 0, .2);
swf_definerect (4, 60, -10, 70, 0, 0);
swf_definerect (5, -60, 0, -70, 10, 0);
swf_addcolor (0, 0, 0, 0);

swf_definefont (10, "Mod");
swf_fontsize (5);
swf_fontslant (10);
swf_definetext (11, "This be Flash wit PHP!", 1);

swf_pushmatrix ();
swf_translate (-50, 80, 0);
swf_placeobject (11, 60);
swf_popmatrix ();

for ($i = 0; $i < 30; $i++) {
    $p = $i/(30-1);
    swf_pushmatrix ();
    swf_scale (1-($p*.9), 1, 1);
    swf_rotate (60*$p, 'z');
    swf_translate (20+20*$p, $p/1.5, 0);
    swf_rotate (270*$p, 'z');
    swf_addcolor ($p, 0, $p/1.2, -$p);
    swf_placeobject (1, 50);
    swf_placeobject (4, 50);
    swf_placeobject (5, 50);
    swf_popmatrix ();
    swf_showframe ();
}

for ($i = 0; $i < 30; $i++) {
    swf_removeobject (50);
    if (($i%4) == 0) {
        swf_showframe ();
    }
}

swf_startdoaction ();
swf_actionstop ();
swf_enddoaction ();

swf_closefile ();
?>
```

It will produce the animation found at the following url (<http://www.designmultimedia.com/swfphp/test.swf>).

Nota: SWF support was added in PHP4 RC2.

swf_openfile (PHP 4 >= 4.0RC2)

Open a new Shockwave Flash file

```
void swf_openfile (string filename, float width, float height, float framerate, float r,  
float g, float b)
```

The **swf_openfile()** function opens a new file named *filename* with a width of *width* and a height of *height* a frame rate of *framerate* and background with a red color of *r* a green color of *g* and a blue color of *b*.

The **swf_openfile()** must be the first function you call, otherwise your script will cause a segfault. If you want to send your output to the screen make the filename: "php://stdout"(support for this is in 4.0.1 and up).

swf_closefile (PHP 4 >= 4.0RC2)

Close the current Shockwave Flash file

```
void swf_closefile (void);
```

Close a file that was opened by the **swf_openfile()** function.

swf_labelframe (PHP 4 >= 4.0RC2)

Label the current frame

```
void swf_labelframe (string name)
```

Label the current frame with the name given by the *name* parameter.

swf_showframe (PHP 4 >= 4.0RC2)

Display the current frame

```
void swf_showframe (void);
```

The **swf_showframe** function will output the current frame.

swf_setframe (PHP 4 >= 4.0RC2)

Switch to a specified frame

```
void swf_setframe (int framenumber)
```

The **swf_setframe()** changes the active frame to the frame specified by *framenumber*.

swf_getframe (PHP 4 >= 4.0RC2)

Get the frame number of the current frame

```
int swf_getframe (void);
```

The **swf_getframe()** function gets the number of the current frame.

swf_mulcolor (PHP 4 >= 4.0RC2)

Sets the global multiply color to the *rgba* value specified

```
void swf_mulcolor (float r, float g, float b, float a)
```

The **swf_mulcolor()** function sets the global multiply color to the *rgba* color specified. This color is then used (implicitly) by the **swf_placeobject()**, **swf_modifyobject()** and the **swf_addbuttonrecord()** functions. The color of the object will be multiplied by the *rgba* values when the object is written to the screen.

Nota: The *rgba* values can be either positive or negative.

swf_adddcolor (PHP 4 >= 4.0RC2)

Set the global add color to the *rgba* value specified

```
void swf_adddcolor (float r, float g, float b, float a)
```

The **swf_adddcolor()** function sets the global add color to the *rgba* color specified. This color is then used (implicitly) by the **swf_placeobject()**, **swf_modifyobject()** and the **swf_addbuttonrecord()** functions. The color of the object will be add by the *rgba* values when the object is written to the screen.

Nota: The *rgba* values can be either positive or negative.

swf_placeobject (PHP 4 >= 4.0RC2)

Place an object onto the screen

```
void swf_placeobject (int objid, int depth)
```

Places the object specified by *objid* in the current frame at a depth of *depth*. The *objid* parameter and the *depth* must be between 1 and 65535.

This uses the current mulcolor (specified by **swf_mulcolor()**) and the current addcolor (specified by **swf_adddcolor()**) to color the object and it uses the current matrix to position the object.

Nota: Full RGBA colors are supported.

swf_modifyobject (PHP 4 >= 4.0RC2)

Modify an object

```
void swf_modifyobject (int depth, int how)
```

Updates the position and/or color of the object at the specified depth, *depth*. The parameter *how* determines what is updated. *how* can either be the constant MOD_MATRIX or MOD_COLOR or it can be a combination of both (MOD_MATRIX|MOD_COLOR).

MOD_COLOR uses the current mulcolor (specified by the function **swf_mulcolor()**) and addcolor (specified by the function **swf_addcolor()**) to color the object. MOD_MATRIX uses the current matrix to position the object.

swf_removeobject (PHP 4 >= 4.0RC2)

Remove an object

```
void swf_removeobject (int depth)
```

Removes the object at the depth specified by *depth*.

swf_nextid (PHP 4 >= 4.0RC2)

Returns the next free object id

```
int swf_nextid (void);
```

The **swf_nextid()** function returns the next available object id.

swf_startdoaction (PHP 4 >= 4.0RC2)

Start a description of an action list for the current frame

```
void swf_startdoaction (void);
```

The **swf_startdoaction()** function starts the description of an action list for the current frame. This must be called before actions are defined for the current frame.

swf_actiongotoframe (PHP 4 >= 4.0RC2)

Play a frame and then stop

```
void swf_actiongotoframe (int framenumber)
```

The **swf_actionGotoFrame()** function will go to the frame specified by *framenumber*, play it, and then stop.

swf_actiongetUrl (PHP 4 >= 4.0RC2)

Get a URL from a Shockwave Flash movie

```
void swf_actiongetUrl (string url, string target)
```

The **swf_actionGetUrl()** function gets the URL specified by the parameter *url* with the target *target*.

swf_actionnextframe (PHP 4 >= 4.0RC2)

Go foward one frame

```
void swf_actionnextframe (void);
```

Go foward one frame.

swf_actionprevframe (PHP 4 >= 4.0RC2)

Go backwards one frame

```
void swf_actionprevframe (void);
```

swf_actionplay (PHP 4 >= 4.0RC2)

Start playing the flash movie from the current frame

```
void swf_actionplay (void);
```

Start playing the flash movie from the current frame.

swf_actionstop (PHP 4 >= 4.0RC2)

Stop playing the flash movie at the current frame

```
void swf_actionstop (void);
```

Stop playing the flash movie at the current frame.

swf_actiontogglequality (PHP 4 >= 4.0RC2)

Toggle between low and high quality

```
void swf_actiontogglequality (void);
```

Toggle the flash movie between high and low quality.

swf_actionwaitForframe (PHP 4 >= 4.0RC2)

Skip actions if a frame has not been loaded

```
void swf_actionwaitForframe (int framenumber, int skipcount)
```

The **swf_actionWaitForFrame()** function will check to see if the frame, specified by the *framenumber* parameter has been loaded, if not it will skip the number of actions specified by the *skipcount* parameter. This can be useful for "Loading..." type animations.

swf_actionsettarget (PHP 4 >= 4.0RC2)

Set the context for actions

```
void swf_actionsettarget (string target)
```

The **swf_actionSetTarget()** function sets the context for all actions. You can use this to control other flash movies that are currently playing.

swf_actiongotoLabel (PHP 4 >= 4.0RC2)

Display a frame with the specified label

```
void swf_actiongotoLabel (string label)
```

The **swf_actionGotoLabel()** function displays the frame with the label given by the *label* parameter and then stops.

swf_enddoaction (PHP 4 >= 4.0RC2)

End the current action

```
void swf_enddoaction (void);
```

Ends the current action started by the **swf_startdoaction()** function.

swf_defineline (PHP 4 >= 4.0RC2)

Define a line

```
void swf_defineline (int objid, float x1, float y1, float x2, float y2, float width)
```

The **swf_defineline()** defines a line starting from the x coordinate given by *x1* and the y coordinate given by *y1* parameter. Up to the x coordinate given by the *x2* parameter and the y coordinate given by the *y2* parameter. It will have a width defined by the *width* parameter.

swf_definerect (PHP 4 >= 4.0RC2)

Define a rectangle

```
void swf_definerect (int objid, float x1, float y1, float x2, float y2, float width)
```

The **swf_definerect()** defines a rectangle with an upper left hand coordinate given by the x, *x1*, and the y, *y1*. And a lower right hand coordinate given by the x coordinate, *x2*, and the y coordinate, *y2*. Width of the rectangles border is given by the *width* parameter, if the width is 0.0 then the rectangle is filled.

swf_definepoly (PHP 4 >= 4.0.0)

Define a polygon

```
void swf_definepoly (int objid, array coords, int npoints, float width)
```

The **swf_definepoly()** function defines a polygon given an array of x, y coordinates (the coordinates are defined in the parameter *coords*). The parameter *npoints* is the number of overall points that are contained in the array given by *coords*. The *width* is the width of the polygon's border, if set to 0.0 the polygon is filled.

swf_startshape (PHP 4 >= 4.0RC2)

Start a complex shape

```
void swf_startshape (int objid)
```

The **swf_startshape()** function starts a complex shape, with an object id given by the *objid* parameter.

swf_shapelinesolid (PHP 4 >= 4.0RC2)

Set the current line style

```
void swf_shapelinesolid (float r, float g, float b, float a, float width)
```

The **swf_shapeLineSolid()** function sets the current line style to the color of the *rgba* parameters and width to the *width* parameter. If 0.0 is given as a width then no lines are drawn.

swf_shapefilloff (PHP 4 >= 4.0RC2)

Turns off filling

```
void swf_shapefilloff (void);
```

The **swf_shapeFillOff()** function turns off filling for the current shape.

swf_shapefillsolid (PHP 4 >= 4.0RC2)

Set the current fill style to the specified color

```
void swf_shapefillsolid (float r, float g, float b, float a)
```

The **swf_shapeFillSolid()** function sets the current fill style to solid, and then sets the fill color to the values of the *rgba* parameters.

swf_shapefillbitmapclip (PHP 4 >= 4.0RC2)

Set current fill mode to clipped bitmap

```
void swf_shapefillbitmapclip (int bitmapid)
```

Sets the fill to bitmap clipped, empty spaces will be filled by the bitmap given by the *bitmapid* parameter.

swf_shapefillbitmaptile (PHP 4 >= 4.0RC2)

Set current fill mode to tiled bitmap

```
void swf_shapefillbitmaptile (int bitmapid)
```

Sets the fill to bitmap tile, empty spaces will be filled by the bitmap given by the *bitmapid* parameter (tiled).

swf_shapemoveto (PHP 4 >= 4.0RC2)

Move the current position

```
void swf_shapemoveto (float x, float y)
```

The **swf_shapeMoveTo()** function moves the current position to the x coordinate given by the *x* parameter and the y position given by the *y* parameter.

swf_shapelineto (PHP 4 >= 4.0RC2)

Draw a line

```
void swf_shapelineto (float x, float y)
```

The **swf_shapeLineTo()** draws a line to the x,y coordinates given by the *x* parameter & the *y* parameter. The current position is then set to the x,y parameters.

swf_shapeCurveTo (PHP 4 >= 4.0RC2)

Draw a quadratic bezier curve between two points

```
void swf_shapeCurveTo (float x1, float y1, float x2, float y2)
```

The **swf_shapeCurveTo()** function draws a quadratic bezier curve from the x coordinate given by *x1* and the y coordinate given by *y1* to the x coordinate given by *x2* and the y coordinate given by *y2*. The current position is then set to the x,y coordinates given by the *x2* and *y2* parameters

swf_shapeCurveTo3 (PHP 4 >= 4.0RC2)

Draw a cubic bezier curve

```
void swf_shapeCurveTo3 (float x1, float y1, float x2, float y2, float x3, float y3)
```

Draw a cubic bezier curve using the x,y coordinate pairs *x1*,*y1* and *x2*,*y2* as off curve control points and the x,y coordinate *x3*, *y3* as an endpoint. The current position is then set to the x,y coordinate pair given by *x3*,*y3*.

swf_shapeArc (PHP 4 >= 4.0RC2)

Draw a circular arc

```
void swf_shapeArc (float x, float y, float r, float ang1, float ang2)
```

The **swf_shapeArc()** function draws a circular arc from angle A given by the *ang1* parameter to angle B given by the *ang2* parameter. The center of the circle has an x coordinate given by the *x* parameter and a y coordinate given by the *y*, the radius of the circle is given by the *r* parameter.

swf_endshape (PHP 4 >= 4.0RC2)

Completes the definition of the current shape

```
void swf_endshape (void);
```

The **swf_endshape()** completes the definition of the current shape.

swf_definefont (PHP 4 >= 4.0RC2)

Defines a font

```
void swf_definefont (int fontid, string fontname)
```

The **swf_definefont()** function defines a font given by the *fontname* parameter and gives it the id specified by the *fontid* parameter. It then sets the font given by *fontname* to the current font.

swf_setfont (PHP 4 >= 4.0RC2)

Change the current font

```
void swf_setfont (int fontid)
```

The **swf_setfont()** sets the current font to the value given by the *fontid* parameter.

swf_fontsize (PHP 4 >= 4.0RC2)

Change the font size

```
void swf_fontsize (float size)
```

The **swf_fontsize()** function changes the font size to the value given by the *size* parameter.

swf_fontslant (PHP 4 >= 4.0RC2)

Set the font slant

```
void swf_fontslant (float slant)
```

Set the current font slant to the angle indicated by the *slant* parameter. Positive values create a foward slant, negative values create a negative slant.

swf_fonttracking (PHP 4 >= 4.0RC2)

Set the current font tracking

```
void swf_fonttracking (float tracking)
```

Set the font tracking to the value specified by the *tracking* parameter. This function is used to increase the spacing between letters and text, positive values increase the space and negative values decrease the space between letters.

swf_getfontinfo (PHP 4 >= 4.0RC2)

The height in pixels of a capital A and a lowercase x

```
array swf_getfontinfo (void);
```

The **swf_getfontinfo()** function returns an associative array with the following parameters:

- *Aheight* - The height in pixels of a capital A.
- *xheight* - The height in pixels of a lowercase x.

swf_definetext (PHP 4 >= 4.0RC2)

Define a text string

```
void swf_definetext (int objid, string str, int docenter)
```

Define a text string (the *str* parameter) using the current font and font size. The *docenter* is where the word is centered, if *docenter* is 1, then the word is centered in x.

swf_textwidth (PHP 4 >= 4.0RC2)

Get the width of a string

```
float swf_textwidth (string str)
```

The **swf_textwidth()** function gives the width of the string, *str*, in pixels, using the current font and font size.

swf_definebitmap (PHP 4 >= 4.0RC2)

Define a bitmap

```
void swf_definebitmap (int objid, string image_name)
```

The **swf_definebitmap()** function defines a bitmap given a GIF, JPEG, RGB or FI image. The image will be converted into a Flash JPEG or Flash color map format.

swf_getbitmapinfo (PHP 4 >= 4.0RC2)

Get information about a bitmap

```
array swf_getbitmapinfo (int bitmapid)
```

The **swf_getbitmapinfo()** function returns an array of information about a bitmap given by the *bitmapid* parameter. The returned array has the following elements:

- "size"- The size in bytes of the bitmap.
- "width"- The width in pixels of the bitmap.
- "height"- The height in pixels of the bitmap.

swf_startsymbol (PHP 4 >= 4.0RC2)

Define a symbol

```
void swf_startsymbol (int objid)
```

Define an object id as a symbol. Symbols are tiny flash movies that can be played simultaneously. The *objid* parameter is the object id you want to define as a symbol.

swf_endsymbol (PHP 4 >= 4.0RC2)

End the definition of a symbol

```
void swf_endsymbol (void);
```

The **swf_endsymbol()** function ends the definition of a symbol that was started by the **swf_startsymbol()** function.

swf_startbutton (PHP 4 >= 4.0RC2)

Start the definition of a button

```
void swf_startbutton (int objid, int type)
```

The **swf_startbutton()** function starts off the definition of a button. The *type* parameter can either be TYPE_MENUBUTTON or TYPE_PUSHBUTTON. The TYPE_MENUBUTTON constant allows the focus to travel from the button when the mouse is down, TYPE_PUSHBUTTON does not allow the focus to travel when the mouse is down.

swf_addbuttonrecord (PHP 4 >= 4.0RC2)

Controls location, appearance and active area of the current button

```
void swf_addbuttonrecord (int states, int shapeid, int depth)
```

The **swf_addbuttonrecord()** function allows you to define the specifics of using a button. The first parameter, *states*, defines what states the button can have, these can be any or all of the following constants: BSHitTest, BSDown, BSOver or BSUp. The second parameter, the *shapeid* is the look of the button, this is usually the object id of the shape of the button. The *depth* parameter is the placement of the button in the current frame.

Ejemplo 1. Swf_addbuttonrecord() function example

```
swf_startButton ($objid, TYPE_MENUBUTTON);
    swf_addButtonRecord (BSDown|BSOver, $buttonImageId, 340);
    swf_onCondition (MenuEnter);
        swf_actionGetUrl ("http://www.designmultimedia.com", "_level1");
    swf_onCondition (MenuExit);
        swf_actionGetUrl ("", "_level1");
swf_endButton ();
```

swf_oncondition (PHP 4 >= 4.0RC2)

Describe a transition used to trigger an action list

```
void swf_oncondition (int transition)
```

The **swf_onCondition()** function describes a transition that will trigger an action list. There are several types of possible transitions, the following are for buttons defined as TYPE_MENUBUTTON:

- IdletoOverUp
- OverUptoIdle
- OverUptoOverDown
- OverDowntoOverUp
- IdletoOverDown
- OutDowntoIdle
- MenuEnter (IdletoOverUp|IdletoOverDown)
- MenuExit (OverUptoIdle|OverDowntoIdle)

For TYPE_PUSHBUTTON there are the following options:

- IdletoOverUp
- OverUptoIdle
- OverUptoOverDown
- OverDowntoOverUp
- OverDowntoOutDown
- OutDowntoOverDown
- OutDowntoIdle
- ButtonEnter (IdletoOverUp|OutDowntoOverDown)
- ButtonExit (OverUptoIdle|OverDowntoOutDown)

swf_endbutton (PHP 4 >= 4.0RC2)

End the definition of the current button

```
void swf_endbutton (void);
```

The **swf_endButton()** function ends the definition of the current button.

swf_viewport (PHP 4 >= 4.0RC2)

Select an area for future drawing

```
void swf_viewport (double xmin, double xmax, double ymin, double ymax)
```

The **swf_viewport()** function selects an area for future drawing for *xmin* to *xmax* and *ymin* to *ymax*, if this function is not called the area defaults to the size of the screen.

swf_ortho (PHP 4 >= 4.0.1)

Defines an orthographic mapping of user coordinates onto the current viewport

```
void swf_ortho (double xmin, double xmax, double ymin, double ymax, double zmin, double zmax)
```

The **swf_ortho()** function defines a orthographic mapping of user coordinates onto the current viewport.

swf_ortho2 (PHP 4 >= 4.0RC2)

Defines 2D orthographic mapping of user coordinates onto the current viewport

```
void swf_ortho2 (double xmin, double xmax, double ymin, double ymax)
```

The **swf_ortho2()** function defines a two dimensional orthographic mapping of user coordinates onto the current viewport, this defaults to one to one mapping of the area of the Flash movie. If a perspective transformation is desired, the **swf_perspective()** function can be used.

swf_perspective (PHP 4 >= 4.0RC2)

Define a perspective projection transformation

```
void swf_perspective (double fovy, double aspect, double near, double far)
```

The **swf_perspective()** function defines a perspective projection transformation. The *fovy* parameter is field-of-view angle in the y direction. The *aspect* parameter should be set to the aspect ratio of the viewport that is being drawn onto. The *near* parameter is the near clipping plane and the *far* parameter is the far clipping plane.

Nota: Various distortion artifacts may appear when performing a perspective projection, this is because Flash players only have a two dimensional matrix. Some are not to pretty.

swf_polarview (PHP 4 >= 4.0RC2)

Define the viewer's position with polar coordinates

```
void swf_polarview (double dist, double azimuth, double incidence, double twist)
```

The **swf_polarview()** function defines the viewer's position in polar coordinates. The *dist* parameter gives the distance between the viewpoint to the world space origin. The *azimuth* parameter defines the azimuthal angle in the x,y coordinate plane, measured in distance from the y axis. The *incidence* parameter defines the angle of incidence in the y,z plane, measured in distance from the z axis. The incidence angle is defined as the angle of the viewport relative to the z

axis. Finally the *twist* specifies the amount that the viewpoint is to be rotated about the line of sight using the right hand rule.

swf_lookat (PHP 4 >= 4.0RC2)

Define a viewing transformation

```
void swf_lookat (double view_x, double view_y, double view_z, double reference_x, double  
  reference_y, double reference_z, double twist)
```

The **swf_lookat()** function defines a viewing transformation by giving the viewing position (the parameters *view_x*, *view_y*, and *view_z*) and the coordinates of a reference point in the scene, the reference point is defined by the *reference_x*, *reference_y* , and *reference_z* parameters. The *twist* controls the rotation along with viewer's z axis.

swf_pushmatrix (PHP 4 >= 4.0RC2)

Push the current transformation matrix back unto the stack

```
void swf_pushmatrix (void);
```

The **swf_pushmatrix()** function pushes the current transformation matrix back onto the stack.

swf_popmatrix (PHP 4 >= 4.0RC2)

Restore a previous transformation matrix

```
void swf_popmatrix (void);
```

The **swf_popmatrix()** function pushes the current transformation matrix back onto the stack.

swf_scale (PHP 4 >= 4.0RC2)

Scale the current transformation

```
void swf_scale (double x, double y, double z)
```

The **swf_scale()** scales the x coordinate of the curve by the value of the *x* parameter, the y coordinate of the curve by the value of the *y* parameter, and the z coordinate of the curve by the value of the *z* parameter.

swf_translate (PHP 4 >= 4.0RC2)

Translate the current transformations

```
void swf_translate (double x, double y, double z)
```

The **swf_translate()** function translates the current transformation by the *x*, *y*, and *z* values given.

swf_rotate (PHP 4 >= 4.0RC2)

Rotate the current transformation

```
void swf_rotate (double angle, string axis)
```

The **swf_rotate()** rotates the current transformation by the angle given by the *angle* parameter around the axis given by the *axis* parameter. Valid values for the axis are 'x' (the x axis), 'y' (the y axis) or 'z' (the z axis).

swf_posround (PHP 4 >= 4.0RC2)

Enables or Disables the rounding of the translation when objects are placed or moved

```
void swf_posround (int round)
```

The **swf_posround()** function enables or disables the rounding of the translation when objects are placed or moved, there are times when text becomes more readable because rounding has been enabled. The *round* is whether to enable rounding or not, if set to the value of 1, then rounding is enabled, if set to 0 then rounding is disabled.

LXXI. Funciones SNMP

Para usar las funciones SNMP en Unix necesita instalar el paquete UCD SNMP (<http://ucd-snmp.ucdavis.edu/>). En Windows estas funciones están solamente disponibles en NT y no en Win95/98.

Importante: Para usar el paquete UCD SNMP, necesita definir NO_ZEROLENGTH_COMMUNITY a 1 antes de compilarlo. Despues de configurar UCD SNMP, edite config.h y busque NO_ZEROLENGTH_COMMUNITY. Descomente la linea #define. Debería de verse como sigue:

```
#define NO_ZEROLENGTH_COMMUNITY 1
```

Si ve faltas de segmentación desconocidas en combinación con los comandos SNMP, no siga las siguientes instrucciones. Si no desea recompilar UCD SNMP, puede compilar PHP con la opción –enable-ucd-snmp-hack la cual trabajará entorno a las mismas características.

snmpget (PHP 3, PHP 4)

Va a buscar un objeto SNMP

```
string snmpget (string hostname, string community, string object_id [, int timeout [, int retries]])
```

Devuelve el valor de un objeto SNMP en caso de exito y false en caso de error.

La función **snmpget()** es usada para leer el valor de un objeto SNMP especificado por el *object_id*. El agente SNMP es especificado por el *hostname* y la comunidad lectora es especificada por el parametro *community*.

```
$syscontact = snmpget("127.0.0.1", "public", "system.SysContact.0")
```

snmpset (PHP 3>= 3.0.12, PHP 4 >= 4.0b2)

Va a buscar un objeto SNMP

```
string snmpset (string hostname, string community, string object_id, string type, mixed value [, int timeout [, int retries]])
```

Establece el valor especificado para el objeto SNMP, devolviendo true en caso de exito o false en caso de error.

La función **snmpset()** es usada para establecer el valor de un objeto SNMP especificado por el *object_id*. El agente SNMP es especificado por el *hostname* y la comunidad lectora por el parametro *community*.

snmpwalk (PHP 3, PHP 4)

Busqueda por un arbol de informacion acerca de un entidad de red

```
array snmpwalk (string hostname, string community, string object_id [, int timeout [, int retries]])
```

Devuelve una matriz de valores de objetos SMNP comenzando por el **object_id()** como raíz y false en caso de error.

La función **snmpwalk()** es usada para leer todos los valores de un agente SNMP especificado por el *hostname*. *Community* especifica la comunidad lectora para el agente. Un *object_id* nulo se toma como la raíz del arbol de los objetos SNMP y todos los objetos por debajo de ese arbol son devueltos como una matriz. Si *object_id* es especificado, todos los objetos SNMP por debajo de *object_id* son devueltos.

```
$a = snmpwalk("127.0.0.1", "public", "");
```

Encima de una función de llamada podrían devolverse todos los objetos SNMP del agente SNMP en ejecución en el servidor local. Uno puede pasar por todos los valores con un bucle.

```
for ($i=0; $i<count($a); $i++) {  
    echo $a[$i];  
}
```

snmpwalkoid (PHP 3>= 3.0.8, PHP 4 >= 4.0b2)

Busqueda por un arbol de informacion acerca de un entidad de red

```
array snmpwalkoid (string hostname, string community, string object_id [, int timeout [, int retries]])
```

Devuelve una matriz asociativa con los identificadores de los objetos y sus respectivos valores comenzando por el *object_id* como raíz y false en caso de error.

La función **snmpwalkoid()** es usada para leer todos los identificadores de objetos y sus respectivos valores de un agente SNMP especificado por el nombre del servidor. La lectura de *community* especifica la comunidad para el agente. Un *object_id* nulo es tomado como la raíz del arbol de objetos SNMP y todos los objetos por debajo de este arbol son devueltos como una matriz. Si *object_id* es especificado, todos los objetos SNMP inferiores al *object_id* son devueltos.

La existencia de **snmpwalkoid()** y **snmpwalk()** tiene razones historicas. Ambas funciones son proporcionadas para compatibilidad hacia atrás.

```
$a = snmpwalkoid("127.0.0.1", "public", "");
```

La llamada a las funciones superiores devuelve todos los objetos SNMP del agente SNMP en ejecución en el servidor local. Uno puede pasar por todos los valores con un bucle.

```
for (reset($a); $i = key($a); next($a)) {
    echo "$i: $a[$i]<br>\n";
}
```

snmp_get_quick_print (PHP 3>= 3.0.8, PHP 4 >= 4.0b2)

Va a buscar el valor actual de la biblioteca UCD estableciendo quick_print

```
boolean snmp_get_quick_print (void )
```

Delvuele el valor actual almacenado en la biblioteca UCD para quick_print. quick_print está desactivado por defecto.

```
$quickprint = snmp_get_quick_print();
```

La llamada a la función superior podría devolver false si quick_print está activo, y true si quick_print está activo.

snmp_get_quick_print() está solamente disponible cuando estemos usando la biblioteca UCD SNMP. Esta función no está disponible cuando estemos usando la biblioteca Windows SNMP.

Ver: **snmp_get_quick_print()** para una descripción completa de lo que hace quick_print.

snmp_set_quick_print (PHP 3>= 3.0.8, PHP 4 >= 4.0b2)

Establece el valor de quick_print con el de la biblioteca UCD SNMP.

```
void snmp_set_quick_print (boolean quick_print)
```

Establece el valor de quick_print con la biblioteca UCD SNMP. Cuando esto está establecido (1), la biblioteca SNMP devolverá valores 'quick printed'. De esta manera sólo el valor será impreso. Cuando quick_print no está activada (por defecto) la biblioteca UCD SNMP imprime información extra incluyendo el tipo del valor (p. Ej. IPAddress o OID). Adicionalmente, si quick_print no está activado, la biblioteca imprime valores hexadecimales adicionales para todas las cadenas de 3 o menos caracteres.

El ajuste de quick_print es generalmente usado cuando usando la información devuelta con anterioridad se muestra.

```
snmp_set_quick_print(0);
$a = snmpget("127.0.0.1", "public", ".1.3.6.1.2.1.2.2.1.9.1");
echo "$a<BR>\n";
snmp_set_quick_print(1);
$a = snmpget("127.0.0.1", "public", ".1.3.6.1.2.1.2.2.1.9.1");
echo "$a<BR>\n";
```

El primer valor impreso debe de ser: 'Timeticks: (0) 0:00:00.00', donde quick_print se activa, solo se imprimirá '0:00:00.00'.

Por defecto la biblioteca UCD SNMP devuelve valores detallados, quick_print es usado para devolver solamente el valor.

Las cadenas son mantenidas normalmente con comillas extra, esto será corregido en versiones posteriores.

snmp_get_quick_print() está sólo disponible cuando estemos usando la biblioteca UCD SNMP. Esta función no está disponible cuando estemos usando la biblioteca Windows SNMP.

LXXII. Socket functions

The socket extension implements a low-level interface to the socket communication functions, providing the possibility to act as a socket server as well as a client.

For a more generic client-side socket interface, see **fsockopen()** and **pfsockopen()**.

When using the socket functions described here, it is important to remember that while many of them have identical names to their C counterparts, they often have different declarations. Please be sure to read the descriptions to avoid confusion.

That said, those unfamiliar with socket programming can still find a lot of useful material in the appropriate Unix man pages, and there is a great deal of tutorial information on socket programming in C on the web, much of which can be applied, with slight modifications, to socket programming in PHP.

Ejemplo 1. Socket example: Simple TCP/IP server

This example shows a simple talkback server. Change the address and port variables to suit your setup and execute. You may then connect to the server with a command similar to: **telnet 192.168.1.53 10000** (where the address and port match your setup). Anything you type will then be output on the server side, and echoed back to you. To disconnect, enter 'quit'.

```
<?php
error_reporting(E_ALL);

/* Allow the script to hang around waiting for connections. */
set_time_limit(0);

$address = '192.168.1.53';
$port = 10000;

if (($sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
    echo "socket() failed: reason: " . strerror($sock) . "\n";
}

if (($ret = bind($sock, $address, $port)) < 0) {
    echo "bind() failed: reason: " . strerror($ret) . "\n";
}

if (($ret = listen($sock, 5)) < 0) {
    echo "listen() failed: reason: " . strerror($ret) . "\n";
}

do {
    if (($msgsock = accept_connect($sock)) < 0) {
        echo "accept_connect() failed: reason: " . strerror($msgsock) . "\n";
        break;
    }
    do {
        $buf = "";
        $ret = read($msgsock, $buf, 2048);
        if ($ret < 0) {
            echo "read() failed: reason: " . strerror($ret) . "\n";
            break 2;
        }
        if ($ret == 0) {
            break 2;
        }
        $buf = trim($buf);
        if ($buf == 'quit') {
            close($msgsock);
            break 2;
        }
    }
    $talkback = "PHP: You said '$buf'.\n";
}
```

```

        write($msgsock, $talkback, strlen($talkback));
        echo "$buf\n";
    } while (true);
    close($msgsock);
} while (true);

close($sock);
?>

```

Ejemplo 2. Socket example: Simple TCP/IP client

This example shows a simple, one-shot HTTP client. It simply connects to a page, submits a HEAD request, echoes the reply, and exits.

```

<?php
error_reporting(E_ALL);

echo "<h2>TCP/IP Connection</h2>\n";

/* Get the port for the WWW service. */
$service_port = getservbyname('www', 'tcp');

/* Get the IP address for the target host. */
$address = gethostbyname('www.php.net');

/* Create a TCP/IP socket. */
$socket = socket(AF_INET, SOCK_STREAM, 0);
if ($socket < 0) {
    echo "socket() failed: reason: " . strerror($socket) . "\n";
} else {
    "socket() successful: " . strerror($socket) . "\n";
}

echo "Attempting to connect to '$address' on port '$service_port'...";
$result = connect($socket, $address, $service_port);
if ($result < 0) {
    echo "connect() failed.\nReason: ($result) " . strerror($result) . "\n";
} else {
    echo "OK.\n";
}

$in = "HEAD / HTTP/1.0\r\n\r\n";
$out = "";

echo "Sending HTTP HEAD request...";
write($socket, $in, strlen($in));
echo "OK.\n";

echo "Reading response:\n\n";
while (read($socket, $out, 2048)) {
    echo $out;
}

echo "Closing socket...";
close($socket);
echo "OK.\n\n";
?>

```


accept_connect (PHP 4 >= 4.0.2)

Accepts a connection on a socket.

```
int accept_connect (int socket)
```

After the socket *socket* has been created using **socket()**, bound to a name with **bind()**, and told to listen for connections with **listen()**, this function will accept incoming connections on that socket. Once a successful connection is made, a new socket descriptor is returned, which may be used for communication. If there are multiple connections queued on the socket, the first will be used. If there are no pending connections, **accept_connect()** will block until a connection becomes present. If *socket* has been made non-blocking using **socket_set_blocking()** or **set_nonblock()**, an error code will be returned.

The socket descriptor returned by **accept_connect()** may not be used to accept new connections. The original listening socket *socket*, however, remains open and may be reused.

Returns a new socket descriptor on success, or a negative error code on failure. This code may be passed to **strerror()** to get a textual explanation of the error.

See also **bind()**, **connect()**, **listen()**, **socket()**, and **strerror()**.

bind (PHP 4 >= 4.0.2)

Binds a name to a socket.

```
int bind (int socket, string address [, int protocol])
```

bind() binds the name given in *address* to the socket described by *socket*, which must be a valid socket descriptor created with **socket()**.

The *address* parameter is either an IP address in dotted-quad notation (e.g. 127.0.0.1), if the socket is of the AF_INET family; or the pathname of a Unix-domain socket, if the socket family is AF_UNIX.

The *port* parameter is only used when connecting to an AF_INET socket, and designates the port on the remote host to which a connection should be made.

Returns zero on success, or a negative error code on failure. This code may be passed to **strerror()** to get a textual explanation of the error.

See also **accept_connect()**, **connect()**, **listen()**, **socket()**, and **strerror()**.

connect (PHP 4 >= 4.0.2)

Initiates a connection on a socket.

```
int connect (int socket, string address [, int port])
```

Initiates a connection using the socket descriptor *socket*, which must be a valid socket descriptor created with **socket()**.

The *address* parameter is either an IP address in dotted-quad notation (e.g. 127.0.0.1), if the socket is of the AF_INET family; or the pathname of a Unix-domain socket, if the socket family is AF_UNIX.

The *port* parameter is only used when connecting to an AF_INET socket, and designates the port on the remote host to which a connection should be made.

Returns zero on success, or a negative error code on failure. This code may be passed to **strerror()** to get a textual explanation of the error.

See also **bind()**, **listen()**, **socket()**, and **strerror()**.

listen (PHP 4 >= 4.0.2)

Listens for a connection on a socket.

```
int listen (int socket, int backlog)
```

After the socket *socket* has been created using **socket()** and bound to a name with **bind()**, it may be told to listen for incoming connections on *socket*. A maximum of *backlog* incoming connections will be queued for processing.

listen() is applicable only to sockets with type **SOCK_STREAM** or **SOCK_SEQPACKET**.

Returns zero on success, or a negative error code on failure. This code may be passed to **strerror()** to get a textual explanation of the error.

See also **accept_connect()**, **bind()**, **connect()**, **socket()**, and **strerror()**.

socket (PHP 4 >= 4.0.2)

Create a socket (endpoint for communication).

```
int socket (int domain, int type, int protocol)
```

Creates a communication endpoint (a socket), and returns a descriptor to the socket.

The *domain* parameter sets the domain. Currently, **AF_INET** and **AF_UNIX** are understood.

The *type* parameter selects the socket type. This is one of **SOCK_STREAM**, **SOCK_DGRAM**, **SOCK_SEQPACKET**, **SOCK_RAW**, **SOCK_RDM**, or **SOCK_PACKET**.

protocol sets the protocol.

Returns a valid socket descriptor on success, or a negative error code on failure. This code may be passed to **strerror()** to get a textual explanation of the error.

For more information on the usage of **socket()**, as well as on the meanings of the various parameters, see the Unix man page `socket (2)`.

See also **accept_connect()**, **bind()**, **connect()**, **listen()**, and **strerror()**.

strerror (PHP 4 >= 4.0.2)

Return a string describing a socket error.

```
string strerror (int errno)
```

strerror() takes as its *errno* parameter the return value of one of the socket functions, and returns the corresponding explanatory text. This makes it a bit more pleasant to figure out why something didn't work; for instance, instead of having to track down a system include file to find out what '-111' means, you just pass it to **strerror()**, and it tells you what happened.

Ejemplo 1. strerror() example

```
<?php
if (($socket = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
    echo "socket() failed: reason: " . strerror($socket) . "\n";
}

if (($ret = bind($socket, '127.0.0.1', 80)) < 0) {
    echo "bind() failed: reason: " . strerror($ret) . "\n";
}
?>
```

The expected output from the above example (assuming the script is not run with root privileges):

```
bind() failed: reason: Permission denied
```

See also **accept_connect()**, **bind()**, **connect()**, **listen()**, and **socket()**.

LXXIII. Funciones de cadenas

Todas estas funciones manipulan cadenas de varias maneras. En las secciones sobre expresiones regulares y manejo de URL se pueden encontrar secciones más especializadas.

AddCSlashes (PHP 4 >= 4.0b4)

Marca una cadena con barras al estilo del C

```
string addcslashes (string cad, string listcar)
```

Devuelve una cadena con barras invertidas antes de los caracteres listados en el parámetro *listcar*. También marca \n, \r etc. Al estilo del C, los caracteres con código ASCII inferior a 32 y superior a 126 son convertidos a representación octal. Tenga cuidado cuando marque caracteres alfanuméricos. Puede especificar un rango en *listcar* como el "\0..\37", que marcaría todos los caracteres con código ASCII entre 0 y 31.

Ejemplo 1. Ejemplo de Addcslashes()

```
$tradformado = addcslashes ($no_transf, "\0..\37!@\177..\377");
```

Nota: Añadida en PHP4b3-dev.

Vea también **stripcslashes()**, **stripslashes()**, **htmlspecialchars()**, **htmlspecialchars()**, y **quotemeta()**.

AddSlashes (PHP 3, PHP 4)

Marca una cadena con barras

```
string addslashes (string cad)
```

Devuelve una cadena con barras invertidas frente a los caracteres que necesitan marcarse en consultas de bases de datos, etc. Estos son la comilla simple ('), comilla doble ("), barra invertida (\) y NUL (el byte nulo).

Vea también **stripslashes()**, **htmlspecialchars()**, y **quotemeta()**.

bin2hex (PHP 3>= 3.0.9, PHP 4)

Convierte datos binarios en su representación hexadecimal

```
string bin2hex (string cad)
```

Devuelve una cadena ASCII que contiene la representación hexadecimal de *cad*. La conversión se realiza byte a byte, con los 4 bits superiores primero.

Chop (PHP 3, PHP 4)

Elimina espacios sobrantes al final

```
string chop (string cad)
```

Devuelve la cadena argumento sin los espacios sobrantes, incluyendo los saltos de línea.

Ejemplo 1. Ejemplo de Chop()

```
$recortada = chop ($linea);
```

Vea también **trim()**.

Chr (PHP 3, PHP 4)

Devuelve un carácter específico

```
string chr (int ascii)
```

Devuelve una cadena de un carácter que contiene el carácter especificado por *ascii*.

Ejemplo 1. Ejemplo de Chr()

```
$cad .= chr (27); /* añade un carácter de escape al final de $cad */
/* A veces esto es más útil */

$cad = sprintf ("La cadena termina en escape: %c", 27);
```

Esta función complementa a **ord()**. Vea también **sprintf()** con una cadena de formato %c.

chunk_split (PHP 3>= 3.0.6, PHP 4)

Divide una cadena en trozos más pequeños

```
string chunk_split (string cadena [, int tamanozo [, string final]])
```

Se puede utilizar para trocear una cadena en pedazos más pequeños, lo que es útil, p.ej., para convertir la salida de la función **base64_encode** a la semántica del RFC 2045. Inserta la cadena *final* cada *tamanozo* (por defecto vale 76) caracteres. Devuelve la nueva cadena y deja intacta la original.

Ejemplo 1. Ejemplo de Chunk_split()

```
# formatear $datos usando la semántica del RFC 2045

$nueva_cad = chunk_split (base64_encode($datos));
```

Esta función es notablemente más rápida que **ereg_replace()**.

Nota: Esta función se añadió en la 3.0.6.

convert_cyr_string (PHP 3>= 3.0.6, PHP 4)

Convierte de un juego de caracteres Cirílico a otro

```
string convert_cyr_string (string cad, string desde, string hasta)
```

Esta función convierte la cadena dada de un juego de caracteres Cirílico a otro. Los argumentos *desde* y *hasta* son caracteres sencillos que representan los juegos de caracteres Cirílicos fuente y destino. Los tipos soportados son:

- k - koi8-r
- w - windows-1251
- i - iso8859-5
- a - x-cp866
- d - x-cp866
- m - x-mac-cyrillic

count_chars (PHP 4 >= 4.0b4)

Devuelve información sobre los caracteres usados en una cadena

```
mixed count_chars (string cadena [, modo])
```

Cuenta el número de apariciones de cada valor de byte (0..255) en *cadena* y lo devuelve de varias maneras. El parámetro opcional *modo* vale por defecto 0. Dependiendo de *modo*, **count_chars()** puede devolver:

- 0 - una matriz con el valor del byte como clave y la frecuencia de cada uno como valor.
- 1 - como el 0, pero listando únicamente los valores de byte con frecuencia superior a cero.
- 2 - como el 0, pero listando únicamente los valores de byte con frecuencia igual a 0.
- 3 - se devuelve una cadena que contiene todos los valores de byte utilizados.
- 4 - se devuelve una cadena que contiene todos los valores de byte no utilizados.

Nota: Esta función se añadió en el PHP 4.0.

crc32 (PHP 4 >= 4.0.1)

Calcula el polinomio crc32 de una cadena

```
int crc32 (string cad)
```

Genera el polinomio de comprobación de redundancia cíclica de 32 bits de *cad*. Se suele utilizar para validar la integridad de los datos transmitidos.

Vea también: **md5()**

crypt (PHP 3, PHP 4)

Encripta una cadena mediante DES

```
string crypt (string cad [, string semilla])
```

crypt() encriptará una cadena utilizando el método estándar de encriptación del Unix DES. Los argumentos son una cadena a encriptar y una cadena semilla de 2 caracteres en la que basar la encriptación. Vea la página de manual de Unix sobre crypt para más información.

Si el argumento de semilla no se proporciona, será generado aleatoriamente por el PHP.

Algunos sistemas operativos soportan más de un tipo de encriptación. De hecho, algunas veces la encriptación estándar DES es sustituida por un algoritmo de encriptación basado en MD5. El tipo de encriptación es disparado por el argumento semilla. En tiempo de instalación, el PHP determina la capacidad de la función de encriptación y aceptará semillas para otros tipos de encriptación. Si no se proporciona la semilla, el PHP intentará generar una semilla estándar DES de 2 caracteres por defecto, excepto si el tipo de encriptación estándar del sistema es el MD5, en cuyo caso se generará una semilla aleatoria compatible con MD5. El PHP fija una constante llamada CRYPT_SALT_LENGTH que le especifica si su sistema soporta una semilla de 2 caracteres o si se debe usar la semilla de 12 caracteres del NDS.

La función estándar de encriptación **crypt()** contiene la semilla como los dos primeros caracteres de la salida.

En los sistemas en los que la función **crypt()** soporta múltiples tipos de encriptación, las siguientes constantes son fijadas a 0 ó 1 dependiendo de si está disponible el tipo dado:

- CRYPT_STD_DES - Encriptación DES estándar con semilla de 2 caracteres
- CRYPT_EXT_DES - Encriptación DES extendida con semilla de 9 caracteres
- CRYPT_MD5 - Encriptación MD5 con semilla de 12 caracteres y comenzando por \$1\$
- CRYPT_BLOWFISH - Encriptación DES extendida con semilla de 16 caracteres y comenzando por \$2\$

No hay función de desencriptado porque **crypt()** utiliza un algoritmo de una sola vía.

Vea también: **md5()**.

echo (unknown)

Da salida a una o más cadenas

```
echo (string arg1, string [argn]...)
```

Da salida a todos sus parámetros.

Echo() no es realmente una función (es una sentencia del lenguaje) de modo que no se requiere el uso de los paréntesis.

Ejemplo 1. Ejemplo de Echo()

```
echo "Hola Mundo";  
  
echo "Esto se extiende  
por varias líneas. Los saltos de línea  
también se envían";  
  
echo "Esto se extiende\npor varias líneas. Los saltos de línea\n\ttambién se envían";
```

Nota: De hecho, si desea pasar más de un parámetro a echo no debe encerrarlos entre paréntesis.

Vea también: **print()**, **printf()**, y **flush()**.

explode (PHP 3, PHP 4)

Divide una cadena por otra

```
array explode (string separador, string cadena [, int limite])
```

Devuelve una matriz de cadenas, cada una de las cuales es una subcadena de *cadena* formada mediante su división en las fronteras marcadas por la cadena *separador*. Si se especifica *limite*, la matriz devuelta contendrá un máximo de *limite* elementos con el último conteniendo el resto de la *cadena*.

Ejemplo 1. Ejemplo de Explode()

```
$pizza = "trozo1 trozo2 trozo3 trozo4 trozo5 trozo6";
$trozos = explode (" ", $pizza);
```

Vea también **split()** e **implode()**.

get_html_translation_table (PHP 4 >= 4.0b4)

Devuelve la tabla de traducción utilizada por **htmlspecialchars()** y **htmlentities()**

```
string get_html_translation_table (int tabla)
```

get_html_translation_table() devolverá la tabla de traducción que se usa internamente para **htmlspecialchars()** y **htmlentities()**. Hay dos nuevas definiciones (*HTML_ENTITIES*, *HTML_SPECIALCHARS*) que le permiten especificar la tabla deseada.

Ejemplo 1. Ejemplo de Tabla de Traducción

```
$trad = get_html_translation_table (HTML_ENTITIES);
$cad = "Hallo & <Frau> & Krämer";
$codif = strtr ($cad, $trad);
```

La variable *\$codif* contendrá ahora: "Hallo & <Frau> & Krämer".

Lo interesante es usar la función **array_flip()** para cambiar la dirección de la traducción.

```
$trad = array_flip ($trad);
$original = strtr ($cad, $trad);
```

El contenido de *\$original* sería: "Hallo & <Frau> & Krämer".

Nota: Esta función fue añadida en PHP 4.0.

Vea también: **htmlspecialchars()**, **htmlentities()**, **strtr()**, y **array_flip()**.

get_meta_tags (PHP 3>= 3.0.4, PHP 4)

Extrae todas las etiquetas meta de un archivo y retorna una matriz

```
array get_meta_tags (string nombre fich [, int use_ruta_include])
```

Abre el *nombre fich* y lo trocea línea a línea buscando etiquetas <meta> de la forma

Ejemplo 1. Ejemplo de Etiquetas Meta

```
<meta name="autor" content="nombre">
<meta name="etiquetas" content="documentación de php3">
</head> <!-- el proceso se detiene aquí -->
```

(preste atención a los finales de línea - el PHP utiliza una función nativa para trocear la entrada, de modo que un archivo de Mac no funcionará en Unix).

El valor de la propiedad name queda como clave y el valor de la propiedad content queda como el valor de la matriz devuelta, de modo que pueda usar fácilmente funciones estándar de matrices para recorrerla o para acceder a valores individuales. Los caracteres especiales en el valor de name son sustituidos por '_' y el resto es convertido a minúsculas.

Fijando *use_ruta_include* a 1 hará que el PHP intente abrir el archivo a través de la ruta de inclusión.

hebrev (PHP 3, PHP 4)

Convierte Hebreo lógico a texto visual

```
string hebrev (string texto_hebreo [, int max_cars_por_linea])
```

El parámetro opcional *max_cars_por_linea* indica el máximo número de caracteres que se emitirán por línea. La función intenta evitar cortar palabras.

Vea también **hebrevc()**

hebrevc (PHP 3, PHP 4)

Convierte Hebreo lógico a texto visual con conversión de saltos de línea

```
string hebrevc (string texto_hebreo [, int max_cars_por_linea])
```

Esta función es similar a **hebrev()** con la diferencia que convierte las nuevas líneas (\n) a "
\n". El parámetro opcional *max_cars_por_linea* indica el máximo número de caracteres que se emitirán por línea. La función intenta evitar cortar palabras.

Vea también **hebrev()**

htmlentities (PHP 3, PHP 4)

Convierte todos los caracteres aplicables a entidades HTML

```
string htmlentities (string cadena)
```

Esta función es del todo idéntica a **htmlspecialchars()**, excepto que traduce todos los caracteres que tienen equivalente como entidad HTML.

Actualmente se utiliza el juego de caracteres ISO-8859-1.

Vea también **htmlspecialchars()** y **nl2br()**.

htmlspecialchars (PHP 3, PHP 4)

Convierte caracteres especiales a entidades HTML

```
string htmlspecialchars (string cadena)
```

Ciertos caracteres tienen significados especiales en HTML, y deben ser representados por entidades HTML si se desea preservar su significado. Esta función devuelve una cadena con dichas conversiones realizadas.

Esta función es útil para evitar que el texto entrado por el usuario contenga marcas HTML, como ocurre en aplicaciones de foros o libros de visita.

Actualmente, las traducciones hechas son:

- '&' (ampersand) se convierte en '&'
- '""' (doble comilla) se convierte en '"'
- '<' (menor que) se convierte en '<'
- '>' (mayor que) se convierte en '>'

Nótese que esta función no traduce nada más que lo mostrado más arriba. Para una traducción de entidades completa, vea **htmlentities()**.

Vea también **htmlentities()** y **nl2br()**.

implode (PHP 3, PHP 4)

Unir elementos de una matriz mediante una cadena

```
string implode (string cola, array piezas)
```

Devuelve una cadena que contiene una representación de todos los elementos de la matriz en el mismo orden, pero con la cadena *cola* en medio de los mismos.

Ejemplo 1. Ejemplo de **Implode()**

```
$separada_dospuntos = implode (":", $matrizay);
```

Vea también **explode()**, **join()**, y **split()**.

join (PHP 3, PHP 4)

Une elementos de una tabla mediante una cadena

```
string join (string cola, array piezas)
```

join() es un alias para **implode()**, y es idéntica en todo.

Vea también **explode()**, **implode()**, y **split()**.

levenshtein (PHP 3>= 3.0.17, PHP 4 >= 4.0.1)

Calcula la distancia Levenshtein entre dos cadenas

```
int levenshtein (string cad1, string cad2)
```

Esta función devuelve la distancia Levenshtein entre las dos cadenas argumento, ó -1 si alguna de las cadenas tiene más de 255 caracteres.

La distancia Levenshtein se define como el mínimo número de caracteres que se tienen que sustituir, insertar o borrar para transformar *cad1* en *cad2*. La complejidad del algoritmo es $O(m*n)$, donde *n* y *m* son las longitudes de *cad1* y *cad2* (bastante bueno si se la compara con **similar_text()**, que es $O(\max(n,m)^{**}3)$, pero aún es cara).

Vea también **soundex()**, **similar_text()** y **metaphone()**.

ltrim (PHP 3, PHP 4)

Elimina el espacio en blanco del principio de una cadena

```
string ltrim (string cad)
```

Esta función elimina el espacio en blanco del principio de una cadena y devuelve la cadena resultante. Los caracteres de espacio que elimina realmente son: "\n", "\r", "\t", "\v", "\0", y el espacio en sí.

Vea también **chop()** y **trim()**.

md5 (PHP 3, PHP 4)

Calcula el hash md5 de una cadena

```
string md5 (string cad)
```

Calcula el hash (extracto) MD5 de *cad* usaneo el Algoritmo de Resumen de Mensajes MD5 de RSA Data Security, Inc. (<http://www.faqs.org/rfcs/rfc1321.html>).

Vea también: **crc32()**

Metaphone (PHP 4 >= 4.0b4)

Calcula la clave "metáfona" de una cadena

```
string metaphone (string cad)
```

Calcula la clave "metáfona" de *cad*.

Similarmente a **soundex()**, metaphone crea la misma clave para palabras que suenan parecidas. Es más precisa que la función **soundex()**, pues conoce las reglas básicas de la pronunciación del Inglés. Las claves metafónicas generadas son de longitud variable.

Metaphone fue desarrollado por Lawrence Philips <lphilips@verity.com>. Se describe en ["Practical Algorithms for Programmers", Binstock & Rex, Addison Wesley, 1995].

Nota: Esta función se añadió en PHP 4.0.

nl2br (PHP 3, PHP 4)

Convierte nuevas líneas a saltos de línea HTML

```
string nl2br (string cadena)
```

Devuelve la *cadena* con '
' insertados antes de cada nueva línea.

Vea también **htmlspecialchars()**, **htmlentities()** y **wordwrap()**.

Ord (PHP 3, PHP 4)

Devuelve el valor ASCII de un carácter

```
int ord (string cadena)
```

Devuelve el valor ASCII del primer carácter de *cadena*. Esta función complementa a **chr()**.

Ejemplo 1. Ejemplo de Ord()

```
if (ord ($cad) == 10) {
    echo "El primer carácter de \$cad es un salto de línea.\n";
}
```

Vea también **chr()**.

parse_str (PHP 3, PHP 4)

Divide la cadena en variables

```
void parse_str (string cad)
```

Divide *cad* como si fuera la cadena de consulta enviada por un URL y crea las variables en el ámbito actual.

Ejemplo 1. Usando parse_str()

```
$cad = "primero=valor&segundo[ ]=esto+funciona&segundo[ ]=otro";
parse_str($cad);
echo $primero;      /* escribe "valor" */
echo $segundo[0];   /* escribe "esto funciona" */
echo $segundo[1];   /* escribe "otro" */
```

print (unknown)

Emite una cadena

```
print (string arg)
```

Emite *arg*.

Vea también: **echo()**, **printf()**, y **flush()**.

printf (PHP 3, PHP 4)

Emite una cadena con formato

```
int printf (string formato [, mixed args...])
```

Produce una salida según el *formato*, que es descrito en la documentación para **sprintf()**.

Vea también: **print()**, **sprintf()**, **sscanf()**, **fscanf()**, y **flush()**.

quoted_printable_decode (PHP 3>= 3.0.6, PHP 4)

Convierte una cadena con marcación imprimible a una cadena de 8 bits

```
string quoted_printable_decode (string cad)
```

Esta función devuelve una cadena binaria de 8 bit que se corresponde con la cadena con marcación imprimible decodificada. Esta función es similar a **imap_qprint()**, pero sin requerir que el módulo IMAP funcione.

quotemeta (PHP 3, PHP 4)

Quote meta characters

```
string quotemeta (string cad)
```

Devuelve una versión de la cadena con una barra invertida (\) antes de cada carácter de este conjunto:

```
. \ \ + * ? [ ^ ] ( $ )
```

Vea también **addslashes()**, **htmlentities()**, **htmlspecialchars()**, **nl2br()**, y **stripslashes()**.

rtrim (PHP 3, PHP 4)

Elimina espacios en blanco al final de la cadena.

```
string rtrim (string cad)
```

Devuelve la cadena argumento sin espacios en blanco ni saltos de línea al final. Es un alias para **chop()**.

Ejemplo 1. Ejemplo de rtrim()

```
$recortada = rtrim ($linea);
```

Vea también **trim()**, **ltrim()**.

sscanf (PHP 4 >= 4.0.1)

Trocea la entrada desde una cadena según un formato dado

```
mixed sscanf (string cad, string formato [, string var1...])
```

La función **sscanf()** es la función de entrada análoga de **printf()**. **sscanf()** lee del parámetro de cadena *cad* y lo interpreta según el *formato* especificado. Si sólo se pasan dos parámetros a esta función, los valores devueltos se harán en una matriz.

Ejemplo 1. Ejemplo de sscanf()

```
// obteniendo el número de serie
$numserie = sscanf("SN/2350001","SN/%d");
// y la fecha de fabricación
$fecha = "01 Enero 2000";
list($dia, $mes, $anno) = sscanf($fecha,"%d %s %d");
echo "El objeto $numserie fue fabricado el: $anno-$mes,$dia\n";
```

Si se pasan los parámetros opcionales, la función devolverá el número de valores asignados. Los parámetros opcionales deben ser pasados por referencia.

Ejemplo 2. Ejemplo de sscanf() - usando parámetros opcionales

```
// obtener autor y generar la ficha DocBook
$autor = "24\tLewis Carroll";
$n = sscanf($autor,"%d\t%s %s", &$id, &$nombre, &$apell);
echo "<autor id='".$id'>
      <firstname>$nombre</firstname>
      <surname>$apell</surname>
```

```
</author>\n" ;
```

Vea también: **fscanf()**, **printf()**, y **sprintf()**.

setlocale (PHP 3, PHP 4)

Fija la información de localidad

```
string setlocale (string categoria, string localidad)
```

categoria es una cadena que especifica la categoría de las funciones afectadas por el ajuste de localidad:

- LC_ALL para todas las funciones
- LC_COLLATE para la comparación de cadenas - aún no incluída en el PHP
- LC_CTYPE para la conversión y clasificación de caracteres, como por ejemplo **strtoupper()**
- LC_MONETARY para localeconv() - aún no incluída en el PHP
- LC_NUMERIC para el separador decimal
- LC_TIME para el formato de fecha y hora con **strftime()**

Si *localidad* es la cadena vacía " ", los nombres de localidad se fijarán a partir de las variables de entorno con los mismos nombres de las categorías anteriores, o desde "LANG".

Si la localidad es cero o "0", el ajuste de localidad no se ve afectado y sólo se devuelve el ajuste actual.

setlocale devuelve la nueva localidad, o false si la funcionalidad de localización no está disponible en la plataforma, la localidad especificada no existe o el nombre de categoría no es válido. Un nombre de categoría no válido también produce un mensaje de aviso.

similar_text (PHP 3>= 3.0.7, PHP 4 >= 4.0b2)

Calcula la similitud entre dos cadenas

```
int similar_text (string primera, string segunda [, double porcentaje])
```

Esta función calcula la similitud entre dos cadenas según se describe en Oliver [1993]. Nótese que esta implementación no utiliza una pila como en el pseudo-código de Oliver, sino llamadas recursivas que pueden o no acelerar el proceso completo. Nótese también que la complejidad de este algoritmo es O(N**3), donde N es la longitud de la cadena más larga.

Pasando una referencia como tercer argumento, **similar_text()** calculará para usted la similitud como porcentaje. Devuelve el número de caracteres coincidentes en ambas cadenas.

soundex (PHP 3, PHP 4)

Calcula la clave soundex de una cadena

```
string soundex (string cad)
```

Calcula la clave soundex de *cad*.

Las claves soundex tienen la propiedad de que las palabras que se pronuncian de forma parecida tienen la misma clave, de modo que se pueden usar para simplificar la búsqueda en las bases de datos cuando se conoce la pronunciación pero no la transcripción. Esta función soundex devuelve una cadena de 4 caracteres que comienza por una letra.

Esta función soundex en particular es la descrita por Donald Knuth en "The Art Of Computer Programming, vol. 3: Sorting And Searching", Addison-Wesley (1973), pp. 391-392.

Ejemplo 1. Ejemplos de Soundex

```
soundex ("Euler") == soundex ("Ellery") == 'E460';
soundex ("Gauss") == soundex ("Ghosh") == 'G200';
soundex ("Knuth") == soundex ("Kant") == 'H416';
soundex ("Lloyd") == soundex ("Ladd") == 'L300';
soundex ("Lukasiewicz") == soundex ("Lissajous") == 'L222';
```

sprintf (PHP 3, PHP 4)

Devuelve una cadena con formato

```
string sprintf (string formato [, mixed args...])
```

Devuelve una cadena producida de acuerdo a la cadena de *formato*.

La cadena de formato está compuesta por cero o más directivas: caracteres ordinarios (excepto %) que son copiados directamente al resultado, y *especificaciones de conversión*, cada una de las cuales provoca la obtención de su propio parámetro. Esto se aplica tanto a **sprintf()** como a **printf()**.

Cada especificación de conversión consiste en uno de estos elementos, por orden:

1. Un *especificador de relleno* opcional que indica qué carácter se utilizará para llenar el resultado hasta el tamaño de cadena correcto. Este puede ser un espacio o un 0 (caracter cero). El valor por defecto es llenar con espacios. Un carácter de relleno alternativo se puede especificar prefijándolo con una comilla simple (''). Vea los ejemplos más abajo.
2. Un *especificador de alineación* opcional que indica si el resultado debe ser alineado a la izquierda o a la derecha. Por defecto se alinea a la derecha; un carácter – aquí lo justificará a la izquierda.
3. Un número opcional, un *especificador de ancho* que dice el número de caracteres (mínimo) en que debería resultar esta conversión.
4. Un *especificador de precisión* opcional que indica cuántos dígitos decimales deben mostrarse para los números en coma flotante. Esta opción no tiene efecto para otros tipos que no sean double. (Otra función útil para formatear números es **number_format()**).
5. Un *especificador de tipo* que indica el tipo a usar para tratar los datos de los argumentos. Los tipos posibles son:
 - % - un carácter literal de porcentaje. No se precisa argumento.
 - b - el argumento es tratado como un entero y presentado como un número binario.
 - c - el argumento es tratado como un entero, y presentado como el carácter con dicho valor ASCII.
 - d - el argumento es tratado como un entero y presentado como un número decimal.
 - f - el argumento es tratado como un doble y presentado como un número de coma flotante.
 - o - el argumento es tratado como un entero, y presentado como un número octal.
 - s - el argumento es tratado como una cadena y es presentado como tal.

x - el argumento es tratado como un entero y presentado como un número hexadecimal (con minúsculas)

X - el argumento es tratado como un entero y presentado como un número hexadecimal (con mayúsculas)

Vea también: **printf()**, **sscanf()**, **fscanf()**, y **number_format()**.

Ejemplo 1. Ejemplo de Sprintf(): enteros llenados con ceros

```
$fechaiso = sprintf ("%04d-%02d-%02d", $anno, $mes, $dia);
```

Ejemplo 2. Ejemplo de Sprintf(): formateando monedas

```
$pelas1 = 68.75;
$pelas2 = 54.35;
$pelas = $pelas1 + $pelas2;
// echo $pelas mostrará "123.1";
$formateado = sprintf ("%01.2f", $pelas);
// echo $formateado mostrará "123.10"
```

strcasecmp (PHP 3>= 3.0.2, PHP 4)

Comparación de cadenas insensible a mayúsculas y minúsculas y segura en modo binario

```
int strcasecmp (string cad1, string cad2)
```

Devuelve < 0 si *cad1* es menor que *cad2*; > 0 si *cad1* es mayor que *cad2*, y 0 si son iguales.

Ejemplo 1. Ejemplo de strcasecmp()

```
$var1 = "Hello";
$var2 = "hello";
if (!strcasecmp ($var1, $var2)) {
    echo '$var1 es igual a $var2 en una comparación sin tener en cuenta '
        .'mayúsculas o minúsculas';
}
```

Vea también **ereg()**, **strcmp()**, **substr()**, **stristr()**, y **strrstr()**.

strchr (PHP 3, PHP 4)

Encuentra la primera aparición de un carácter

```
string strchr (string pajar, string aguja)
```

Esta función es un alias para **strrstr()**, y es idéntica en todo.

strcmp (PHP 3, PHP 4)

Comparación de cadenas con seguridad binaria

```
int strcmp (string cad1, string cad2)
```

Devuelve < 0 si *cad1* es menor que *cad2*; > 0 si *cad1* es mayor que *cad2*, y 0 si son iguales.

Nótese que esta comparación es sensible a mayúsculas y minúsculas.

Vea también **ereg()**, **strcasecmp()**, **substr()**, **stristr()**, **strcmp()**, y **strrstr()**.

strcspn (PHP 3>= 3.0.3, PHP 4)

Encuentra la longitud del elemento inicial que no coincide con la máscara

```
int strcspn (string cad1, string cad2)
```

Devuelve la longitud del segmento inicial de *cad1* que *no* contiene ninguno de los caracteres de *cad2*.

Vea también **strspn()**.

strip_tags (PHP 3>= 3.0.8, PHP 4 >= 4.0b2)

Elimina marcas HTML y PHP de una cadena

```
string strip_tags (string cad [, string etiq_permitidas])
```

Esta función intenta eliminar todas las etiquetas HTML y PHP de la cadena dada. Causa error por precaución en caso de etiquetas incompletas o falsas. Utiliza la misma máquina de estados para eliminar las etiquetas que la función **fgetss()**.

Puede usar el parámetro opcional para especificar las etiquetas que no deben eliminarse.

Nota: *etiq_permitidas* fue añadido en PHP 3.0.13, PHP4B3.

stripcslashes (PHP 4 >= 4.0b4)

Desmarca la cadena marcada con **addcslashes()**

```
string stripcslashes (string cad)
```

Devuelve una cadena con las barras invertidas eliminadas. Reconoce las marcas tipo C \n, \r ..., y la representación octal y hexadecimal.

Nota: Añadida en PHP4b3-dev.

Vea también **addcslashes()**.

stripslashes (PHP 3, PHP 4)

Desmarca la cadena marcada con **addslashes()**

```
string stripslashes (string cad)
```

Devuelve una cadena con las barras invertidas eliminadas (\ ' se convierte en ', etc.). Las barras invertidas dobles se convierten en sencillas.

Vea también **addslashes()**.

stristr (PHP 3>= 3.0.6, PHP 4)

stristr() sin tener en cuenta mayúsculas o minúsculas

```
string stristr (string pajar, string aguja)
```

Devuelve todo el *pajar* desde la primera aparición de la *aguja*, siendo el *pajar* examinado sin tener en cuenta mayúsculas o minúsculas.

Si la *aguja* no se encuentra, devuelve false.

Si la *aguja* no es una cadena, es convertida a entero y usada como código de un carácter ASCII.

Vea también **strchr()**, **strrchr()**, **substr()**, y **ereg()**.

strlen (PHP 3, PHP 4)

Obtiene la longitud de la cadena

```
int strlen (string cad)
```

Devuelve la longitud de la *cadena*.

strnatcmp (PHP 4 >= 4.0RC2)

Compara cadenas usando un algoritmo de "orden natural"

```
int strnatcmp (string cad1, string cad2)
```

Esta función implementa un algoritmo de comparación que ordena las cadenas alfanuméricas como lo haría un ser humano, que es lo que se denomina "orden natural". A continuación se puede ver un ejemplo de la diferencia entre este algoritmo y los algoritmos de ordenación de cadenas habituales en los ordenadores (utilizados en **strcmp()**):

```
$matriz1 = $matriz2 = array ("img12.png", "img10.png", "img2.png", "img1.png");
echo "Comparación de cadenas estándar\n";
usort($matriz1, "strcmp");
print_r($matriz1);
```

```
echo "\nComparación de cadenas en orden natural\n";
usort($matriz2,"strnatcmp");
print_r($matriz2);
```

El código anterior generará la siguiente salida:

```
Comparación de cadenas estándar
Array
(
    [0] => img1.png
    [1] => img10.png
    [2] => img12.png
    [3] => img2.png
)

Comparación de cadenas en orden natural
Array
(
    [0] => img1.png
    [1] => img2.png
    [2] => img10.png
    [3] => img12.png
)
```

Para más información, vea la página de Martin Pool sobre Comparación de Cadenas en Orden Natural (<http://www.linuxcare.com.au/projects/natsort/>).

De forma similar a otras funciones de comparación de cadenas, esta devuelve < 0 si *cad1* es menor que *cad2*; > 0 si *cad1* es mayor que *cad2*, y 0 si son iguales.

Nótese que esta comparación es sensible a mayúsculas y minúsculas.

Vea también **ereg()**, **strcasecmp()**, **substr()**, **stristr()**, **strcmp()**, **strncmp()**, **strnatcasecmp()**, y **strrstr()**.

strnatcasecmp (PHP 4 >= 4.0RC2)

Comparación de cadenas insensible a mayúsculas y minúsculas usando un algoritmo de "orden natural"

```
int strnatcasecmp (string cad1, string cad2)
```

Esta función implementa un algoritmo de comparación que ordena las cadenas alfanuméricas como lo haría un ser humano. El comportamiento de esta función es similar a **strnatcmp()**, pero la comparación no es sensible a mayúsculas y minúsculas. Para más información, vea la página de Martin Pool sobre Comparación de Cadenas en Orden Natural (<http://www.linuxcare.com.au/projects/natsort/>).

De forma similar a otras funciones de comparación de cadenas, esta devuelve < 0 si *cad1* es menor que *cad2*; > 0 si *cad1* es mayor que *cad2*, y 0 si son iguales.

Vea también **ereg()**, **strcasecmp()**, **substr()**, **stristr()**, **strcmp()**, **strncmp()**, **strnatcmp()**, y **strrstr()**.

strncmp (PHP 4 >= 4.0b4)

Comparación de los *n* primeros caracteres de cadenas, con seguridad binaria

```
int strncmp (string cad1, string cad2, int largo)
```

Esta función es similar a **strcmp()**, con la diferencia que se puede especificar el (límite superior del) número de caracteres (*largo*) de cada cadena que se usarán en la comparación. Si alguna de las cadenas es menor que el *largo*, se usará su longitud para la comparación.

Devuelve < 0 si *cad1* es menor que *cad2*; > 0 si *cad1* es mayor que *cad2*, y 0 si son iguales.

Nótese que esta comparación es sensible a mayúsculas y minúsculas.

Vea también **ereg()**, **strcasecmp()**, **substr()**, **stristr()**, **strcmp()**, y **strrstr()**.

str_pad (PHP 4 >= 4.0.1)

Rellena una cadena con otra hasta una longitud dada

```
string str_pad (string entrada, int tama_relleno [, string cad_relleno [, int tipo_relleno]])
```

Esta función rellena la cadena *entrada* por la derecha, la izquierda o por ambos lados hasta el largo indicado. Si no se especifica el argumento opcional *cad_relleno*, *entrada* es rellenada con espacios. En caso contrario, será rellenada con los caracteres de *cad_relleno* hasta el límite.

El argumento opcional *tipo_relleno* puede valer STR_PAD_RIGHT, STR_PAD_LEFT, o STR_PAD_BOTH. Si no se especifica, se asume que vale STR_PAD_RIGHT.

Si el valor de *tama_relleno* es negativo o menor que la longitud de la cadena de entrada, no se produce relleno alguno.

Ejemplo 1. Ejemplo de str_pad()

```
$entrada = "Alien";
print str_pad($entrada, 10);           // produce "Alien      "
print str_pad($entrada, 10, "-=", STR_PAD_LEFT); // produce "----Alien"
print str_pad($entrada, 10, "__", STR_PAD_BOTH); // produce "__Alien__"
```

strpos (PHP 3, PHP 4)

Encuentra la posición de la primera aparición de una cadena

```
int strpos (string pajar, string aguja [, int desplazamiento])
```

Devuelve la posición numérica de la primera aparición de la *aguja* en la cadena *pajar*. A diferencia de **strrpos()**, esta función puede tomar una cadena completa como *aguja* y se utilizará en su totalidad.

Si la *aguja* no es hayada, devuelve false.

Nota: Es fácil confundir los valores de retorno para "caracter encontrado en la posición 0"y "caracter no encontrado". Aquí se indica cómo detectar la diferencia:

```
// en PHP 4.0b3 y posteriores:
$pos = strpos ($micadena, "b");
if ($pos === false) { // nota: tres signos igual
    // no encontrado ...
}
```

```
// en versiones anteriores a la 4.0b3:
$pos = strpos ($micadena, "b");
if (is_string ($pos) && !$pos) {
    // no encontrado ...
}
```

Si la *aguja* no es una cadena, se convierte a entero y se aplica como el valor ordinal de un carácter.

El parámetro opcional *desplazamiento* le permite especificar a partir de qué carácter del *pajar* comenzar a buscar. La posición devuelta es aún relativa al comienzo de *pajar*.

Vea también **strrpos()**, **strrchr()**, **substr()**, **stristr()**, y **strrstr()**.

strrchr (PHP 3, PHP 4)

Encuentra la última aparición de un carácter en una cadena

```
string strrchr (string pajar, string aguja)
```

Esta función devuelve la porción del *pajar* que comienza en la última aparición de la *aguja* y continúa hasta el final del *pajar*.

Devuelve false si la *aguja* no es hallada.

Si la *aguja* contiene más de un carácter, sólo se usará el primero.

Si la *aguja* no es una cadena, se convierte a entero y se aplica como el valor ordinal de un carácter.

Ejemplo 1. Ejemplo de Strrchr()

```
// obtener el último directorio de $PATH
$dir = substr (strrchr ($PATH, ":"), 1);

// obtener todo tras el último salto de línea
$texto = "Line 1\nLine 2\nLine 3";
$apell = substr (strrchr ($texto, 10), 1 );
```

Vea también **substr()**, **stristr()**, y **strrstr()**.

str_repeat (PHP 4 >= 4.0b4)

Repite una cadena

```
string str_repeat (string cad_entrada, int veces)
```

Devuelve la *cad_entrada* repetida *veces*. *veces* debe ser mayor que 0.

Ejemplo 1. Ejemplo de Str_repeat()

```
echo str_repeat ("-=", 10);
```

Esto mostrará "=====-====-".

Nota: Esta función fue añadida en el PHP 4.0.

strrev (PHP 3, PHP 4)

Invierte una cadena

```
string strrev (string cadena)
```

Devuelve la *cadena* invertida.

strrpos (PHP 3, PHP 4)

Encuentra la posición de la última aparición de un carácter en una cadena

```
int strrpos (string pajar, char aguja)
```

Devuelve la posición numérica de la última aparición de la *aguja* en el *pajar*. Nótese que la aguja en este caso sólo puede ser un carácter único. Si se pasa una cadena como aguja, sólo se utilizará el primer carácter de la misma.

Si la *aguja* no es encontrada, devuelve false.

Si la *aguja* no es una cadena, se convierte a entero y se aplica como el valor ordinal de un carácter.

Vea también **strpos()**, **strrchr()**, **substr()**, **stripos()**, y **stristr()**.

strspn (PHP 3>= 3.0.3, PHP 4)

Encuentra la longitud del segmento inicial que coincide con la máscara

```
int strspn (string cad1, string cad2)
```

Devuelve la longitud del segmento inicial de *cad1* que consiste por enteros en caracteres contenidos en *cad2*.

```
strspn ("42 es la respuesta. ¿Cuál es la pregunta ...?", "1234567890");
```

devolverá 2 como resultado.

Vea también **strcspn()**.

strstr (PHP 3, PHP 4)

Encuentra la primera aparición de una cadena

```
string strstr (string pajar, string aguja)
```

Devuelve todo el *pajar* desde la primera aparición de la *aguja* hasta el final.

Si la *aguja* no es hayada, devuelve false.

Si la *aguja* no es una cadena, se convierte a entero y se aplica como el valor ordinal de un carácter.

Nota: Nótese que esta función es sensible a mayúsculas y minúsculas. Para búsquedas no sensibles, utilice **stristr()**.

Ejemplo 1. Ejemplo de Strstr()

```
$email = 'sterling@designmultimedia.com';
$dominio = strstr ($email, '@');
print $dominio; // imprime @designmultimedia.com
```

Vea también **stristr()**, **strrchr()**, **substr()**, y **ereg()**.

strtok (PHP 3, PHP 4)

Divide una cadena en elementos

```
string strtok (string arg1, string arg2)
```

strtok() se usa para dividir en elementos una cadena. Es decir, que si tiene una cadena como "Esta es una cadena de ejemplo" podría dividirla en palabras individuales utilizando el espacio como divisor.

Ejemplo 1. Ejemplo de Strtok()

```
$cadena = "Esta es una cadena de ejemplo";
$tok = strtok ($cadena, " ");
while ($tok) {
    echo "Palabra=$tok<br>";
    $tok = strtok (" ");
}
```

Nótese que sólo la primera llamada a strtok utiliza el argumento cadena. Cada llamada subsiguiente necesita sólo el divisor a utilizar, puesto que ella guarda la posición actual en la cadena. Para comenzar de nuevo o para dividir otra cadena, simplemente llame a strtok con el argumento de cadena y se inicializará. Nótese que puede poner divisores múltiples como parámetro. La cadena será dividida cuando alguno de los caracteres del argumento sea hallado.

Además tenga cuidado si sus divisores valen "0", pues evalúa como false en las expresiones condicionales.

Vea también **split()** y **explode()**.

strtolower (PHP 3, PHP 4)

Pasa a minúsculas una cadena

```
string strtolower (string cad)
```

Devuelve la *cadena* con todas sus letras en minúsculas.

Nótese que las letras son definidas por el locale actual. Esto quiere decir que, por ejemplo, en el locale por defecto ("C"), los caracteres como la Ñ no serán convertidos.

Ejemplo 1. Ejemplo de Strtolower()

```
$cad = "María Tenía Un Corderito al que QUERÍA Mucho";
$cad = strtolower($cad);
print $cad; # Visualiza maría tenía un corderito al que quería mucho
```

Vea también **strtoupper()** y **ucfirst()**.

strtoupper (PHP 3, PHP 4)

Pasa a mayúsculas una cadena

```
string strtoupper (string cadena)
```

Devuelve la *cadena* con todas sus letras en mayúsculas.

Nótese que las letras son definidas por el locale actual. Esto quiere decir que, por ejemplo, en el locale por defecto ("C"), los caracteres como la ñ no serán convertidos.

Ejemplo 1. Ejemplo de Strtoupper()

```
$cad = "María Tenía Un Corderito al que QUERÍA Mucho";
$cad = strtoupper ($cad);
print $cad; # Visualiza MARÍA TENÍA UN CORDERITO AL QUE QUERÍA MUCHO
```

Vea también **strtolower()** and **ucfirst()**.

str_replace (PHP 3>= 3.0.6, PHP 4)

Sustituye todas las apariciones de la *aguja* en el *pajar* por la *cad*

```
string str_replace (string aguja, string cad, string pajar)
```

Esta función sustituye todas las apariciones de la *aguja* en el *pajar* por la *cad* dada. Si no precisa reglas especiales de sustitución, deberá usar siempre esta función en lugar de **ereg_replace()**.

Ejemplo 1. Ejemplo de Str_replace()

```
$bodytag = str_replace ("%cuerpo%", "negro", "<body text=%cuerpo%>");
```

Esta función tiene seguridad binaria.

Nota: **Str_replace()** fue añadida en PHP 3.0.6, pero tuvo errores hasta el PHP 3.0.8.

Vea también **ereg_replace()** y **strtr()**.

strtr (PHP 3, PHP 4)

Traduce ciertos caracteres

```
string strtr (string cad, string desde, string hasta)
```

Esta función trabaja sobre *cad*, traduciendo todas las apariciones de cada carácter en *desde* por el carácter correspondiente en *hasta* y devolviendo el resultado.

Si *desde* y *hasta* son de distinta longitud, los caracteres extra en la más larga son ignorados.

Ejemplo 1. Ejemplo de Strtr()

```
$addr = strtr($addr, "äåö", "aao");
```

strtr() puede llamarlo sólo con dos argumentos. Si se llama de esta manera, se comporta de otro modo: *desde* debe ser entonces una matriz que contenga pares cadena -> cadena que serán sustituidos en la cadena fuente. **strtr()** siempre buscará la coincidencia más larga primero y *NO* intentará sustituir nada en lo que haya trabajado ya.

Ejemplos:

```
$trad = array ("hola" => "hey", "hey" => "hola");
echo strtr("hey a todos, dije hola", $trad) . "\n";
```

Mostrará: "hola a todos, dije hey",

Nota: Esta característica (2 argumentos) fue añadida en el PHP 4.0

Vea también **ereg_replace()**.

substr (PHP 3, PHP 4)

Devuelve parte de una cadena

```
string substr (string cadena, int comienzo [, int largo])
```

substr devuelve la porción de *cadena* especificada por los parámetros *comienzo* y *largo*.

Si *comienzo* es positivo, la cadena devuelta comenzará en dicho carácter de *cadena*.

Ejemplos:

```
$resto = substr ("abcdef", 1);      // devuelve "bcdef"
$resto = substr ("abcdef", 1, 3); // devuelve "bcd"
```

Si *comienzo* es negativo, la cadena devuelta comenzará en dicha posición desde el final de *cadena*.

Ejemplos:

```
$resto = substr ("abcdef", -1);      // devuelve "f"
$resto = substr ("abcdef", -2);      // devuelve "ef"
$resto = substr ("abcdef", -3, 1);   // devuelve "d"
```

Si se especifica *largo* y es positivo, la cadena devuelta terminará *largo* caracteres tras el *comienzo*. Si esto resulta en una cadena con longitud negativa (porque el comienzo está pasado el final de la cadena), la cadena devuelta contendrá únicamente el carácter que haya en *comienzo*.

Si se especifica *largo* y es negativo, la cadena devuelta terminará a *largo* caracteres desde el final de *cadena*. Si esto resulta en una cadena con longitud negativa, la cadena devuelta contendrá únicamente el carácter que haya en *comienzo*.

Examples:

```
$resto = substr ("abcdef", 1, -1); // devuelve "bcde"
```

Vea también **strrchr()** y **ereg()**.

substr_count (PHP 4 >= 4.0RC2)

Cuenta el número de apariciones de la subcadena

```
int substr_count (string pajar, string aguja)
```

substr_count() devuelve el número de veces que la subcadena *aguja* se encuentra en la cadena *pajar*.

Ejemplo 1. Ejemplo de **substr_count()**

```
print substr_count("This is a test", "is"); // prints out 2
```

substr_replace (PHP 4 >= 4.0b4)

Sustituye texto en una parte de una cadena

```
string substr_replace (string cadena, string sustituto, int comienzo [, int largo])
```

substr_replace() sustituye la parte de *cadena* delimitada por los parámetros *comienzo* y (opcionalmente) *largo* por la cadena dada en *sustituto*. Se devuelve el resultado.

Si *comienzo* es positivo, la sustitución comenzará en dicha posición dentro de la *cadena*.

Si *comienzo* es negativo, la sustitución comenzará en dicha posición pero contando desde el final de *cadena*.

Si se especifica el *largo* y es positivo, representa el largo de la porción de *cadena* a sustituir. Si es negativo, representa el número de caracteres desde el final de *cadena* en los que dejar de sustituir. Si no se especifica, valdrá por defecto *strlen(cadena)*; es decir, que acabará la sustitución al final de *cadena*.

Ejemplo 1. Ejemplo de Substr_replace()

```
<?php
$var = 'ABCDEFGHIJKLMNPQR/';
echo "Original: $var<br>\n";

/* Estos dos ejemplos sustituyen toda $var por 'bob'. */
echo substr_replace ($var, 'bob', 0) . "<br>\n";
echo substr_replace ($var, 'bob', 0, strlen ($var)) . "<br>\n";

/* Inserta 'bob' justo al inicio de $var. */
echo substr_replace ($var, 'bob', 0, 0) . "<br>\n";

/* Los dos siguientes cambian 'MNRPQR' en $var por 'bob'. */
echo substr_replace ($var, 'bob', 10, -1) . "<br>\n";
echo substr_replace ($var, 'bob', -7, -1) . "<br>\n";

/* Borrar 'MNRPQR' de $var. */
echo substr_replace ($var, "", 10, -1) . "<br>\n";
?>
```

Vea también **str_replace()** y **substr()**.

Nota: **Substr_replace()** fue añadida en el PHP 4.0.

trim (PHP 3, PHP 4)

Elimina espacios del principio y final de una cadena

```
string trim (string cad)
```

Esta función elimina los espacios en blanco del comienzo y del final de una cadena y devuelve el resultado. Los caracteres de espacio que elimina realmente son: "\n", "\r", "\t", "\v", "\0", y el espacio en sí.

Vea también **chop()** y **ltrim()**.

ucfirst (PHP 3, PHP 4)

Pasar a mayúsculas el primer carácter de una cadena

```
string ucfirst (string cad)
```

Pone en mayúsculas el primer carácter de *cad* si es alfabético.

Nótese que 'alfabético' está determinado por la localidad actual. Por ejemplo, en la localidad por defecto "C", los caracteres como la a con diéresis (ä) no serán convertidos.

Ejemplo 1. Ejemplo de Ucfirst()

```
$texto = 'susanita tiene un ratón, un ratón chiquitín.';
$texto = ucfirst ($texto); // $texto vale ahora: Susanita tiene un
// ratón, un ratón chiquitín.
```

Vea también **strtoupper()** y **strtolower()**

ucwords (PHP 3>= 3.0.3, PHP 4)

Pone en mayúsculas el primer carácter de cada palabra de una cadena

```
string ucwords (string cad)
```

Pasa a mayúsculas la primera letra de cada palabra en *cad* si dicho carácter es alfabético.

Ejemplo 1. Ejemplo de ucwords()

```
$texto = "susanita tiene un ratón, un ratón chiquitín.";
$texto = ucwords($texto); // $texto vale ahora: Susanita Tiene Un
                           // Ratón, Un Ratón Chiquitín.
```

Vea también **strtoupper()**, **strtolower()** y **ucfirst()**.

wordwrap (PHP 4 >= 4.0.2)

Corta una cadena en un número dado de caracteres usando un carácter de ruptura de cadenas.

```
string wordwrap (string cad [, int ancho [, string ruptura]])
```

Corta la cadena *cad* en la columna especificada por el parámetro (opcional) *ancho*. La línea se rompe utilizando el parámetro (opcional) *ruptura*.

wordwrap() automáticamente cortará en la columna 75 y usará '\n' (nueva línea) si no se especifican el *ancho* o la *ruptura*.

Ejemplo 1. Ejemplo de wordwrap()

```
$texto = "El veloz murciélagos hindú comía feliz cardillo y kiwi.";
$textonuevo = wordwrap( $texto, 20 );

echo "$textonuevo\n";
```

Este ejemplo mostraría:

```
El veloz murciélagos
hindú comía feliz cardillo y kiwi.
```

Vea también **nl2br()**.

LXXIV. Funciones de Sybase

sybase_affected_rows (PHP 3>= 3.0.6, PHP 4)

obtiene el número de filas afectadas por la última consulta

```
int sybase_affected_rows ([int link_identifier])
```

Devuelve: El número de filas afectadas por la última consulta.

sybase_affected_rows() devuelve el número de filas afectadas por la última acción e tipo INSERT, UPDATE o DELETE en el servidor asociado al identificador de enlace especificado. Si no se especifica un identificador de enlace, se asume el último enlace abierto.

Esta instrucción no es efectiva para sentencias de tipo SELECT, sólo en sentencias que modifican registros. Para obtener el número de filas afectadas por un SELECT, use **sybase_num_rows()**.

Nota: Esta función sólo está disponible usando el interface de la librería CT, y no con la librería DB.

sybase_close (PHP 3, PHP 4)

cierra una conexión Sybase

```
int sybase_close (int link_identifier)
```

Devuelve: true si lo consigue, false ante un error

sybase_close() cierra el enlace a la base de datos Sybase asociada con el identificador de enlace especificado. Si no se especifica un identificador de enlace, se asume el último enlace abierto.

Note que esto no es necesario usualmente, ya que los enlaces no persistentes abiertos son cerrados automáticamente al final de la ejecución del script.

sybase_close() no cerrará enlaces persistentes generados por **sybase_pconnect()**.

Vea también: **sybase_connect()**, **sybase_pconnect()**.

sybase_connect (PHP 3, PHP 4)

abre una conexión con un servidor Sybase

```
int sybase_connect (string servername, string username, string password)
```

Devuelve: Un identificador de enlace Sybase positivo, o false ante un error.

sybase_connect() establece una conexión con un servidor Sybase. El parámetro *servername* tiene que ser un nombre de servidor válido que está definido en el fichero 'interfaces'.

En el caso que se haga una segunda llamada a **sybase_connect()** con los mismos argumentos, no se establecerá un nuevo enlace, en vez de esto, se devolverá el identificador de enlace que ya está abierto.

El enlace al servidor será cerrado tan pronto como la ejecución del script finalice, a menos que sea cerrado antes llamando explícitamente a **sybase_close()**.

Vea también **sybase_pconnect()**, **sybase_close()**.

sybase_data_seek (PHP 3, PHP 4)

mueve el puntero interno de la fila

```
int sybase_data_seek (int result_identifier, int row_number)
```

Devuelve: true si lo hace, false en caso de fallo

sybase_data_seek() mueve el puntero interno de la fila del resultado asociado con el identificador de resultado especificado hacia el número de fila introducido. La siguiente llamada a **sybase_fetch_row()** devolverá esa fila.

Vea también: **sybase_data_seek()**.

sybase_fetch_array (PHP 3, PHP 4)

carga una fila como un array

```
int sybase_fetch_array (int result)
```

Returns: An array that corresponds to the fetched row, or false if there are no more rows.

sybase_fetch_array() es la versión extendida de **sybase_fetch_row()**. Además de almacenar los datos en los índices numéricos del array de resultados, también almacena los datos en índices asociativos, usando los nombres de campo como claves.

Una cosa importante a remarcar es que el uso de **sybase_fetch_array()** NO es significativamente más lento que el uso de **sybase_fetch_row()**, mientras que proporciona un valor añadido significativo.

Para más detalles, vea también **sybase_fetch_row()**

sybase_fetch_field (PHP 3, PHP 4)

obtiene la información del campo

```
object sybase_fetch_field (int result, int field_offset)
```

Devuelve un objeto contenido la información del campo

sybase_fetch_field() puede usarse para obtener información sobre los campos de una consulta determinada. Si no se especifica el offset del campo, el siguiente campo que aún no halla sido tomado por **sybase_fetch_field()** es el que se obtiene.

Las propiedades del objeto son:

- name - column name. si la columna es el resultado de una función, esta propiedad se establece a computed#N, donde #N es un número de serie.
- column_source - la tabla de la cual se ha cogido la columna
- max_length - máxima longitud de la columna
- numeric - 1 si la columna es numérica

Vea también **sybase_field_seek()**

sybase_fetch_object (PHP 3, PHP 4)

carga una fila como un objeto

```
int sybase_fetch_object (int result)
```

Devuelve: Un objeto con las propiedades que corresponden a la fila tomada, o false si no hay más filas.

sybase_fetch_object() es similar a **sybase_fetch_array()**, con una diferencia - se devuelve un objeto, en vez de un array. Indirectamente, esto significa que sólo se puede acceder a los datos por los nombres de campo, y no por sus offsets (los números son nombres de propiedades ilegales).

En el tema de velocidad, la función es idéntica a **sybase_fetch_array()**, y al menos tan rápida como **sybase_fetch_row()** (la diferencia es insignificante).

Vea también: **sybase_fetch-array()** y **sybase_fetch-row()**.

sybase_fetch_row (PHP 3, PHP 4)

obtiene una fila como un array enumerado

```
array sybase_fetch_row (int result)
```

Devuelve: Un array que corresponde a la fila obtenida, o false si no hay más filas.

sybase_fetch_row() obtiene una fila de datos del resultado asociado con el identificador de resultado especificado. La fila se devuelve como un array. Cada columna del resultado es almacenada en un offset del array, comenzando en el offset 0.

Las siguientes llamadas a **sybase_fetch_rows()** devolverán la siguiente fila del conjunto de resultados, o false si no hay más filas.

Vea también: **sybase_fetch_array()**, **sybase_fetch_object()**, **sybase_data_seek()**, **sybase_fetch_lengths()**, y **sybase_result()**.

sybase_field_seek (PHP 3, PHP 4)

establece el offset de un campo

```
int sybase_field_seek (int result, int field_offset)
```

Localiza el campo especificado por el offset. Si la siguiente llamada **sybase_fetch_field()** no incluye un offset se devuelve este campo.

Vea también: **sybase_fetch_field()**.

sybase_free_result (PHP 3, PHP 4)

libera el resultado de la memoria

```
int sybase_free_result (int result)
```

sybase_free_result() sólo se necesita usar en el caso de que este preocupado por el uso de demasiada memoria mientras se ejecuta su script. Todos los resultados en memoria son liberados cuando el script finaliza, puede llamar a **sybase_free_result()** con el identificador de resultado como argumento y la memoria asociada a ese resultado será liberada.

sybase_num_fields (PHP 3, PHP 4)

obtiene el número de campos de un resultado

```
int sybase_num_fields (int result)
```

sybase_num_fields() devuelve el número de campos en un conjunto de resultados.

Vea también: **sybase_db_query()**, **sybase_query()**, **sybase_fetch_field()**, **sybase_num_rows()**.

sybase_num_rows (PHP 3, PHP 4)

obtiene el número de filas de un resultado

```
int sybase_num_rows (string result)
```

sybase_num_rows() devuelve el número de filas de un conjunto de resultados.

Vea también: **sybase_db_query()**, **sybase_query()** and, **sybase_fetch_row()**.

sybase_pconnect (PHP 3, PHP 4)

abre una conexión con Sybase persistente

```
int sybase_pconnect (string servername, string username, string password)
```

Devuelve: Un identificador de enlace persistente de Sybase positivo en caso de que pueda abrirla, en caso de error devuelve false

sybase_pconnect() actua de una forma muy parecida a **sybase_connect()** con dos grandes diferencias.

Primera, cuando se conecta, esta función primero tratará de encontrar un enlace (persistente) que ya este abierto con el mismo host, nombre de usuario y contraseña. Si encuentra uno, devolverá un identificador para él en vez de abrir una nueva conexión.

Segunda, la conexión al servidor SQL no se cerrará cuando finalice la ejecución del script. En vez de esto, el enlace permanecerá abierto para un futuro uso (**sybase_close()** no podrá cerrar enlaces establecidos consybase_pconnect()).

Este tipo de enlaces son llamados 'persistentes'.

sybase_query (PHP 3, PHP 4)

envía una consulta a Sybase

```
int sybase_query (string query, int link_identifier)
```

Devuelve: Un identificador de resultado Sybase positivo si va bien, o false ante un error.

`sybase_query()` envía una consulta a la actual base de datos activa en el servidor que está asociada con el identificador de enlace especificado. Si no se especifica un identificador de enlace, se asume el último enlace abierto. Si no hay un enlace abierto, la función intentará establecer un enlace como si `sybase_connect()` fuese llamada, y lo usará.

Vea también: `sybase_db_query()`, `sybase_select_db()`, y `sybase_connect()`.

sybase_result (PHP 3, PHP 4)

obtiene datos de un resultado

```
int sybase_result (int result, int i, mixed field)
```

Devuelve: El contenido de la celda en la fila y el offset especificado de un conjunto de resultados de Sybase.

`sybase_result()` devuelve el contenido de una celda de un conjunto de resultados de Sybase. El parámetro *field* puede ser el offset del campo, o el nombre del campo, o el nombre de la tabla, un punto y el nombre del campo (*nombre_tabla.nombre_campo*). Si el nombre de la columna tiene un alias ('select foo as bar from...'), use el alias en vez del nombre de la columna.

Cuando trabaje con conjuntos de resultados grandes, debe considerar el uso de alguna de las funciones que cargan una fila entera (especificadas abajo). Ya que estas funciones devuelven el contenido de multiples celdas en una única llamada, son MUCHO más rápidas que `sybase_result()`. Además, note que especificar un offset numérico en el parámetro *field* es mucho más rápido que especificar un nombre de campo o *nombre_tabla.nombre_campo*.

Alternativas recomendadas para alto rendimiento: `sybase_fetch_row()`, `sybase_fetch_array()`, y `sybase_fetch_object()`.

sybase_select_db (PHP 3, PHP 4)

seleccciona una base de datos Sybase

```
int sybase_select_db (string database_name, int link_identifier)
```

Returns: true on success, false on error

`sybase_select_db()` establece como activa la base de datos en el servidor asociada con el identificador de enlace especificado. Si no se especifica un identificador de enlace, se asume el último enlace abierto. Si no hay un enlace abierto, la función intentará establecer un enlace como si `sybase_connect()` fuese llamada, y lo usará.

Cada llamada subsiguiente a `sybase_query()` será hecha en la base de datos activa.

Vea también: `sybase_connect()`, `sybase_pconnect()`, y `sybase_query()`

LXXV. Funciones URL

base64_decode (PHP 3, PHP 4)

decodifica datos cifrados con MIME base64

```
string base64_decode (string datos_cifrados)
```

base64_decode() decodifica *datos_cifrados* y devuelve los datos originales. Los datos devueltos pueden ser binarios.

Vea también: **base64_encode()**, RFC-2045 sección 6.8.

base64_encode (PHP 3, PHP 4)

Codifica datos en MIME base64

```
string base64_encode (string datos)
```

base64_encode() devuelve *datos* cifrados en base64. Esta codificación está pensada para que los datos binarios sobrevivan al transporte a través de capas que no son de 8 bits, como por ejemplo los cuerpos de los mensajes de correo.

Los datos codificados con Base64 ocupan aproximadamente un 33% más de espacio que los datos originales.

Vea también: **base64_decode()**, **chunk_split()**, RFC-2045 sección 6.8.

parse_url (PHP 3, PHP 4)

Analiza una URL y devuelve sus componentes

```
array parse_url (string url)
```

Esta función devuelve una matriz que apunta a alguno de los componentes de la URL que estén presentes. Esto incluye el "esquema", "host", "puerto", "usuario", "pass", "path", "consulta", y "fragmento".

urldecode (PHP 3, PHP 4)

decodifica URL-cifradas en una cadena de texto

```
string urldecode (string cadena)
```

Decodifica cualquier %## cifrado en la cadena dada. Se devuelve la cadena decodificada.

Ejemplo 1. Ejemplo urldecode()

```
$a = split ('&', $querystring);
$i = 0;
while ($i < count ($a)) {
    $b = split ('=', $a [$i]);
    echo 'El valor para el parámetro ', htmlspecialchars (urldecode ($b [0])),
        ' es ', htmlspecialchars (urldecode ($b [1])), "<BR>";
    $i++;
}
```

Vea también **urlencode()**

urlencode (PHP 3, PHP 4)

Codifica una URL en una cadena de texto

```
string urlencode (string cadena)
```

Devuelve una cadena en la que todos los caracteres no alfanuméricos excepto -_. han sido reemplazados con un signo de porcentaje (%) seguido por dos dígitos hexadecimales y los espacios han sido codificados como signos positivos (+). Está codificado de la misma manera que los datos que se envian desde un formulario WWW, es decir de la misma forma que el tipo application/x-www-form-urlencoded. Esto difiere del cifrado RFC1738 (vea **rawurlencode()**) en el que por razones históricas, los espacios son codificados como signos positivos (+). Esta función es conveniente para codificar una cadena de texto que va a ser usada como parte de una consulta de una URL, y es una forma adecuada de pasar variables a la página siguiente:

Ejemplo 1. Ejemplo urlencode()

```
echo '<A HREF="mycgi?foo=' . urlencode ($userinput) . '">' ;
```

Vea también **urldecode()**

LXXVI. Funciones sobre variables

doubleval (PHP 3, PHP 4)

Obtiene el valor double (decimal) de una variable.

```
double doubleval (mixed var)
```

Devuelve el valor double (decimal en punto flotante) de *var*.

var puede ser cualquier tipo escalar. No se puede usar **doubleval()** sobre arrays u objetos.

Ver también **intval()**, **strval()**, **settype()** y [Type juggling](#).

empty (unknown)

Determina si una variable está definida

```
int empty (mixed var)
```

Devuelve false si *var* está definida y tiene un valor no-vacío o distinto de cero; en otro caso devuelve true.

Ver también **isset()** y **unset()**.

gettype (PHP 3, PHP 4)

Obtiene el tipo de una variable.

```
string gettype (mixed var)
```

Devuelve el tipo de la variable PHP *var*.

Los valores posibles de la cadena devuelta son:

- "integer"
- "double"
- "string"
- "array"
- "object"
- "unknown type"

Ver también **settype()**.

intval (PHP 3, PHP 4)

Obtiene el valor entero de una variable.

```
int intval (mixed var [, int base])
```

Devuelve el valor entero de `var`, usando la base de conversión especificada (por defecto es base 10).

`var` puede ser cualquier tipo escalar. No se puede usar `intval()` sobre arrays u objetos.

Ver también `doubleval()`, `strval()`, `settype()` y [Type juggling](#).

is_array (PHP 3, PHP 4)

Averigua si una variable es un array.

```
int is_array (mixed var)
```

Devuelve true si `var` es un array, y false en otro caso.

Ver también `is_double()`, `is_float()`, `is_int()`, `is_integer()`, `is_real()`, `is_string()`, `is_long()`, y `is_object()`.

is_double (PHP 3, PHP 4)

Averigua si una variable es un valor double (número decimal).

```
int is_double (mixed var)
```

Devuelve true si `var` es un double (número decimal), y false en otro caso.

Ver también `is_array()`, `is_float()`, `is_int()`, `is_integer()`, `is_real()`, `is_string()`, `is_long()`, y `is_object()`.

is_float (PHP 3, PHP 4)

Averigua si una variable es un flotante.

```
int is_float (mixed var)
```

Esta función es un alias de `is_double()`.

Ver también `is_double()`, `is_real()`, `is_int()`, `is_integer()`, `is_string()`, `is_object()`, `is_array()`, y `is_long()`.

is_int (PHP 3, PHP 4)

Averigua si una variable es un valor entero.

```
int is_int (mixed var)
```

Esta función es un alias de `is_long()`.

Ver también `is_double()`, `is_float()`, `is_integer()`, `is_string()`, `is_real()`, `is_object()`, `is_array()`, y `is_long()`.

is_integer (PHP 3, PHP 4)

Averigua si una variable es un valor entero.

```
int is_integer (mixed var)
```

Esta función es un alias de **is_long()**.

Ver también **is_double()**, **is_float()**, **is_int()**, **is_string()**, **is_real()**, **is_object()**, **is_array()**, y **is_long()**.

is_long (PHP 3, PHP 4)

Averigua si una variable es un valor entero.

```
int is_long (mixed var)
```

Devuelve true si *var* es un entero (long), y false en otro caso.

Ver también **is_double()**, **is_float()**, **is_int()**, **is_real()**, **is_string()**, **is_object()**, **is_array()**, y **is_integer()**.

is_object (PHP 3, PHP 4)

Averigua si una variable es un objeto.

```
int is_object (mixed var)
```

Devuelve true si *var* es un objeto, y false en otro caso.

Ver también **is_long()**, **is_int()**, **is_integer()**, **is_float()**, **is_double()**, **is_real()**, **is_string()**, y **is_array()**.

is_real (PHP 3, PHP 4)

Averigua si una variable es un número real.

```
int is_real (mixed var)
```

Esta función es un alias de **is_double()**.

Ver también **is_long()**, **is_int()**, **is_integer()**, **is_float()**, **is_double()**, **is_object()**, **is_string()**, y **is_array()**.

is_string (PHP 3, PHP 4)

Averigua si una variable es una cadena de caracteres (string).

```
int is_string (mixed var)
```

Devuelve true si *var* es una cadena, y false en otro caso.

Ver también **is_long()**, **is_int()**, **is_integer()**, **is_float()**, **is_double()**, **is_real()**, **is_object()**, y **is_array()**.

isset (unknown)

Determina si una variable está definida

```
int isset (mixed var)
```

Devuelve true si *var* existe; y false en otro caso.

Si una variable ha sido destruida con **unset()**, ya no estará definida (no será **isset()**).

```
$a = "test";
echo isset($a); // true
unset($a);
echo isset($a); // false
```

Ver también **empty()** y **unset()**.

settype (PHP 3, PHP 4)

Establece el tipo de una variable.

```
int settype (string var, string type)
```

Establece el tipo de la variable *var* como *type*.

Los valores posibles para *type* son:

- "integer"
- "double"
- "string"
- "array"
- "object"

Devuelve true si se lleva a cabo con éxito; en otro caso devuelve false.

Ver también **gettype()**.

strval (PHP 3, PHP 4)

Obtiene una cadena de caracteres a partir de una variable.

```
string strval (mixed var)
```

Devuelve una cadena con el valor de *var*.

var puede ser cualquier tipo escalar. No se puede usar **strval()** sobre arrays u objetos.

Ver también **doubleval()**, **intval()**, **settype()** y [Type juggling](#).

unset (unknown)

Destruye una variable dada

```
int unset (mixed var)
```

unset() destruye la variable especificada y devuelve true.

Ejemplo 1. Ejemplo de unset()

```
unset( $foo );
unset( $bar['quux'] );
```

Ver también **isset()** y **empty()**.

LXXVII. Funciones WDDX

Estas funciones permiten el uso de WDDX (<http://www.wddx.org/>).

Debe saber que todas las funciones que serializan variables usan el primer elemento de un array para determinar si este ha de serializarse en forma de array o como estructura. Si el primer elemento esta indexado por una cadena, se serializa como estructura, y en caso contrario, como array.

Ejemplo 1. Serializacion de un valor simple

```
<?php  
print wddx_serialize_value("Ejemplo de PHP a paquete WDDX", "Paquete PHP");  
?>
```

Este ejemplo producira:

```
<wddxPacket version='0.9'><header comment='Paquete PHP'/><data>  
<string>Ejemplo de PHP a paquete WDDX</string></data></wddxPacket>
```

Ejemplo 2. Uso de paquetes incrementales

```
<?php  
$pi = 3.1415926;  
$packet_id = wddx_packet_start("PHP");  
wddx_add_vars($packet_id, "pi");  
  
/* Suponga que $ciudades se ha obtenido de una base de datos */  
$ciudades = array("Austin", "Novato", "Seattle");  
wddx_add_vars($packet_id, "ciudades");  
  
$packet = wddx_packet_end($packet_id);  
print $packet;  
?>
```

Este ejemplo producira:

```
<wddxPacket version='0.9'><header comment='PHP'/><data><struct>  
<var name='pi'><number>3.1415926</number></var><var name='ciudades'>  
<array length='3'><string>Austin</string><string>Novato</string>  
<string>Seattle</string></array></var></struct></data></wddxPacket>
```


wddx_serialize_value (PHP 3>= 3.0.7, PHP 4 >= 4.0b2)

Serializa un valor simple en un paquete WDDX

```
cadena wddx_serialize_value (varios-tipos var [, cadena comentario])
```

wddx_serialize_value() se utiliza para crear un paquete WDDX desde un valor simple dado. Toma el valor contenido en *var*, y una cadena *comentario* opcional que aparecerá en la cabecera del paquete, y devuelve el paquete WDDX.

wddx_serialize_vars (PHP 3>= 3.0.7, PHP 4 >= 4.0b2)

Serializa variables en un paquete WDDX

```
cadena wddx_serialize_vars (varios-tipos nombre_var [, varios-tipos ...])
```

wddx_serialize_vars() se utiliza para crear un paquete WDDX con una estructura que contiene la representación serializada de las variables pasadas como parámetros.

wddx_serialize_vars() toma un número variable de argumentos, cada uno de los cuales puede ser una cadena con el nombre de una variable o un array con nombres de variables, o de otro array, etc.

Ejemplo 1. wddx_serialize_vars example

```
<?php
$a = 1;
$b = 5.5;
$c = array("azul", "naranja", "violeta");
$d = "colores";

$c1vars = array("c", "d");
print wddx_serialize_vars("a", "b", $c1vars);
?>
```

El ejemplo anterior producirá:

```
<wddxPacket version='0.9'><header/><data><struct><var name='a'><number>1</number></var>
<var name='b'><number>5.5</number></var><var name='c'><array length='3'>
<string>azul</string><string>naranja</string><string>violeta</string></array></var>
<var name='d'><string>colores</string></var></struct></data></wddxPacket>
```

wddx_packet_start (PHP 3>= 3.0.7, PHP 4 >= 4.0b2)

Comienza un nuevo paquete WDDX con una estructura dentro

```
entero wddx_packet_start ([cadena comentario])
```

Utilice **wddx_packet_start()** para comenzar un nuevo paquete WDDX que permita la adición sucesiva de variables. Recibe el parámetro opcional *comentario* y devuelve un identificador de paquete para su uso en posteriores llamadas. Automaticamente define una estructura dentro del paquete para contener las variables.

wddx_packet_end (PHP 3>= 3.0.7, PHP 4 >= 4.0b2)

Finaliza un paquete WDDX con el identificador dado

```
cadena wddx_packet_end (entero packet_id)
```

wddx_packet_end() finaliza el paquete WDDX especificado por el *packet_id* y devuelve la cadena con el paquete.

wddx_add_vars (PHP 3>= 3.0.7, PHP 4 >= 4.0b2)

Finaliza un paquete WDDX con el identificador dado

```
wddx_add_vars (entero packet_id, varios-tipos name_var [, varios-tipos ...])
```

wddx_add_vars() se utiliza para serializar las variables dadas e incorporar el resultado al paquete especificado por *packet_id*. Las variables a serializar se especifican exactamente igual que en **wddx_serialize_vars()**.

wddx_deserialize (PHP 3>= 3.0.7, PHP 4 >= 4.0b2)

Des-serializa un paquete WDDX

```
varios-tipos wddx_deserialize (cadena packet)
```

wddx_deserialize() toma una cadena *packet* y la desserializa. Devuelve el resultado que puede ser de tipo cadena, numérico o array. Las estructuras son desserializadas en forma de arrays asociativos.

LXXVIII. Funciones de intérprete XML

Introducción

Acerca de XML

XML (eXtensible Markup Language) es un formato de información para el intercambio de documentos estructurado en la "Web". Es un estándar definido por el consorcio de la "World Wide Web" (W3C). Se puede encontrar información sobre XML y tecnologías relacionadas en <http://www.w3.org/XML/>.

Instalación

Esta extensión usa expat, que se puede encontrar en <http://www.jclark.com/xml/>. El Makefile que viene con expat no crea una biblioteca por defecto, se puede usar esta regla de make para eso:

```
libexpat.a: $(OBJS)
ar -rc $@ $(OBJS)
ranlib $@
```

Se puede conseguir un paquete RPM de expat en <http://www.guardian.no/~ssb/phpxml.html>.

Nota que si se usa Apache-1.3.7 o posterior, ya tienes la biblioteca requerida expat. Simplemente, configura PHP usando `-with-xml` (sin ninguna ruta adicional) y usará automáticamente la biblioteca expat incluida en Apache.

En UNIX, ejecuta `configure` con la opción `-with-xml`. La biblioteca expat debería ser instalada en algún lugar donde el compilador pueda encontrarlo. Si se compila PHP como un módulo para Apache 1.3.9 o posterior, PHP automáticamente usará la biblioteca integrada expat de Apache. Puede necesitar establecer `CPPFLAGS` y `LDLFLAGS` en su entorno antes de ejecutar "configure" si se ha instalado expat en algún lugar exótico.

Compila PHP. ¡*Ta-tam!* Ya debería estar.

Sobre Esta Extensión

Esta extensión de PHP implementa soporte para expat de James Clark en PHP. Este conjunto de herramientas permite interpretar, pero no validar, documentos XML. Soporta tres [codificaciones de caracteres](#) fuente, también proporcionados por PHP: US-ASCII, ISO-8859-1 y UTF-8. UTF-16 no está soportado.

Esta extensión permite [crear intérpretes de XML](#) y definir entonces *gestores* para diferentes eventos XML. Cada intérprete XML tiene también unos cuantos [parámetros](#) que se pueden ajustar.

Los gestores de eventos XML definidos son:

Tabla 1. Gestores de XML soportados

Función PHP para establecer gestor	Descripción del evento
<code>xml_set_element_handler()</code>	Los eventos de elemento ("element") se producen cuando el intérprete XML encuentra etiquetas de comienzo o fin. Hay gestores separados para etiquetas de comienzo y etiquetas de fin.
<code>xml_set_character_data_handler()</code>	La información de caracteres es, por definición, todo el contenido no "marcado" de los documentos XML, incluidos los espacios en blanco entre etiquetas. Nota que el intérprete XML no añade o elimina ningún espacio en blanco, depende de la aplicación (de ti) decidir si el espacio en blanco es significativo.

Función PHP para establecer gestor	Descripción del evento
<code>xml_set_processing_instruction_handler()</code>	Los programadores de PHP deberían estar ya familiarizados con las instrucciones de procesado (PI). <code><?php ?></code> es una instrucción de procesado, donde <i>php</i> se denomina el "objetivo de procesado". El manejo de éstos es específico a cada aplicación, salvo que todos los objetivos PI que comienzan con "XML" están reservados.
<code>xml_set_default_handler()</code>	Todo lo que no va a otro gestor, va al gestor por defecto. Se tendrán en el gestor por defecto cosas como las declaraciones de tipos de documento y XML.
<code>xml_set_unparsed_entity_decl_handler()</code>	Este gestor se llamará para la declaración de una entidad no analizada (NDATA).
<code>xml_set_notation_decl_handler()</code>	Este gestor se llama para la declaración de una anotación.
<code>xml_set_external_entity_ref_handler()</code>	Este gestor se llama cuando el intérprete XML encuentra una referencia a una entidad general interpretada externa. Puede ser una referencia a un archivo o URL, por ejemplo. Ver el ejemplo de entidad externa para demostración.

Case Folding

Las funciones manejadoras de elementos pueden tomar sus nombres de elementos "*case-folded*". Case-folding se define en el estándar XML como "un proceso aplicado a una secuencia de caracteres, en el cual aquellos identificados como sin-mayúsculas son reemplazados por sus equivalentes en mayúsculas". En otras palabras, cuando se trata de XML, case-folding simplemente significa poner en mayúsculas.

Por defecto, todos los nombres de elementos que se pasan a las funciones gestoras están "pasados a mayúsculas". Esta conducta puede ser observada y controlada por el analizador XML con las funciones `xml_parser_get_option()` y `xml_parser_set_option()`, respectivamente.

Códigos de Error

Las siguientes constantes se definen para códigos de error XML (como los devuelve `xml_parse()`):

```
XML_ERROR_NONE
XML_ERROR_NO_MEMORY
XML_ERROR_SYNTAX
XML_ERROR_NO_ELEMENTS
XML_ERROR_INVALID_TOKEN
XML_ERROR_UNCLOSED_TOKEN
XML_ERROR_PARTIAL_CHAR
XML_ERROR_TAG_MISMATCH
XML_ERROR_DUPLICATE_ATTRIBUTE
XML_ERROR_JUNK_AFTER_DOC_ELEMENT
XML_ERROR_PARAM_ENTITY_REF
XML_ERROR_UNDEFINED_ENTITY
XML_ERROR_RECURSIVE_ENTITY_REF
XML_ERROR_ASYNC_ENTITY
XML_ERROR_BAD_CHAR_REF
```

```
XML_ERROR_BINARY_ENTITY_REF
XML_ERROR_ATTRIBUTE_EXTERNAL_ENTITY_REF
XML_ERROR_MISPLACED_XML_PI
XML_ERROR_UNKNOWN_ENCODING
XML_ERROR_INCORRECT_ENCODING
XML_ERROR_UNCLOSED_CDATA_SECTION
XML_ERROR_EXTERNAL_ENTITY_HANDLING
```

Codificación de caracteres

La extensión XML de PHP soporta el conjunto de caracteres Unicode (<http://www.unicode.org/>) a través de diferentes *codificaciones de caracteres*. Hay dos tipos de codificaciones de caracteres, *codificación de fuente* y *codificación de destino*. La representación interna de PHP del documento está siempre codificada con UTF-8.

La codificación de fuente se hace cuando un documento XML es [interpretado](#). Al [crear un intérprete XML](#), se puede especificar una codificación de fuente (esta codificación no se puede cambiar más tarde durante el tiempo de vida del intérprete XML). Las codificaciones de fuente soportadas son ISO-8859-1, US-ASCII y UTF-8. Las dos primeras son codificaciones de byte-único, lo que significa que cada carácter se representa por un solo byte. UTF-8 puede codificar caracteres compuestos por un número variable de bits (hasta 21) en de uno a cuatro bytes. La codificación fuente por defecto usada por PHP es ISO-8859-1.

La codificación de destino se hace cuando PHP pasa datos a las funciones gestoras XML. Cuando se crea un intérprete XML, la codificación de destino se crea igual a la codificación de fuente, pero se puede cambiar en cualquier momento. La codificación de destino afectará a la información de los caracteres así como a los nombres de las etiquetas y a los objetivos de instrucciones de procesado.

Si el intérprete XML encuentra caracteres fuera del rango que su codificación de fuente es capaz de representar, devolverá un error.

Si PHP encuentra caracteres en el documento XML interpretado que no pueden ser representados en la codificación de destino elegida, los caracteres problema serán "degradados". Actualmente, esto significa que tales caracteres se reemplazan por un signo de interrogación.

Algunos Ejemplos

Aquí hay algunos ejemplos de archivos de comandos PHP que interpretan documentos XML.

Ejemplos de Estructuras de Elementos XML

Este primer ejemplo muestra la estructura del elemento inicio en un documento con indentación.

Ejemplo 1. Muestra la Estructura del Elemento XML

```
$file = "data.xml";
$depth = array();

function startElement($parser, $name, $attrs) {
    global $depth;
    for ($i = 0; $i < $depth[$parser]; $i++) {
        print "    ";
    }
    print "$name\n";
    $depth[$parser]++;
}
```

```

function endElement($parser, $name) {
    global $depth;
    $depth[$parser]--;
}

$xml_parser = xml_parser_create();
xml_set_element_handler($xml_parser, "startElement", "endElement");
if (!($fp = fopen($file, "r")))
{
    die("could not open XML input");
}

while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("XML error: %s at line %d",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}
xml_parser_free($xml_parser);

```

Ejemplo de Mapeo de Etiquetas XML

Ejemplo 2. Traduciendo XML a HTML

Este ejemplo transforma etiquetas de un documento XML directamente a etiquetas HTML. Los elementos no encontrados en el "array de traducción ("map array") son ignorados. Por supuesto, este ejemplo solamente funcionará con un tipo de documentos XML específico.

```

$file = "data.xml";
$map_array = array(
    "BOLD"      => "B",
    "EMPHASIS"  => "I",
    "LITERAL"   => "TT"
);

function startElement($parser, $name, $ attrs) {
    global $map_array;
    if ($htmltag = $map_array[$name]) {
        print "<$htmltag>";
    }
}

function endElement($parser, $name) {
    global $map_array;
    if ($htmltag = $map_array[$name]) {
        print "</>$htmltag";
    }
}

function characterData($parser, $data) {
    print $data;
}

$xml_parser = xml_parser_create();
// usa case-folding para que estemos seguros de encontrar la etiqueta
// en $map_array
xml_parser_set_option($xml_parser, XML_OPTION_CASE_FOLDING, true);

```

```

xml_set_element_handler($xml_parser, "startElement", "endElement");
xml_set_character_data_handler($xml_parser, "characterData");
if (!($fp = fopen($file, "r")))
{
    die("could not open XML input");
}

while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("XML error: %s at line %d",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}
xml_parser_free($xml_parser);

```

Ejemplo de Entidad Externa XML

Este ejemplo resalta el código XML. Ilustra cómo usar un gestor de referencia de entidades extensas para incluir y analizar otros documentos, así como cuántos PIs pueden ser procesados, y un modo de determinar "confianza" para PIs que contienen código.

Los documentos XML que se pueden usar en este ejemplo se encuentran bajo el ejemplo (`xmltest.xml` y `xmltest2.xml`.)

Ejemplo 3. Ejemplo de Entidades Externas

```

$file = "xmltest.xml";

function trustedFile($file) {
    // solamente confía en archivos locales que nos pertenezcan
    if (!eregi("^(a-z)+://", $file)
        && fileowner($file) == getmyuid()) {
        return true;
    }
    return false;
}

function startElement($parser, $name, $attribs) {
    print "<font color=\"#0000cc\">$name</font>";
    if (sizeof($attribs)) {
        while (list($k, $v) = each($attribs)) {
            print " <font color=\"#009900\">$k</font>=<font
                  color=\"#990000\">$v</font>\"";
        }
    }
    print "&gt;";
}

function endElement($parser, $name) {
    print "</font color=\"#0000cc\">$name</font>&gt;";
}

function characterData($parser, $data) {
    print "<b>$data</b>";
}

function PIHandler($parser, $target, $data) {

```

```

switch (strtolower($target)) {
    case "php":
        global $parser_file;
        // Si el documento analizado es "de confianza", diremos
        // que es seguro ejecutar código PHP en su interior.
        // Si no, en vez de ello mostrará el código.
        if (trustedFile($parser_file[$parser])) {
            eval($data);
        } else {
            printf("Untrusted PHP code: <i>%s</i>",
                htmlspecialchars($data));
        }
        break;
}
}

function defaultHandler($parser, $data) {
    if (substr($data, 0, 1) == "&" && substr($data, -1, 1) == ";") {
        printf('<font color="#aa00aa">%s</font>',
            htmlspecialchars($data));
    } else {
        printf('<font size="-1">%s</font>',
            htmlspecialchars($data));
    }
}

function externalEntityRefHandler($parser, $openEntityNames, $base, $systemId,
                                  $publicId) {
    if ($systemId) {
        if (!list($parser, $fp) = new_xml_parser($systemId)) {
            printf("Could not open entity %s at %s\n",
                $openEntityNames,
                $systemId);
            return false;
        }
        while ($data = fread($fp, 4096)) {
            if (!xml_parse($parser, $data, feof($fp))) {
                printf("XML error: %s at line %d while parsing entity %s\n",
                    xml_error_string(xml_get_error_code($parser)),
                    xml_get_current_line_number($parser), $openEntityNames);
                xml_parser_free($parser);
                return false;
            }
        }
        xml_parser_free($parser);
        return true;
    }
    return false;
}

function new_xml_parser($file) {
    global $parser_file;

    $xml_parser = xml_parser_create();
    xml_parser_set_option($xml_parser, XML_OPTION_CASE_FOLDING, 1);
    xml_set_element_handler($xml_parser, "startElement", "endElement");
    xml_set_character_data_handler($xml_parser, "characterData");
    xml_set_processing_instruction_handler($xml_parser, "PIHandler");
    xml_set_default_handler($xml_parser, "defaultHandler");
    xml_set_external_entity_ref_handler($xml_parser, "externalEntityRefHandler");

    if (!$fp = @fopen($file, "r")) {

```

```

        return false;
    }
    if (!is_array($parser_file)) {
        settype($parser_file, "array");
    }
    $parser_file[$xml_parser] = $file;
    return array($xml_parser, $fp);
}

if (!(list($xml_parser, $fp) = new_xml_parser($file))) {
    die("could not open XML input");
}

print "<pre>";
while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("XML error: %s at line %d\n",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}
print "</pre>";
print "parse complete\n";
xml_parser_free($xml_parser);

?>

```

Ejemplo 4. xmltest.xml

```

<?xml version='1.0'?>
<!DOCTYPE chapter SYSTEM "/just/a/test.dtd" [
<!ENTITY plainEntity "FOO entity">
<!ENTITY systemEntity SYSTEM "xmltest2.xml">
]>
<chapter>
<TITLE>Title &plainEntity;</TITLE>
<para>
<informaltable>
<tgroup cols="3">
<tbody>
<row><entry>a1</entry><entry morerows="1">b1</entry><entry>c1</entry></row>
<row><entry>a2</entry><entry>c2</entry></row>
<row><entry>a3</entry><entry>b3</entry><entry>c3</entry></row>
</tbody>
</tgroup>
</informaltable>
</para>
&systemEntity;
<sect1 id="about">
<title>About this Document</title>
<para>
<!-- this is a comment -->
<?php print 'Hi! This is PHP version '.phpversion(); ?>
</para>
</sect1>
</chapter>

```

Este archivo se incluye desde `xmltest.xml`:

Ejemplo 5. `xmltest2.xml`

```
<?xml version="1.0"?>
<!DOCTYPE foo [
<!ENTITY testEnt "test entity">
]>
<foo>
  <element attrib="value"/>
  &testEnt;
  <?php print "This is some more PHP code being executed."; ?>
</foo>
```

xml_parser_create (PHP 3>= 3.0.6, PHP 4)

crea un analizador de XML

```
int xml_parser_create ([string encoding])
```

encoding (opcional)

Qué codificación de caracteres debería usar el analizador. Las siguientes codificación de caracteres están soportadas:

ISO-8859-1 (por defecto)

US-ASCII

UTF-8

Esta función crea un analizador XML y devuelve un índice para usarlo con otras funciones XML. Devuelve `false` en caso de fallo.

xml_set_object (PHP 4 >= 4.0b4)

Usa un analizador XML dentro de un objeto

```
void xml_set_object (int parser, object &object)
```

Esta función hace a *parser* utilizable dentro de *object*. Todas las funciones de callback establecidas por `xml_set_element_handler()` etc se asumen como métodos de *object*.

```
<?php
class xml {
var $parser;

function xml() {
    $this->parser = xml_parser_create();
    xml_set_object($this->parser,&$this);
    xml_set_element_handler($this->parser, "tag_open", "tag_close");
    xml_set_character_data_handler($this->parser, "cdata");
}

function parse($data) {
    xml_parse($this->parser,$data);
}

function tag_open($parser,$tag,$attributes) {
    var_dump($parser,$tag,$attributes);
}

function cdata($parser,$cdata) {
    var_dump($parser,$cdata);
}

function tag_close($parser,$tag) {
    var_dump($parser,$tag);
}

} // end of class xml

$xml_parser = new xml();
```

```
$xml_parser->parse( "<A ID=\\"hallo\\">PHP</A>" );
?>
```

Nota: `xml_set_object()` es gestionable a partir de PHP 4.0.

xml_set_element_handler (PHP 3>= 3.0.6, PHP 4)

establece gestores de los elementos principio y fin

```
int xml_set_element_handler ( int parser, string startElementHandler, string
endElementHandler )
```

Establece las funciones de gestión de elementos para el analizador XML *parser*. *startElementHandler* y *endElementHandler* son strings que contienen los nombres de las funciones que deben existir cuando `xml_parse()` es llamado por *parser*.

La función denominada *startElementHandler* debe aceptar tres parámetros:

```
startElementHandler ( int parser, string name, string attribs )
```

parser

El primer parámetro, *parser*, es una referencia al analizador XML que llama al gestor.

name

El segundo parámetro, *name*, contiene el nombre del elemento para el que se llama a este gestor. Si la propiedad de [case-folding](#) tiene efecto para este analizador, el nombre del elemento estará en mayúsculas.

attribs

El tercer parámetro, *attribs*, contiene un array asociativo con los atributos de los elementos (si hay). Las claves de este array son los nombres de los atributos, los valores son los valores de los atributos. Los nombres de los atributos están en mayúsculas ([case-folded](#)) con el mismo criterio que los nombres de los elementos. Los valores de los atributos *no* sufren las consecuencias de case-folding.

El orden original de los atributos se puede recuperar recorriendo *attribs* del modo usual, usando `each()`. La primera clave del array es el el primer atributo, y así sucesivamente.

La función llamada *endElementHandler* debe aceptar dos parámetros:

```
endElementHandler ( int parser, string name )
```

parser

El primer parámetro, *parser*, es una referencia al analizador XML que llama al gestor.

name

El segundo parámetro, *name*, contiene el nombre del elemento para el que se llama a este gestor. Si la propiedad de [case-folding](#) tiene efecto para este analizador, el nombre del elemento estará en mayúsculas.

Si una función gestora se establece como la cadena vacía, o `false`, el gestor en cuestión se deshabilita.

Se devuelve true si se establecieron los gestores, false si *parser* no es un analizador.

En la actualidad no hay soporte para gestores objeto/método.

xml_set_character_data_handler (PHP 3>= 3.0.6, PHP 4)

Establece gestores de datos de caracteres

```
int xml_set_character_data_handler (int parser, string handler)
```

Establece la función gestora de datos de caracteres para el analizador XML *parser*. *handler* es un string que contiene el nombre de la función que debe existir cuando **xml_parse()** es llamado por *parser*.

La función nombrada en *handler* debe aceptar dos parámetros:

```
handler (int parser, string data)
```

parser

El primer parámetro, *parser*, es una referencia al analizador XML que llama al gestor.

data

El segundo parámetro, *data*, contiene los datos caracteres como string.

Si una función gestora se establece como la cadena vacía, o `false`, el gestor en cuestión se deshabilita.

Se devuelve true si se estableció el gestor, false si *parser* no es un analizador.

En la actualidad no hay soporte para gestores objeto/método.

xml_set_processing_instruction_handler (PHP 3>= 3.0.6, PHP 4)

Establece el gestor de instrucciones de procesado (PI)

```
int xml_set_processing_instruction_handler (int parser, string handler)
```

Establece la función de gestión de instrucciones de procesado (PI) para el analizador XML *parser*. *handler* es un string que contiene el nombre de una función que debe existir cuando **xml_parse()** es llamada por *parser*.

Una instrucción de procedado tiene el siguiente formato:

```
<?
  target
  data?>
```

Puedes poner código PHP en esa etiqueta, pero ten en cuenta una limitación: en una PI XML, la etiqueta de fin de la PI (`?>`) no puede ser citada, por lo que esta secuencia de caracteres no debería aparecer en el código PHP que insertes con las

PIs en documentos XML. Si lo hace, el resto del código PHP, así como la etiqueta de fin de PI "real", serán tratados como datos de caracteres.

La función nombrada en *handler* debe aceptar tres parámetros:

```
handler (int parser, string target, string data)
```

parser

El primer parámetro, *parser*, es una referencia al analizador XML que llama al gestor.

target

El segundo parámetro, *target*, contiene el objetivo PI.

data

El tercer parámetro, *data*, contiene los datos PI.

Si una función gestora se establece como la cadena vacía, o `false`, el gestor en cuestión se deshabilita.

Se devuelve true si se estableció el gestor, false si *parser* no es un analizador.

En la actualidad no hay soporte para gestores objeto/método.

xml_set_default_handler (PHP 3>= 3.0.6, PHP 4)

set up default handler

```
int xml_set_default_handler (int parser, string handler)
```

Establece la función gestora por defecto para un analizador XML *parser*. *handler* es un string que contiene el nombre de la función que debe existir cuando `xml_parse()` es llamado por *parser*.

La función nombrada en *handler* debe aceptar dos parámetros:

```
handler (int parser, string data)
```

parser

El primer parámetro, *parser*, es una referencia al analizador XML que llama al gestor.

data

El segundo parámetro, *data*, contiene los caracteres de dato. Esto puede ser la declaración XML, la declaración de tipo de documento, entidades u otros datos para los cuales no existe otro gestor.

Si una función gestora se establece como la cadena vacía, o `false`, el gestor en cuestión se deshabilita.

Se devuelve true si se estableció el gestor, false si *parser* no es un analizador.

En la actualidad no hay soporte para gestores objeto/método.

xml_set_unparsed_entity_decl_handler (PHP 3>= 3.0.6, PHP 4)

Establece un gestor de declaraciones de entidades no analizadas

```
int xml_set_unparsed_entity_decl_handler (int parser, string handler)
```

Establece la función gestora de declaración de entidades no analizadas para el analizador XML *parser*. *handler* es una cadena que contiene el nombre de una función que debe existir cuando **xml_parse()** es llamada por *parser*.

Este gestor será llamado si el analizador XML encuentra una declaración de entidades externas con una declaración NDATA, como la siguiente:

```
<!ENTITY name {publicId | systemId}
  NDATA notationName>
```

Mira la sección 4.2.2 de las especificaciones XML 1.0

(<http://www.w3.org/TR/1998/REC-xml-19980210#sec-external-ent>) para la definición de entidades externas de notación declarada.

La función nombrada en *handler* debe aceptar seis parámetros:

```
handler (int parser, string entityName, string base, string systemId, string publicId,
string notationName)
```

parser

El primer parámetro, *parser*, es una referencia al analizador XML que llama al gestor.

entityName

El nombre de la entidad que va a ser definida.

base

Esta es la base para resolver el identificador de sistema (*systemId*) de la entidad externa. Actualmente este parámetro siempre será una cadena vacía.

systemId

Identificador de Sistema para la entidad externa.

publicId

Identificador público para la entidad externa.

notationName

Nombre de la notación de esta entidad (ver **xml_set_notation_decl_handler()**).

Si una función gestora se establece como la cadena vacía, o *false*, el gestor en cuestión se deshabilita.

Se devuelve true si se estableció el gestor, false si *parser* no es un analizador.

En la actualidad no hay soporte para gestores objeto/método.

xml_set_notation_decl_handler (PHP 3>= 3.0.6, PHP 4)

Establece gestores de declaraciones de notación

```
int xml_set_notation_decl_handler (int parser, string handler)
```

Establece las funciones gestoras de declaraciones de notación para el analizador XML *parser*. *handler* es un string que contiene el nombre de una función que debe existir cuando **xml_parse()** es llamado por *parser*.

Una declaración de notación es parte del DTD del documento y tiene el siguiente formato:

```
<!NOTATION name
{systemId | publicId}
>
```

Ver la sección 4.7 de las especificaciones XML 1.0 (<http://www.w3.org/TR/1998/REC-xml-19980210#Notations>) para la definición de declaraciones de notación.

La función llamada por *handler* debe aceptar cinco parámetros:

```
handler (int parser, string notationName, string base, string systemId, string publicId)
```

parser

El primer parámetro, *parser*, es una referencia al analizador XML que llama al gestor.

notationName

Este es el *nombre* de la notación, como se describió arriba en el formato de notación.

base

Esta es la base para resolver el identificador de sistema (*systemId*) de la declaración. En la actualidad este parámetro es siempre la cadena vacía.

systemId

Identificador de sistema de la declaración de notación externa.

publicId

Identificador público de la declaración de notación externa.

Si una función gestora se establece como la cadena vacía, o `false`, el gestor en cuestión se deshabilita.

Se devuelve `true` si se estableció el gestor, `false` si *parser* no es un analizador.

En la actualidad no hay soporte para gestores objeto/método.

xml_set_external_entity_ref_handler (PHP 3>= 3.0.6, PHP 4)

Establece gestores de referencia de entidades externas

```
int xml_set_external_entity_ref_handler (int parser, string handler)
```

Establece la función gestora de declaraciones de notación para el analizador XML *parser*. *handler* es un string que contiene el nombre de una función que debe existir cuando **xml_parse()** es llamado por *parser*.

La función llamada por *handler* debe aceptar cinco parámetros, y debería devolver un valor entero. Si el valor devuelto desde el gestor (*handler*) es falso (lo cual ocurrirá si no se devuelve un valor), el analizador XML dejará de analizar y **xml_get_error_code()** devolverá `XML_ERROR_EXTERNAL_ENTITY_HANDLING`.

```
int handler (int parser, string openEntityNames, string base, string systemId, string publicId)
```

parser

El primer parámetro, *parser*, es una referencia al analizador XML que llama al gestor.

openEntityNames

El segundo parámetro, *openEntityNames*, es una lista, separada por espacios, de los nombres de las entidades que se abren para el análisis de esta entidad (incluido el nombre de la entidad referenciada).

base

Esta es la base para resolver el identificador de sistema (*systemid*) de la entidad externa. En la actualidad este parámetro es siempre la cadena vacía.

systemId

El cuarto parámetro, *systemId*, es el identificador del sistema tal como se especificó en la declaración de la entidad.

publicId

El quinto parámetro, *publicId*, es el identificador público como se especificó en la declaración de la entidad, o una cadena vacía si no se especificó ninguno; el espacio en blanco en el identificador público se habrá normalizado como se requiere en las especificaciones XML.

Si una función gestora se establece como la cadena vacía, o `false`, el gestor en cuestión se deshabilita.

Se devuelve `true` si se estableció el gestor, `false` si *parser* no es un analizador.

En la actualidad no hay soporte para gestores objeto/método.

xml_parse (PHP 3>= 3.0.6, PHP 4)

comienza a analizar un documento XML

```
int xml_parse (int parser, string data [, int isFinal])
```

parser

Una referencia al analizador XML que se va a utilizar.

data

Conjunto de información que se analizará. Un documento puede ser analizado por trozos llamando varias veces a **xml_parse()** con nueva información, siempre que se establezca el parámetro *isFinal* y sea `true` cuando el último dato sea analizado.

isFinal (optional)

Si existe y es `true`, *data* es el último pedazo de información enviado en este análisis.

Cuando el documento XML es analizado, se hacen llamadas a los gestores para los eventos configurados tantas veces como sea necesario, después de que esta función devuelva `true` o `false`.

Devuelve `true` si el análisis se realiza con éxito, `false` si no tiene éxito, o si *parser* no referencia a un analizador válido. Para análisis fallidos, se puede recuperar información de error con **xml_get_error_code()**, **xml_error_string()**, **xml_get_current_line_number()**, **xml_get_current_column_number()** y **xml_get_current_byte_index()**.

xml_get_error_code (PHP 3>= 3.0.6, PHP 4)

obtiene el código de error del analizador XML

```
int xml_get_error_code (int parser)
```

parser

Una referencia al analizador XML del que obtener el código de error.

Esta función devuelve false si *parser* no referencia un analizador válido, o si no devuelve uno de los códigos de error listados en la [sección de códigos de error](#).

xml_error_string (PHP 3>= 3.0.6, PHP 4)

obtiene la cadena de error del analizador XML

```
string xml_error_string (int code)
```

code

Un código de error de **xml_get_error_code()**.

Devuelve una cadena con una descripción textual del código de error *code*, o false si no se encontró descripción.

xml_get_current_line_number (PHP 3>= 3.0.6, PHP 4)

obtiene el número de línea actual de un analizador XML

```
int xml_get_current_line_number (int parser)
```

parser

Una referencia al analizador XML del que obtener el número de línea.

Esta función devuelve false si *parser* no referencia un analizador válido, o si no devuelve en qué línea se encuentra actualmente el buffer de datos del analizador.

xml_get_current_column_number (PHP 3>= 3.0.6, PHP 4)

Obtiene el número de columna actual para un analizador XML.

```
int xml_get_current_column_number (int parser)
```

parser

Una referencia al analizador XML del que obtener el número de columna.

Esta función devuelve false si *parser* no referencia un analizador válido, o si no devuelve en qué columna de la línea actual (como se obtiene de **xml_get_current_line_number()**) en la que se encuentra el analizador.

xml_get_current_byte_index (PHP 3>= 3.0.6, PHP 4)

obtiene el índice del byte actual para un analizador XML

```
int xml_get_current_byte_index ( int parser )
```

parser

Una referencia al analizador XML del que obtener el índice del byte.

Esta función devuelve false si *parser* no referencia un analizador válido, o si no devuelve en qué índice de byte se encuentra el buffer de datos del analizador (empezando en 0).

xml_parser_free (PHP 3>= 3.0.6, PHP 4)

Libera un analizador XML

```
string xml_parser_free ( int parser )
```

parser

Una referencia al analizador XML que se liberará.

Esta función devuelve false si *parser* no referencia un analizador válido, o si no libera el analizador y devuelve true.

xml_parser_set_option (PHP 3>= 3.0.6, PHP 4)

establece las opciones de un analizador XML

```
int xml_parser_set_option ( int parser, int option, mixed value )
```

parser

Una referencia al analizador XML en el que establecer opciones.

option

Opción que se establecerá. Ver más abajo.

value

El nuevo valor de la opción.

Esta función devuelve false si *parser* no referencia un analizador válido, o si la opción no pudo ser establecida. Si no, la opción se establece y devuelve true.

Las opciones siguientes están disponibles:

Tabla 1. Opciones de analizador XML

Opción constante	Tipo de Datos	Descripción
XML_OPTION_CASE_FOLDING	integer	Controla si case-folding se habilita para este analizador XML. Habilitado por defecto.
XML_OPTION_TARGET_ENCODING	string	Establece qué codificación destino se usa en este analizador XML. Por defecto, esta puesta al mismo que la codificación fuente usada por xml_parser_create() . Las codificaciones de destino soportadas son ISO-8859-1, US-ASCII y UTF-8.

xml_parser_get_option (PHP 3>= 3.0.6, PHP 4)

obtiene las opciones de un analizador XML

```
mixed xml_parser_get_option (int parser, int option)
```

parser

Una referencia al analizador XML del que obtener opciones.

option

Qué opción recuperar. Ver **xml_parser_set_option()** para una lista de opciones.

Esta función devuelve false si *parser* no referencia un analizador válido, o si la opción no pudo ser establecida. Si no, se devuelve la opción.

Mirar **xml_parser_set_option()** para la lista de opciones.

utf8_decode (PHP 3>= 3.0.6, PHP 4)

Convierte una cadena codificada UTF-8 a ISO-8859-1

```
string utf8_decode (string data)
```

Esta función decodifica *data*, asume codificación UTF-8 , a ISO-8859-1.

Mira **utf8_encode()** para una explicación de codificación UTF-8.

utf8_encode (PHP 3>= 3.0.6, PHP 4)

codifica una cadena ISO-8859-1 a UTF-8

```
string utf8_encode (string data)
```

Esta función codifica la cadena *data* a UTF-8, y devuelve la versión codificada. UTF-8 es un mecanismo estándar usado por Unicode para codificar valores de *caracteres amplios* en un chorro de bytes. UTF-8 es transparente a caracteres de ASCII plano, es auto-sincronizado (significa que es posible para un programa averiguar dónde comienzan los caracteres en el chorro de bytes) y se puede usar con funciones de comparación de cadenas normales para ordenar y otros fines. PHP codifica caracteres UTF-8 en hasta cuatro bytes, como esto:

Tabla 1. Codificación UTF-8

bytes	bits	representación
1	7	0bbbbbbb
2	11	110bbbb 10bbbbbb
3	16	1110bbbb 10bbbbbb 10bbbbbb
4	21	11110bbb 10bbbbbb 10bbbbbb 10bbbbbb

Cada *b* representa un bit que puede ser usado para almacenar datos de caracteres.

LXXIX. XSLT functions

Introduction

About XSLT and Sablotron

XSLT (Extensible Stylesheet Language (XSL) Transformations) is a language for transforming XML documents into other XML documents. It is a standard defined by The World Wide Web consortium (W3C). Information about XSLT and related technologies can be found at <http://www.w3.org/TR/xslt>.

Installation

This extension uses Sabloton and expat, which can both be found at <http://www.gingerall.com/>. Binaries are provided as well as source.

On UNIX, run **configure** with the `-with-sablot` and `-enable-sablot-errors-descriptive` options. The Sablotron library should be installed somewhere your compiler can find it.

About This Extension

This PHP extension implements support Sablotron from Ginger Alliance in PHP. This toolkit lets you transform XML documents into other documents, including new XML documents, but also into HTML or other target formats. It basically provides a standardized and portable template mechanism, separating content and design of a website.

xslt_closelog (PHP 4 >= 4.0.3)

Clear the logfile for a given instance of Sablotron

```
bool xslt_closelog (resource xh)
```

xh

A reference to the XSLT parser.

This function returns false if *parser* does not refer to a valid parser, or if the closing of the logfile fails. Otherwise it returns true.

xslt_create (PHP 4 >= 4.0.3)

Create a new XSL processor.

```
resource xslt_create(void);
```

This function returns a handle for a new XSL processor. This handle is needed in all subsequent calls to XSL functions.

xslt_errno (PHP 4 >= 4.0.3)

Return the current error number

```
int xslt_errno ([int xh])
```

Return the current error number of the given XSL processor. If no handle is given, the last error number that occurred anywhere is returned.

xslt_error (PHP 4 >= 4.0.3)

Return the current error string

```
mixed xslt_error ([int xh])
```

Return the current error string of the given XSL processor. If no handle is given, the last error string that occurred anywhere is returned.

xslt_fetch_result (PHP 4 >= 4.0.3)

Fetch a (named) result buffer

```
string xslt_fetch_result ([int xh string result_name])
```

Fetch a result buffer from the XSLT processor identified by the given handle. If no result name is given, the buffer named "/_result" is fetched.

xslt_free (PHP 4 >= 4.0.3)

Free XSLT processor

```
void xslt_free (resource xh)
```

Free the XSLT processor identified by the given handle.

xslt_openlog (PHP 4 >= 4.0.3)

Set a logfile for XSLT processor messages

```
bool xslt_openlog ([resource xh string logfile int loglevel])
```

Set a logfile for the XSLT processor to place all of its error messages.

xslt_output_begintransform (PHP 4 >= 4.0.3)

unknown

```
unknown xslt_output_begintransform (unknown)
```

This function lacks a prototype definition.

xslt_output_endtransform (PHP 4 >= 4.0.3)

unknown

```
unknown xslt_output_endtransform (unknown)
```

This function lacks a prototype definition.

xslt_output_process (unknown)

unknown

```
unknown xslt_process (unknown)
```

This function lacks a prototype definition.

xslt_run (PHP 4 >= 4.0.3)

Apply a XSLT stylesheet to a file.

```
bool xslt_run ([resource xh string xslt_file string xml_data_file string result array  
  xslt_params array xslt_args]])
```

Process the *xml_data_file* by applying the *xslt_file* stylesheet to it. The stylesheet has access to *xslt_params* and the processor is started with *xslt_args*. The result of the XSLT transformation is placed in the named buffer (default is "/_result").

xslt_set_sax_handler (PHP 4 >= 4.0.3)

Set SAX handlers for a XSLT processor

```
bool xslt_set_sax_handler (resource xh array handlers)
```

Set SAX handlers on the ressource handle given by *xh*.

xslt_transform (PHP 4 >= 4.0.3)

unknown

```
unknown xslt_transform (unknown)
```

This function lacks a prototype definition.

LXXX. YAZ

The **yaz()** functions wrap the YAZ API. The home page of the project is <http://www.indexdata.dk/yaz/>. Information about the phpyaz module can be found at <http://www.indexdata.dk/phypyaz/>.

PHP/YAZ is much simpler to use than the C API for YAZ but less flexible. The intent is to make it easy to build basic client functions. It supports persistent stateless connections very similar to those offered by the various SQL APIs that are available for PHP. This means that sessions are stateless but shared amongst users, thus saving the connect and INIT steps in many cases.

Before compiling PHP with the PHP/YAZ module you'll need the YAZ toolkit. Build YAZ and install it. Build PHP with your favourite modules and add option **-with-yaz**. Your task is roughly the following:

```
gunzip -c yaz-1.6.tar.gz|tar xf -
gunzip -c php-4.0.X.tar.gz|tar xf -
cd yaz-1.6
./configure --prefix=/usr
make
make install
cd .. /php-4.0.X
./configure --with-yaz=/usr/bin
make
make install
```

PHP/YAZ keeps track of connections with targets (Z-Associations). A positive integer represents the ID of a particular association.

The script below demonstrates the parallel searching feature of the API. When invoked it either prints a query form (if no arguments are supplied) or if there are arguments (term and one or more hosts) it searches the targets in array host.

Ejemplo 1. YAZ()

```
$num_hosts = count ($host);
if (empty($term) || count($host) == 0) {
    echo '<form method="get">
        <input type="checkbox"
            name="host[]"
            value="bagel.indexdata.dk/gils">
            GILS test
        <input type="checkbox"
            name="host[]"
            value="localhost:9999/Default">
            local test
        <input type="checkbox" checked="1"
            name="host[]"
            value="z3950.bell-labs.com/books">
            BELL Labs Library
    <br>
    RPN Query:
    <input type="text" size="30" name="term">
    <input type="submit" name="action" value="Search">
';
} else {
    echo 'You searched for ' . htmlspecialchars($term) . '<br>';
    for ($i = 0; $i > $num_hosts; $i++) {
        $id[] = yaz_connect($host[$i]);
        yaz_syntax($id[$i],"sutrs");
        yaz_search($id[$i],"rpn",$term);
    }
    yaz_wait();
    for ($i = 0; $i < $num_hosts; $i++) {
        echo '<hr>' . $host[$i] . ":" ;
        $error = yaz_error($id[$i]);
        if (!empty($error)) {
```

```
    echo "Error: $error";
} else {
    $hits = yaz_hits($id[$i]);
    echo "Result Count $hits";
}
echo '<dl>';
for ($p = 1; $p <= 10; $p++) {
    $rec = yaz_record($id[$i],$p,"string");
    if (empty($rec)) continue;
    echo "<dt><b>$p</b></dt><dd>";
    echo ereg_replace("\n", "<br>\n", $rec);
    echo "</dd>";
}
echo '</dl>';
}
```

yaz_addinfo (PHP 4 >= 4.0.1)

Returns additional error information

```
int yaz_addinfo (int id)
```

Returns additional error message for target (last request). An empty string is returned if last operation was a success or if no additional information was provided by the target.

yaz_close (PHP 4 >= 4.0.1)

Closes a YAZ connection

```
int yaz_close (int id)
```

Closes a connection to a target. The application can no longer refer to the target with the given id.

yaz_connect (PHP 4 >= 4.0.1)

Returns a positive association ID on success; zero on failure

```
int yaz_connect (string zurl)
```

Yaz_connect() prepares for a connection to a Z39.50 target. The zurl argument takes the form host[:port][/database]. If port is omitted 210 is used. If database is omitted Default is used. This function is non-blocking and doesn't attempt to establish a socket - it merely prepares a connect to be performed later when **yaz_wait()** is called.

yaz_errno (PHP 4 >= 4.0.1)

Returns error number

```
int yaz_errno (int id)
```

Returns error for target (last request). A positive value is returned if the target returned a diagnostic code; a value of zero is returned if no errors occurred (success); negative value is returned for other errors targets didn't indicate the error in question.

Yaz_errno() should be called after network activity for each target - (after **yaz_wait()** returns) to determine the success or failure of the last operation (e.g. search).

yaz_error (PHP 4 >= 4.0.1)

Returns error description

```
int yaz_error (int id)
```

Returns error message for target (last request). An empty string is returned if last operation was a success.

Yaz_error() returns a english message corresponding to the last error number as returned by **yaz_errno()**.

yaz_hits (PHP 4 >= 4.0.1)

Returns number of hits for last search

```
int yaz_hits (int id)
```

Yaz_hits() returns number of hits for last search.

yaz_range (PHP 4 >= 4.0.1)

Specifies the maximum number of records to retrieve

```
int yaz_range (int id, int start, int number)
```

This function is used in conjunction with **yaz_search()** to specify the maximum number of records to retrieve (number) and the first record position (start). If this function is not invoked (only **yaz_search()**) start is set to 1 and number is set to 10.

Returns true on success; false on error.

yaz_record (PHP 4 >= 4.0.1)

Returns a record

```
int yaz_record (int id, int pos, string type)
```

Returns record at position or empty string if no record exists at given position.

The **yaz_record()** function inspects a record in the current result set at the position specified. If no database record exists at the given position an empty string is returned. The argument, type, specifies the form of the returned record. If type is "string"the record is returned in a string representation suitable for printing (for XML and SUTRS). If type is "array"the record is returned as an array representation (for structured records).

yaz_search (PHP 4 >= 4.0.1)

Prepares for a search

```
int yaz_search (int id, string type, string query)
```

Yaz_search() prepares for a search on the target with given id. The type represents the query type - only "rpn" is supported now in which case the third argument is a prefix notation query as used by YAZ. Like **yaz_connect()** this function is non-blocking and only prepares for a search to be executed later when **yaz_wait()** is called.

yaz_syntax (PHP 4 >= 4.0.1)

Specifies the preferred record syntax for retrieval

```
int yaz_syntax (int id, string syntax)
```

This function is used in conjunction with **yaz_search()** to specify the preferred record syntax for retrieval.

yaz_wait (PHP 4 >= 4.0.1)

Executes queries

```
int yaz_wait (int id, string syntax)
```

This function carries out networked (blocked) activity for outstanding requests which have been prepared by the functions **yaz_connect()**, **yaz_search()**. **yaz_wait()** returns when all targets have either completed all requests or otherwise completed (in case of errors).

LXXXI. NIS funciona

NIS (anteriormente llamado Paginas Amarillas) permite la administracion de red de los archivos de administracion importantes (e.g.El archivo de contraseñas). Para mas informacion dirigirse a las paginas de ayuda de NIS y a la direccion. Introduccion a YP/NIS (<http://www.desy.de/~sieversm/ypdoku/ypdoku.html>) Hay tambien un libro llamado gestionando NFS Y NIS (<http://www.oreilly.com/catalog/nfs/noframes.html>) por Hal Stern.

Para obtener estas funciones de trabajo, usted tiene que configure PHP con - con- yp.

yp_get_default_domain (PHP 3>= 3.0.7, PHP 4)

Trae el valor por omision de dominios de maquina NIS.

```
int yp_get_default_domain (nulo)
```

yp_get_default_domain() Retorna el valor por omision del dominio del nodo o FALSO. Puede ser usado el parametro de dominio para sucesivas llamadas a NIS.

Un dominio de NIS puede ser descrito en un grupo de mapas NIS. Cada host necesita buscar uniones de informacion en un mismo dominio. Acudir a los documentos mencionados en el comienzo para mas informacion.

Ejemplo 1. Ejemplo para el dominio por omision

Ver tambien: [yp_errno](#) (nombre de la funcion) y [yp_err_string](#) (nombre de la funcion)

yp_order (PHP 3>= 3.0.7, PHP 4)

Retorna el numero de orden para el mapa.

```
int yp_order (nombre de la funcion) (cadena dominio, cadena mapa)
```

yp_order(nombre de la funcion)() Retorna el numero de orden para un mapa o FALSO.

Ejemplo 1. Ejemplo para ordenar el NIS

Ver tambien: [yp_get_default_domain](#) [yp_errno](#) [yp_err_string](#)

yp_master (PHP 3>= 3.0.7, PHP 4)

Retorna el nombre del servidor de NIS maestro para el mapa.

```
cadena yp_master (cadena dominio, cadena mapa)
```

yp_master() Retorna el nombre de maquina del servidor de NIS maestro para un mapa.

Ejemplo 1. Ejemplo para el NIS domina

Ver tambien: [yp_get_default_domain\(nombre de la funcion\)](#) [yp_errno\(nombre de la funcion\)](#) y [yp_err_string\(nombre de la funcion\)](#)

yp_match (PHP 3>= 3.0.7, PHP 4)

Retorna la linea compaÑera (pareja).

```
cadena yp_match (cadena dominio, cadena mapa, cadena teclea)
```

yp_match(nombre de la funcion)() Retorna el valor asociado con la llave pasada fuera del mapa especificado o FALSO. esta llave tiene que ser exacta.

Ejemplo 1. Ejemplo para NIS parejo

En este caso esto puede ser: Joe:##joe:11111:100:joe usuario:/hogar/j/joe: User:/usr/local/bin/bash

Ver tambien: [yp_get_default_domain](#) [yp_errno](#) y [yp_err_string](#)

yp_first (unknown)

devuelve la primera clave emparejada con el nombrado mapa.

```
string[] yp_first (cadena dominio, cadena mapa)
```

yp_first(nombre de la funcion)() Retorna la primera clave de valor pareada del mapa nombrado en el dominio, de otra manera FALSO.

Ejemplo 1. Ejemplo para el primer NIS

Ver tambien: [yp_get_default_domain](#) [yp_errno](#) y [yp_err_string](#)

yp_next (PHP 3>= 3.0.7, PHP 4)

Devuelve la siguiente clave tecleada en el nombre de mapa

```
string[] yp_next (cadena dominio, cadena mapa, cadena teclea)
```

yp_next() devuelve el siguiente par de valor tecleado en el mapa de nombres despues de la clave especificada o FALSO.

Ejemplo 1. Ejemplo para NIS siguiente

Ver tambien: [yp_get_default_domain](#) [yp_errno](#) y [yp_err_string](#)

yp_errno (unknown)

Retorna el codigo de error de la operacion previa.

```
int yp_errno ()
```

yp_errno() retorna el codigo de error de la operacion previa.

Los errores posibles son:

- 1 args para funcionar son malos
- 2 fallo de RPC- dominio ha sido unbound
- 3 no puede unir a servidor en este dominio
- 4 ningun tal mapa en dominio de servidor
- 5 ninguna tal llave en
- 6 interno yp error de cliente o servidor
- 7 fallo de asignacion de recurso
- 8 ningunos mas registros en base de datos de mapa
- 9 no puede comunicar with portmapper
- 10 no puede comunicar con ypbnd
- 11 no puede comunicar con ypserv
- 12 nombre de dominio local no conjunto
- 13 yp base de datos es malo
- 14 yp La version mismatch
- 15 violacion de acceso
- 16 base de datos ocupar

Ver tambien: [yp_err_string](#)

yp_err_string (unknown)

devuelve el mensaje de error asociado con la operacion previa.Util que indica el problema exacto.

```
cadena yp_err_string (nulo)
```

yp_err_string() Retorna el mensaje de error asociado con la operacion previa.Util para indicar que salio mal exactamente.

Ejemplo 1. Ejemplo para errores de NIS

Vea tambien: [yp_errno](#)

LXXXII. Funciones de Compresión

Este módulo usa la función de zlib (<http://www.info-zip.org/pub/infozip/zlib/>) de Jean-loup Gailly y Mark Adler para leer y grabar archivos comprimidos .gz, de un modo transparente. Con este módulo, es requisito usar una versión de zlib igual o posterior a 1.0.9.

Este módulo contiene versiones de la mayoría de las funciones de [Sistema de archivos](#) que funcionan con los archivos comprimidos con gzip (y con los no-comprimidos tambien, pero no con conectores (sockets)).

Pequeño código de ejemplo

Abre un archivo temporal y escribe en él, una cadena de prueba, y luego presenta el coterido del archivo dos veces

Ejemplo 1. Ejemplo de Zlib

```
<?php
$filename = tempnam('/tmp', 'zlibtest').'.gz';
print "<html>\n<head></head>\n<body>\n<pre>\n";
$s = "Sólo es una prueba, prueba, prueba, prueba, prueba, prueba!\n";
// Abre el archivo para escribirlo con máximo de compresión
$zp = gzopen($filename, "w9");
// Escribe la cadena en él
gzwrite($zp, $s);
// Cierra el fichero
gzclose($zp);
// Abre el fichero para lectura
$zp = gzopen($filename, "r");
// Lee 3 caracteres
print gzread($zp, 3);
// Salida hasta el final del fichero, para cerrarlo luego.
gzpassthru($zp);
print "\n";
// Abre el fichero y muestra su contenido (por segunda vez).
if (readgzfile($filename) != strlen($s)) {
    echo "Error con las funciones zlib!";
}
unlink($filename);
print "<pre>\n</h1></body>\n</html>\n";
?>
```


gzclose (PHP 3, PHP 4)

cierra un puntero a archivo-gz abierto

```
int gzclose (int zp)
```

El archivo-gz al que apunta *zp* se cierra.

Devuelve true (verdadero) si fue exitoso, si hubo errores devuelve false.

El puntero al archivo-gz debe ser válido y debe apuntar a un archivo correctamente abierto con **gzopen()**.

gzeof (PHP 3, PHP 4)

prueba el fin-de-archivo de un puntero de archivo-gz

```
int gzeof (int zp)
```

Devuelve verdadero si el puntero-a-archivo está en el fin-de-archivo, o ha ocurrido un error. De otro modo devuelve falso.

El puntero al archivo-gz debe ser válido y debe apuntar a un archivo correctamente abierto con **gzopen()**.

gzfile (PHP 3, PHP 4)

lee el archivo gz completo en un arreglo

```
array gzfile (string nombre_archivo [, int usar_include_path])
```

Identico a **readgzfile()**, solo que **gzfile()** devuelve el fichero en un arreglo.

Se puede usar el segundo parámetro opcional poniéndolo a "1", si se quiere que la función busque también el archivo en la trayectoria definida como [include_path](#).

Vea también **readgzfile()**, y **gzopen()**.

gzgetc (PHP 3, PHP 4)

toma caracteres de un archivo-gz

```
cadena gzgetc (int zp)
```

Devuelve una cadena conteniendo un carácter en particular (sin comprimir) leído del archivo al que apunta *zp*. Devuelve FALSE cuando está al final del archivo (al igual que **gzeof()**).

El puntero al archivo-gz debe ser válido y debe apuntar a un archivo correctamente abierto con **gzopen()**.

Vea también **gzopen()**, y **gzgets()**.

gzgets (PHP 3, PHP 4)

toma una linea del archivo apuntado

```
string gzgets (int zp, int longitud)
```

Devuelve una cadena (descomprimida) con longitud - 1 bytes de largo, leída del archivo apuntado por fp. La lectura finaliza cuando se han leído longitud - 1 bytes, ante un salto de linea o un fin-de-archivo (lo que ocurra primero).

Si ocurre un error, devuelve falso.

El puntero al archivo-gz debe ser válido y debe apuntar a un archivo correctamente abierto con **gzopen()**.

Vea también **gzopen()**, **gzgetc()**, y **fgets()**.

gzgetss (PHP 3, PHP 4)

toma una linea del archivo-gz apuntado y le quita los tags HTML

```
string gzgetss (int zp, int longitud [, string tags_permitidos])
```

Idéntica a **gzgets()**, excepto que gzgetss intenta quitar cualquier "tag"HTML o PHP del texto que lee.

Se puede usar el tercer parámetro para indicar qué parametros no deben ser extraídos.

Nota: *tags_permitidos* fue agregado en la versión de PHP 3.0.13, PHP4B3.

Véase también **gzgets()**, **gzopen()**, y **strip_tags()**.

gzopen (PHP 3, PHP 4)

open gz-file

```
int gzopen (string nombre_fichero, string modo [, int use_include_path])
```

Abre un archivo gzip (.gz) para lectura o escritura. El parámetro modo es, como en **fopen()** ("rb"o "wb") pero puede incluir tambien el nivel de compresión ("wb9") o la estrategia: 'f' para filtrado de datos como en "wb6f", 'h' para comprimir solo por Huffman igual que en "wb1h". (Ver la descripción de deflateInit2 en zlib.h para más información sobre el parámetro de estrategia.)

Gzopen puede usarse para leer o escribir un fichero que no esté en formato gzip; en ese caso **gzread()** leerá el archivo directamente, sin descomprimirlo.

Gzopen devuelve un puntero al archivo abierto y luego, cualquier proceso de lectura o escritura relacionado con ese descriptor de archivo, será transparente: se comprimirá o descomprimirá los datos según la necesidad, de manera automática.

Si la apertura fallase, se devolverá falso.

Se puede usar el tercer parámetro opcional, poniéndolo a "1", si se quiere buscar también el fichero en la trayectoria **include_path**.

Ejemplo 1. ejemplo de gzopen()

```
$fp = gzopen( "/tmp/file.gz" , "r" );
```

Vea también **gzclose()**.

gzpassthru (PHP 3, PHP 4)

Devuelve el remanente de datos de un fichero-gz

```
int gpassthru ( int zp )
```

Lee hasta el Fin-De-Archivo del archivo gz dado, y escribe los resultados (descomprimidos) en la salida standard.

Si ocurre un error, devuelve Falso.

El puntero al archivo-gz debe ser válido y debe apuntar a un archivo correctamente abierto con **gzopen()**.

El archivo-gz es cerrado cuando **gpassthru()** termina de leerlo (dejando *zp* sin utilidad).

gzputs (PHP 3, PHP 4)

Escribe al fichero-gz que se apunta

```
int gzputs ( int zp, string str [, int longitud] )
```

gzputs() es un alias a **gzwrite()**, y es absolutamente idéntico.

gzread (PHP 3, PHP 4)

Lee archivos-gz en modo Binario

```
string gzread ( int zp, int longitud )
```

gzread() lee hasta *longitud* bytes del archivo-gz apuntado por el parámetro *zp*. La lectura termina cuando se han leído *longitud* bytes (descomprimidos) o se alcanza el fin-de-archivo, lo que sucediera primero.

```
// Pone los contenidos del gz, a una cadena
$filename = "/usr/local/algo.txt.gz";
$zd = gzopen( $filename, "r" );
$contents = gzread( $zd, 10000 );
gzclose( $zd );
```

Ver también **gzwrite()**, **gzopen()**, **gzgets()**, **gzgetss()**, **gzfile()**, y **gpassthru()**.

gzrewind (PHP 3, PHP 4)

Reposiciona al puntero de archivo-gz, al inicio de aquel

```
int gzrewind ( int zp )
```

Reubica el indicador de posición del archivo, al comienzo del mismo.

si surge un error, devuelve 0.

El puntero al archivo-gz debe ser válido y debe apuntar a un archivo correctamente abierto con **gzopen()**.

Ver también: **gzseek()** y **gztell()**.

gzseek (PHP 3, PHP 4)

Posiciona el puntero del archivo-gz

```
int gzseek (int zp, int offset)
```

Busca la posición dentro del archivo *zp*, indicada en bytes por el parametro de desplazamiento *offset*. Es equivalente a llamar (en C) `gzseek(zp, offset, SEEK_SET)`.

Si el archivo se abre para lectura, la función será emulada, pero puede ponerse extremadamente lenta. Si se trata de escritura, solo está soportada la búsqueda hacia adelante; gzseek comprime entonces una secuencia de ceros hasta que alcanza la nueva ubicación.

Si se completa el pedido con éxito, devuelve 0; de lo contrario, devuelve -1. Note que la busqueda más allá del fin-de-archivo no se considera un error.

Vea también **gztell()** y **gzrewind()**.

gztell (PHP 3, PHP 4)

Indica la posición de lecto-escritura en el archivo

```
int gztell (int zp)
```

Devuelve la posición dentro del fichero referido por *zp*; p.e., su desplazamiento en el cuerpo del archivo.

Si hay algún error, devuelve falso.

El puntero al archivo-gz debe ser válido y debe apuntar a un archivo correctamente abierto con **gzopen()**.

Ver también **gzopen()**, **gzseek()** y **gzrewind()**.

gzwrite (PHP 3, PHP 4)

Escritura de ficheros gz en modo Binario

```
int gzwrite (int zp, string cadena [, int largo])
```

gzwrite() escribe el contenido de *cadena* al fichero gz referido por *zp*. Si el parámetro *largo* está presente, se detendrá la escritura luego de escribir *largo* bytes (descomprimidos) o al llegar el final de la *cadena*, lo que ocurriese primero.

Note que si se pasa el argumento *largo*, la opcion **magic_quotes_runtime** será ignorada y no se quitarán barras de la *cadena* en cuestión.

Ver también **gzread()**, **gzopen()**, y **gzputs()**.

readgzfile (PHP 3, PHP 4)

devuelve el fichero-gz

```
int readgzfile (string nombre_archivo [, int usar_trayectoria_include])
```

Lee el archivo, lo descomprime y lo escribe en la salida estándar.

Readgzfile() puede usarse para leer un archivo comprimido o no con gzip; en cuyo caso readgzfile() leerá directamente el archivo, sin descomprimirlo.

Devuelve el numero de bytes (descomprimidos) leídos del archivo, si ocurre un error, se devuelve falso y hasta que se llame como @readgzfile, se imprime un mensaje de error.

El archivo *nombre_archivo* se abrirá en el sistema de archivos y su contenido enviado a la salida estándar.

Puede usarse el segundo paametro opcional dándole el valor "1", si se quiere que se busque el archivo también dentro de la trayectoria "include": [include_path](#).

Ver también [gzpassthru\(\)](#), [gzfile\(\)](#), y [gzopen\(\)](#).

Parte V. Apéndices

Apéndice A. Migrando de PHP/FI 2.0 a PHP 3.0

Acerca de las incompatibilidades en PHP 3.0

PHP 3.0 ha sido reescrito desde cero. Tiene un intérprete propio, que es mucho más robusto y consistente que el de 2.0. 3.0 es también significativamente más rápido, y usa menos memoria. De cualquier modo, algunas de estas mejoras no fueron posibles sin alterar la compatibilidad, tanto de sintaxis como de funcionamiento.

Además, los desarrolladores de PHP han intentado clarificar la sintaxis y la semántica de PHP 3.0, y eso ha causado también algunas incompatibilidades. Creemos que, a la larga, estos cambios serán para mejor.

Este capítulo, intentará guiarle a través de las incompatibilidades que encontrará si migra de PHP/FI 2.0 a PHP 3.0, además de ayudarle a resolverlas. No se mencionarán aquí las herramientas nuevas, hasta que sea necesario.

Existe un programa que convierte automáticamente sus viejos guiones PHP/FI 2.0. Puede hallarlo en el subdirectorio convertor de la distribución 3.0 de PHP. Este programa, solo actúa en referencia a los cambios sintácticos, así que debería leer este capítulo detenidamente, de todos modos.

Tags de inicio y fin

Probablemente lo primero que note, es que los tags de inicio y fin de PHP han cambiado. La vieja forma <? > ha sido reemplazada por tres nuevas formas posibles:

Ejemplo A-1. Migración: Cambio de los tags de inicio y fin

```
<? echo "Esto es código PHP/FI 2.0.\n"; ?>
```

Como en la versión 2.0, PHP/FI también soporta esta variante:

Ejemplo A-2. Migración: Nuevos tags de inicio y fin, Variante 1

```
<? echo "Esto es código PHP 3.0!\n"; ?>
```

Note que el tag de fin, consiste ahora en un signo de interrogación y un mayor-que, mientras que antes era sólo un mayor-que. Sin embargo si Ud. planea usar XML en su servidor, tendrá problemas con esta nueva variante, porque PHP tratará de ejecutar las marcas en los documentos XML, como código PHP. Es por esto que se han introducido las siguientes variantes:

Ejemplo A-3. Migración: Nuevos tags de inicio y fin, Variante 2

```
<?php echo "Esto es código PHP 3.0!\n"; ?>
```

Algunas personas han tenido problemas con editores que no reconocen del todo los tags de instrucciones de procesamiento. Microsoft FrontPage es uno de esos editores, y a modo de remedio, se ha introducido también esta otra variante:

Ejemplo A-4. Migración: Nuevos tags de inicio y fin, Variante 3

```
<script language="php">
  echo "Esto es código PHP 3.0!\n";
</script>
```

sintaxis de if..endif

La manera ‘alternativa’ de escribir la declaración if/elseif/else, usando if();elseif(); else; endif; no puede implementarse eficientemente sin agregar una gran complejidad al intérprete 3.0. Por eso la sintaxis ha cambiado:

Ejemplo A-5. Migración: vieja sintaxis if..endif

```
if ($foo);
    echo "sip\n";
elseif ($bar);
    echo "casi\n";
else;
    echo "nop\n";
endif;
```

Ejemplo A-6. Migración: nueva sintaxis if..endif (si...finsi)

```
if ($foo):
    echo "sip\n";
elseif ($bar):
    echo "casi\n";
else:
    echo "nop\n";
endif;
```

Nótese que los punto-y-coma, han sido reemplazados por los los dos-puntos en todas las declaraciones, excepto en la de finalización (endif).

sintaxis de while (mientras)

Al igual que para if..endif, la sintaxis de while..endwhile se ha cambiado:

Ejemplo A-7. Migración: vieja sintaxis while..endwhile

```
while ($more_to_come);
    ...
endwhile;
```

Ejemplo A-8. Migración: nueva sintaxis while..endwhile

```
while ($more_to_come):
    ...
endwhile;
```

Aviso

Si se utiliza la sintaxis vieja de mientras...fin-mientras en PHP 3.0, el resultado es un bucle infinito.

Tipos de expresiones

PHP/FI 2.0 usó el lado izquierdo de las expresiones para determinar de qué tipo debía ser el resultado. PHP 3.0 toma ambos lados en cuenta cuando determina el tipo de resultado, y eso puede producir resultados impredecibles si ejecuta guiones 2.0 en 3.0.

Considere este ejemplo:

```
$a[0]=5;
$a[1]=7;

$key = key($a);
while ("" != $key) {
    echo "$keyn";
    next($a);
```

```
}
```

En PHP/FI 2.0, esto mostraría los dos índices de \$a. En PHP 3.0, no muestra nada. El motivo es que en PHP 2.0, al ser el argumento izquierdo de tipo cadena, se comparaba cadenas, y puesto que ciertamente " " no es igual a "0", el bucle continúa. En PHP 3.0, cuando se compara una cadena con un entero, se realiza una comparación de enteros (la cadena es convertida a entero). Esto es comparar entonces `atoi(" ")` que es 0, y `variablelist` que tambien es 0, y como `0==0`, el bucle no avanzará.

La solución a esto es simple, reemplaze la declaración while con:

```
while ((string)$key != "") {
```

Cambios en los mensajes de error

Los mensajes de error de PHP 3.0 son usualmente mas precisos que los de 2.0, pero ya no podrá ver el fragmento de código que causó el error. En vez de eso, se le mostrará el nombre de archivo y número de línea del error.

Evaluación booleana por corto-circuito

En PHP 3.0, la evaluación booleana es por cortocircuito. Eso significa que en una expresión como `(1 || prueba_me())`, la función `prueba_me()` no será ejecutada ya que nada puede cambiar el resultado de la expresión despues del 1.

Si bien es éste un detalle menor de la compatibilidad, puede provocar inesperados efectos colaterales

Retorno de valores en funciones verdadero/falso

La mayoría de las funciones internas han sido reescritas, asi que devuelven TRUE (verdadero) si hubo éxito, o FALSE (falso) si hubo falla, en oposición a los valores 0 y -1 de PHP/FI 2.0, para idénticas circunstancias. La nueva respuesta permite mas código lógico, como `$fp = fopen("/su/fichero") or fail("diablos!");`. Puesto que PHP/FI 2.0 no tiene reglas claras acerca de lo que devuelven las funciones cuando fallan, scripts de este tipo deberán revisarse manualmente despues de usar el conversor de 2.0 a 3.0.

Ejemplo A-9. Migración desde 2.0: Valores de retorno, código viejo

```
$fp = fopen($file, "r");
if ($fp == -1);
    echo("No se pudo abrir el fichero $file para su lectura<br>\n");
endif;
```

Ejemplo A-10. Migración desde 2.0: Valores de retorno, código nuevo

```
$fp = @fopen($file, "r") or print("No se pu-
do abrir el fichero $file para su lectura<br>\n");
```

Otras incompatibilidades

- El modulo Apache de 3.0 no soportará versiones de Apache anteriores a la 1.2. Es requisito Apache 1.2 o posterior.
- `echo()` no soportará las cadenas de formato. Use en su lugar la función `printf()`.
- En PHP/FI 2.0, un efecto colateral en la implementación hacía que `$foo[0]` tenga el mismo efecto que `$foo`. Esto ya no es así en PHP 3.0.

- La lectura de arreglos con \$array[] ya no está soportada

Esto es, no se puede recorrer un arreglo mediante un bucle que ejecute \$data = \$array[]. Use **current()** y **next()** en su lugar.

Tambien, \$array1[] = \$array2 no agrega los valores de \$array2 a \$array1, pero agrega \$array2 como la última entrada de \$array1. Véase tambien el soporte de arreglos multidimensionales.

- "+" ya no funciona como operador de concatenación de cadenas, sino que convierte los argumentos, a números, y realiza una suma numérica. Utilize " ." en su lugar.

Ejemplo A-11. Migración desde 2.0: concatenando cadenas

```
echo "1" + "1";
```

En PHP 2.0 esto mostraría 11, en PHP 3.0 sería 2. Use en su lugar:

```
echo "1"."1";
$a = 1;
$b = 1;
echo $a + $b;
```

Esto dá 2 igualmente en PHP 2.0 y 3.0.

```
$a = 1;
$b = 1;
echo $a.$b;
```

Esto dá 11 en PHP 3.0.

Apéndice B. Migrating from PHP 3.0 to PHP 4.0

What has changed in PHP 4.0

PHP 4.0 and the integrated Zend engine have greatly improved PHP's performance and capabilities, but great care has been taken to break as little existing code as possible. So migrating your code from PHP 3.0 to 4.0 should be much easier than migrating from PHP/FC 2.0 to PHP 3.0. A lot of existing PHP 3.0 code should be ready to run without changes, but you should still know about the few differences and take care to test your code before switching versions in production environments. The following should give you some hints about what to look for.

Parser behavior

Parsing and execution are now two completely separated steps, no execution of a file's code will happen until the complete file and everything it requires has completely and successfully been parsed.

One of the new requirements introduced with this split is that required and included files now have to be syntactically complete. You can no longer spread the different controlling parts of a control structure across file boundaries. That is you cannot start a `for` or `while` loop, an `if` statement or a `switch` block in one file and have the end of loop, `else`, `endif`, `case` or `break` statements in a different file.

It is still perfectly legal to include additional code within loops or other control structures, only the controlling keywords and corresponding curly braces `{ ... }` have to be within the same compile unit (file or `eval()`ed string).

This should not harm too much as spreading code like this should be considered as very bad style anyway.

Another thing no longer possible, though rarely seen in PHP 3.0 code is returning values from a required file. Returning a value from an included file is still possible.

Error reporting

Configuration changes

With PHP 3.0 the error reporting level was set as a simple numeric value formed by summing up the numbers related to different error levels. Usual values were 15 for reporting all errors and warnings or 7 for reporting everything but simple notice messages reporting bad style and things like that.

PHP 4.0 now has a greater set of different error and warning levels and comes with a configuration parser that now allows for symbolic constants to be used for setting up the intended behavior.

Error reporting level should now be configured by explicitly taking away the warning levels you do not want to generate error messages by x-oring them from the symbolic constant `E_ALL`. Sounds complicated? Well, let's say you want the error reporting system to report all but the simple style warnings that are categorized by the symbolic constant `E_NOTICE`. Then you'll put the following into your `php.ini`: `error_reporting = E_ALL & ~ (E_NOTICE)`. If you want to suppress warnings too you add up the appropriate constant within the braces using the binary or operator '|':
`error_reporting= E_ALL & ~ (E_NOTICE | E_WARNING).`

Aviso

Using the old values 7 and 15 for setting up error reporting is a very bad idea as this suppresses some of the newly added error classes including parse errors. This may lead to very strange behavior as scripts might no longer work without error messages showing up anywhere.

This has led to a lot of unreproducible bug reports in the past where people reported script engine problems they were not capable to track down while the true case was usually some missing ')' in a required file that the parser was not able to report due to a misconfigured error reporting system.

So checking your error reporting setup should be the first thing to do whenever your scripts silently die. The Zend engine can be considered mature enough nowadays to not cause this kind of strange behavior.

Additional warning messages

A lot of existing PHP 3.0 code uses language constructs that should be considered as very bad style as this code, while doing the intended thing now, could easily be broken by changes in other places. PHP 4.0 will output a lot of notice messages in such situations where PHP 3.0 didn't. The easy fix is to just turn off E_NOTICE messages, but it is usually a good idea to fix the code instead.

The most common case that will now produce notice messages is the use of unquoted string constants as array indices. Both PHP 3.0 and 4.0 will fall back to interprete these as strings if no keyword or constant is known by that name, but whenever a constant by that name had been defined anywhere else in the code it might break your script. This can even become a security risk if some intruder manages to redefine string constants in a way that makes your script give him access rights he wasn't intended to have. So PHP 4.0 will now warn you whenever you use unquoted string constants as for example in `$HTTP_SERVER_VARS[REQUEST_METHOD]`. Changing it to `$HTTP_SERVER_VARS['REQUEST_METHOD']` will make the parser happy and greatly improve the style and security of your code.

Another thing PHP 4.0 will now tell you about is the use of uninitialized variables or array elements.

Initializers

Static variable and class member initializers only accept scalar values while in PHP 3.0 they accepted any valid expression. This is, once again, due to the split between parsing and execution as no code has yet been executed when the parser sees the initializer.

For classes you should use constructors to initialize member variables instead. For static variables anything but a simple static value rarely makes sense anyway.

`empty("0")`

The perhaps most controversial change in behavior has happened to the behavior of the `empty()`. A String containing only the character '0' (zero) is now considered empty while it wasn't in PHP 3.0.

This new behavior makes sense in web applications, with all input fields returning strings even if numeric input is requested, and with PHP's capabilities of automatic type conversion. But on the other hand it might break your code in a rather subtle way, leading to misbehavior that is hard to track down if you do not know about what to look for.

Missing functions

While PHP 4.0 comes with a lot of new features, functions and extensions, you may still find some functions from version 3.0 missing. A small number of core functions has vanished because they do not work with the new scheme of splitting parsing and execution as introduced into 4.0 with the Zend engine. Other functions and even complete extensions have become obsolete as newer functions and extensions serve the same task better and/or in a more general way. Some functions just simply haven't been ported yet and finally some functions or extensions may be missing due to license conflicts.

Functions missing due to conceptual changes

As PHP 4.0 now separates parsing from execution it is no longer possible to change the behavior of the parser (now embedded in the Zend engine) at runtime as parsing already happened by then. So the function `short_tags()` has ceased to exist. You can still change the parser's behavior by setting appropriate values in the `php.ini` file.

Another feature from PHP 3.0 that didn't make it into 4.0 is the experimental debugging interface as described in ??? in this manual. A separate true debugger is promised to be released as a Zend product but hasn't become visible yet.

Deprecate functions and extensions

The Adabas and Solid database extensions are no more. Long live the unified ODBC extension instead.

Changed status for **unset()**

unset(), although still available, is implemented a little different in PHP 4.0 so that it no longer counts as a 'real' function.

This has no direct consequences as the behavior of **unset()** didn't change, but testing for "unset" using **function_exists()** will return **false** as it would with other lowlevel functions like **echo()**.

Another more practical change is that it is no longer possible to call **unset()** indirectly, that is `$func="unset"; $func($somevar)` won't work anymore.

PHP 3.0 extension

Extensions written for PHP 3.0 will not work with PHP 4.0 anymore, neither as binaries nor at the source level. It is not too difficult to port your extensions to PHP 4.0 if you have access to the original sources. A detailed description of the actual porting process is not part of this text (yet).

Variable substitution in strings

PHP 4.0 adds a new mechanism to variable substitution in strings. You can now finally access object member variables and elements from multidimensional arrays within strings.

To do so you have to enclose your variables with curly braces with the dollar sign immediately following the opening brace: `{$...}`

To embed the value of an object member variable into a string you simply write `"text {$obj->member} text"` while in PHP 3.0 you had to use something like `"text ".$obj->member."text"`.

This should lead to more readable code, while it may break existing scripts written for PHP 3.0. But you can easily check for this kind of problem by checking for the character combination `{$` in your code and by replacing it with `\{$` with your favourite search-and-replace tool.

Cookies

PHP 3.0 had the bad habit of setting cookies in the reverse order of the **setcookie()** calls in your code. PHP 4.0 breaks with this habit and creates the cookie header lines in exactly the same order as you set the cookies in the code.

This might break some existing code, but the old behaviour was so strange to understand that it deserved a change to prevent further problems in the future.

Apéndice C. Desarrollo en PHP

Añadiendo funciones al PHP3

Prototipo de Función

Todas las funciones son como esta:

```
void php3_algo(INTERNAL_FUNCTION_PARAMETERS) {
}
```

Incluso si su función no lleva argumentos, es así como se le llama.

Argumentos de Función

Los argumentos son siempre de tipo pval. Este tipo contiene una unión que es el tipo actual del argumento. Así, si su función tiene dos argumentos, deberá hacer algo como lo sigue al principio de la misma:

Ejemplo C-1. Extrayendo argumentos de función

```
pval *arg1, *arg2;
if (ARG_COUNT(ht) != 2 || getParameters(ht, 2, &arg1, &arg2)==FAILURE) {
    WRONG_PARAM_COUNT;
}
```

NOTA: Los argumentos pueden pasarse tanto por valor como por referencia. En ambos casos, necesitará pasar &(pval *) a getParameters. Si desea comprobar si el enésimo parámetro le ha sido enviado o no por referencia, puede utilizar la función ParameterPassedByReference(ht,n). Esta devolverá 1 ó 0, según corresponda.

Cuando cambie alguno de los parámetros pasados, tanto si son enviados por referencia o por valor, puede volver a comenzar con éste llamando la función pval_destructor sobre el mismo, o, si es una ARRAY a la que quiere añadir algo, puede utilizar funciones similares a las incluidas en internal_functions.h, que manipulan el valor return_value como si fuera de tipo ARRAY.

Además, si cambia un parámetro a IS_STRING, asegúrese primero de asignar el valor y el tamaño a la cadena creada por estrdup() y sólo entonces cambiar su tipo a IS_STRING. Si modifica la cadena de un parámetro que ya es IS_STRING o IS_ARRAY, deberá primero aplicarle la función pval_destructor.

Argumentos de Función Variables

Una función puede tomar un número variable de argumentos. Si su función puede tomar tanto 2 como 3 argumentos, utilice el siguiente código:

Ejemplo C-2. Argumentos de función variables

```
pval *arg1, *arg2, *arg3;
int arg_count = ARG_COUNT(ht);

if (arg_count < 2 || arg_count > 3 ||
    getParameters(ht, arg_count, &arg1, &arg2, &arg3)==FAILURE) {
    WRONG_PARAM_COUNT;
}
```

Usando los Argumentos de Función

El tipo de cada argumento se guarda en el campo type del pval. Este tipo puede ser:

Tabla C-1. Tipos Internos de PHP

IS_STRING	Cadena
IS_DOUBLE	Coma flotante de doble precisión
IS_LONG	Entero largo
IS_ARRAY	Matriz
IS_EMPTY	Nada
IS_USER_FUNCTION	??
IS_INTERNAL_FUNCTION	?? (N.D.: si alguno de estos no se puede pasar a una función, bórrese)
IS_CLASS	??
IS_OBJECT	??

Si obtiene un argumento de un tipo y desea utilizarlo como si fuera de otro, o si quiere forzar a que un argumento sea de un tipo determinado, puede usar una de las siguientes funciones de conversión:

```
convert_to_long(arg1);
convert_to_double(arg1);
convert_to_string(arg1);
convert_to_boolean_long(arg1); /* Si la cadena es "" o "0" pasa a ser 0; si no, vale 1 */
convert_string_to_number(arg1); /* Convierte la cadena a LONG o a DOUBLE, dependiendo de su contenido */
```

Estas funciones convierten el valor in-situ. No devuelven nada.

El argumento real es almacenado en una unión cuyos miembros son:

- IS_STRING: arg1->value.str.val
- IS_LONG: arg1->value.lval
- IS_DOUBLE: arg1->value.dval

Manejo de Memoria en las Funciones

La memoria necesitada por una función deberá ser asignada usando emalloc() o estrdup(). Estas son funciones abstractas de manejo de memoria que son similares a las funciones normales malloc() y strdup(). La memoria deberá liberarse con efree().

Hay dos tipos de memoria en este programa: la memoria que se devuelve al troceador (parser) en una variable, y la memoria que se necesita para almacenamiento temporal de datos en sus funciones. Cuando asigne una cadena a una variable que se devolverá al troceador deberá asegurarse previamente de asignar la memoria con emalloc() o con estrdup(). Esta memoria NUNCA debe ser liberada por usted, salvo si más adelante, en la misma función, sobreescribe la asignación original (aunque este hábito de programación no es bueno).

Para cada trozo de memoria temporal/permanente que precise en sus funciones/librería deberá utilizar las funciones emalloc(), estrdup(), y efree(). Estas se comportan EXACTAMENTE como sus funciones equivalentes. Cualquier cosa que asigne con emalloc() o estrdup() deberá liberarla con efree() en uno u otro momento, salvo que se suponga que deba permanecer activa hasta el final del programa; de otro modo, se producirá una fuga de memoria. El significado de "estas se comportan exactamente como sus funciones equivalentes" es: si llama a efree() sobre algo que no ha sido asignado

previamente con emalloc() o con estrdup(), puede provocar un fallo de segmentación. Por ello debe tener cuidado y liberar toda la memoria desperdiciada.

Si compila con "-DDEBUG", el PHP3 mostrará una lista de toda la memoria que fue asignada usando emalloc() y estrdup(), pero que nunca fue liberada con efree(), al terminar de ejecutar el guión especificado.

Asignando Variables en la Tabla de Símbolos

Están disponibles una serie de macros que hacen más fácil el asignar una variable en la tabla de símbolos:

- SET_VAR_STRING(nombre,valor)¹
- SET_VAR_DOUBLE(nombre,valor)
- SET_VAR_LONG(nombre,valor)

¹

Las tablas de símbolos en PHP 3.0 se implementan como tablas hash (con extracto). En todo momento, &symbol_table es un puntero a la tabla de símbolos 'principal', mientras que active_symbol_table apunta a la tabla de símbolos activa (pueden ser idénticas, al principio de todo, o diferentes, si se está dentro de una función).

Los ejemplos siguientes utilizan 'active_symbol_table'. Deberá reemplazarla por &symbol_table si desea trabajar específicamente con la tabla de símbolos 'principal'. También se pueden aplicar las mismas funciones a matrices, como se explica más abajo.

Ejemplo C-3. Comprobando si \$algo existe en una tabla de símbolos

```
if (hash_exists(active_symbol_table,"algo",sizeof("algo"))) { existe... }
else { no existe }
```

Ejemplo C-4. Hallando el tamaño de una variable en una tabla de símbolos

```
hash_find(active_symbol_table,"algo",sizeof("algo"),&valptr);
check(valptr.type);
```

Las matrices en PHP 3.0 se implementan utilizando las mismas tablas hash que para las tablas de símbolos. Ello quiere decir que las dos funciones anteriores se pueden usar también para comprobar variables dentro de matrices.

Si desea definir un nuevo símbolo de matriz en una tabla de símbolos, deberá hacer lo que sigue.

Primero, deberá comprobar si ya existe usando hash_exists() o hash_find() y abortar la ejecución de forma apropiada.

Luego inicialice la matriz:

Ejemplo C-5. Inicializando una nueva matriz

```
pval matriz;
if (array_init(&matriz) == FAILURE) { falló... };
hash_update(active_symbol_table,"algo",sizeof("algo"),&matriz,sizeof(pval),NULL);
```

Este código declara una nueva matriz, llamada \$algo, en la tabla de símbolos activa. Esta matriz está vacía.

Ahora se muestra cómo añadirle elementos:

Ejemplo C-6. Añadir entradas a una nueva matriz

```

pval elemento;

elemento.type = IS_LONG;
elemento.value.lval = 5;

/* define $algo["bar"] = 5 */
hash_update(matriz.value.ht, "bar", sizeof("bar"), &elemento, sizeof(pval), NULL);

/* define $algo[7] = 5 */
hash_index_update(matriz.value.ht, 7, &elemento, sizeof(pval), NULL);

/* define el siguiente puesto libre en $algo[],
 * $algo[8], como 5 (funciona como en php2)
 */
hash_next_index_insert(matriz.value.ht, &elemento, sizeof(pval), NULL);

```

Si desea modificar un valor que ha insertado en una matriz asociativa, deberá primero extraerlo de ella. Para evitar esa sobrecarga, puede pasarle un puntero pval ** a la función para insertar en una matriz asociativa, y será actualizada con la dirección pval * del elemento insertado dentro de la matriz. Si dicho valor es NULL (como en todos los ejemplos anteriores), el parámetro se ignora.

hash_next_index_insert() usa más o menos la misma lógica que "\$algo[] = bar;" en el PHP 2.0.

Si está preparando una matriz como valor devuelto por una función, puede inicializar la misma como antes, haciendo:

```
if (array_init(return_value) == FAILURE) { falló...; }
```

... y luego añadiéndole valores con las funciones auxiliares:

```

add_next_index_long(return_value, val_long);
add_next_index_double(return_value, val_double);
add_next_index_string(return_value, estrdup(val_cadena));

```

Por supuesto, si la adición no se realiza justo después de inicializar la matriz, probablemente tenga que buscarla antes:

```

pval *matriz;

if (hash_find(active_symbol_table, "algo", sizeof("algo"), (void **) &ma-
triz) == FAILURE) { no se hayó... }
else { usar matriz->value.ht... }

```

Nótese que hash_find recibe un puntero a un puntero a pval, y no un puntero a pval.

Casi cualquier función de matrices asociativas devuelve SUCCESS o FAILURE (excepto por hash_exists(), que devuelve un valor lógico de certeza).

Devolviendo valores simples

Están disponibles varias macros para facilitar la devolución de valores de una función.

Todas las macros RETURN_* fijan el valor y retornan de la función:

- RETURN
- RETURN_FALSE
- RETURN_TRUE
- RETURN_LONG(l)

- RETURN_STRING(s,dup) Si dup es true, duplica la cadena
- RETURN_STRINGL(s,l,dup) Devuelve la cadena (s) especificando el largo (l).
- RETURN_DOUBLE(d)

Las macros RETVAL_* fijan el valor, pero no retornan.

- RETVAL_FALSE
- RETVAL_TRUE
- RETVAL_LONG(l)
- RETVAL_STRING(s,dup) Si dup es true, duplica la cadena
- RETVAL_STRINGL(s,l,dup) Devuelve la cadena (s) especificando el largo (l).
- RETVAL_DOUBLE(d)

Las macros anteriores harán un estrdup() del argumento 's', de modo que puede liberar con seguridad el argumento después de llamar a la macro, o, alternativamente, utilizar memoria asignada estáticamente.

Si su función devuelve respuestas lógicas de éxito/error, use siempre RETURN_TRUE y RETURN_FALSE respectivamente.

Devolviendo valores complejos

Su función también puede devolver un tipo de datos complejo, tal como un objeto o una matriz.

Devolviendo un objeto:

1. Llame a object_init(return_value).
2. Rellénela con valores. Las funciones disponibles para ello son listadas más abajo.
3. Posiblemente registre funciones para este objeto. Para obtener valores del objeto, la función deberá de obtener "this" desde la active_symbol_table. Su tipo deberá ser IS_OBJECT, y básicamente se trata de una matriz asociativa estándar (es decir, que podrá usar funciones de matriz asociativa sobre .value.ht). El registro en sí de la función se puede hacer utilizando:

```
add_method( return_value, nombre_func, puntero_func );
```

Las funciones utilizadas para llenar un objeto son:

- add_property_long(return_value, nombre_propiedad, l) - Añade una propiedad llamada 'nombre_propiedad', de tipo long, y con valor 'l'
- add_property_double(return_value, nombre_propiedad, d) - Igual, pero añadiendo un double
- add_property_string(return_value, nombre_propiedad, cad) - Igual, pero añadiendo una cadena
- add_property_stringl(return_value, nombre_propiedad, cad, l) - Igual, pero añadiendo una cadena de longitud 'l'

Devolviendo una matriz:

1. Llame a array_init(return_value).
2. Rellénela con valores. Las funciones disponibles para ello son listadas más abajo.

Las funciones utilizadas para rellanar una matriz son:

- add_assoc_long(return_value,clave,l) - añade un elemento asociativo con clave 'clave' y valor long 'l'
- add_assoc_double(return_value,clave,d)
- add_assoc_string(return_value,clave,cad,duplicar)
- add_assoc_stringl(return_value,clave,cad,largo,duplicar) - especifica el largo de la cadena
- add_index_long(return_value,indice,l) - añade un elemento en la posición 'indice' con valor long 'l'
- add_index_double(return_value,indice,d)
- add_index_string(return_value,indice,cad)
- add_index_stringl(return_value,indice,cad,largo) - especifica el largo de la cadena
- add_next_index_long(return_value,l) - añade un elemento a la matriz en la próxima posición libre con valor long 'l'
- add_next_index_double(return_value,d)
- add_next_index_string(return_value,cad)
- add_next_index_stringl(return_value,cad,largo) - especifica el largo de la cadena

Usando la lista de recursos

El PHP 3.0 tiene una forma estandarizada de tratar con distintos tipos de recursos. Esto sustituye a las listas enlazadas locales del PHP 2.0.

Funciones disponibles:

- php3_list_insert(ptr, tipo) - devuelve el 'id' del recurso recién insertado
- php3_list_delete(id) - borra el recurso con el id especificado
- php3_list_find(id,*tipo) - devuelve el puntero al recurso con el id especificado, y actualiza 'tipo' al tipo del mismo

Estas funciones se utilizan típicamente para controladores SQL, pero pueden utilizarse para cualquier otra cosa, como, por ejemplo, para mantener descriptores de archivo.

El código típico de un lista sería como este:

Ejemplo C-7. Añadiendo un nuevo recurso

```
RESOURCE *recurso;

/* ...asignar memoria para el recurso y adquirirlo... */
/* añadir un recurso a la lista */
return_value->value.lval = php3_list_insert((void *) recurso, LE_RESOURCE_TYPE);
return_value->type = IS_LONG;
```

Ejemplo C-8. Utilizando un recurso existente

```
pval *id_recurso;
RESOURCE *recurso;
int tipo;

convert_to_long(id_recurso);
recurso = php3_list_find(id_recurso->value.lval, &tipo);
if (tipo != LE_RESOURCE_TYPE) {
```

```

php3_error(E_WARNING,"el recurso número %d tiene el tipo equivocado",id_recurso-
>value.lval);
RETURN_FALSE;
}
/* ...usar recurso... */

```

Ejemplo C-9. Borrando un recurso

```

pval *id_recurso;
RESOURCE *recurso;
int tipo;

convert_to_long(id_recurso);
php3_list_delete(id_recurso->value.lval);

```

Los tipos de recursos deben registrarse en php3_list.h, en la enumeración list_entry_type. Además, hay que añadir código de desconexión para cada tipo de recurso definido en la función list_entry_destructor() de list.c (incluso si no hay nada que hacer para la desconexión, deberá añadir un caso vacío).

Utilizando la tabla de recursos persistentes

El PHP 3.0 tiene una forma estándar de almacenar recursos persistentes (es decir, recursos que se mantienen entre accesos). El primer módulo que utilizó esta característica fue el MySQL y tras él fue el mSQL, así que uno puede hacerse una buena idea de cómo utilizar un recurso persistente leyendo mysql.c. Las funciones a revisar son:

```

php3_mysql_do_connect
php3_mysql_connect()
php3_mysql_pconnect()

```

La idea general de los módulos persistentes es:

1. Codifique todos sus módulos para que funcionen con la lista regulares de recursos mencionadas en la sección (9).
2. Codifique funciones extra de conexión que comprueben si el recurso ya está en la lista de recursos persistentes. Si ya está, regístrello en la lista regular como un puntero a la lista de recursos persistentes (debido a 1., el resto del código deberá funcionar de inmediato). Si no está en la lista, créelo, añádalo a la lista de recursos persistentes Y añada un puntero al mismo desde la lista regular de recursos. Así todo el código funcionará porque está en la lista regular, pero en la siguiente conexión el recurso ya estará en la lista persistente y podrá ser usado sin re-creararlo. Deberá registrar estos recursos con un tipo diferente (por ejemplo, LE MYSQL LINK para el enlace no persistente y LE MYSQL PLINK para un enlace persistente).

Si se leyera mysql.c, notaría que, salvo por que hay una función de conexión más compleja, no hay que cambiar nada más del resto del módulo.

Existe exactamente la misma interfaz para la lista de recursos regular y para la lista de recursos persistente, pero cambiando únicamente 'lista' por 'listap':

- php3 plist_insert(ptr, tipo) - devuelve el 'id' del recurso recién insertado
- php3 plist_delete(id)- borra el recurso con el id especificado
- php3 plist_find(id,*tipo) - devuelve el puntero al recurso con el id especificado, y actualiza 'tipo' al tipo del mismo

Sin embargo, es más que probable que estas funciones se muestren inútiles cuando intente implementar un módulo persistente. Típicamente usted querrá usar el hecho de que la tabla de recursos persistentes es en realidad una matriz asociativa. Por ejemplo, en los módulos MySQL/mSQL, cuando hay una llamada a pconnect() (conexión persistente), la

función combina en una cadena el servidor/usuario/clave que se pasaron a la función y codifica el enlace SQL con esta cadena como clave. La siguiente vez que alguien llame a pconnect() con el mismo servidor/usuario/clave, se generará la misma clave, y la función hará el enlace SQL en la lista persistente.

Hasta que se documente mejor, deberá mirar en mysql.c o en msqli.c para ver como utilizar las capacidades de matriz asociativa de la listap.

Una cosa importante: a los recursos que van a parar a la lista de recursos persistentes *NO* se les debe asignar memoria usando el gestor de memoria del PHP, es decir, que NO deben ser creados utilizando emalloc() o estrdup(), etc. En este caso se debe usar las funciones habituales malloc(), strdup(), etc. La razón para esto es simple: al final de la petición (final del acceso), se borran todos los trozos de memoria asignados con el gestor de memoria del PHP. Como la lista persistente se supone que no se debe borrar al final de una petición, no se debe utilizar el gestor de memoria del PHP para asignar memoria a los recursos de la misma.

Cuando registre un recurso que vaya a estar en la lista persistente, deberá añadir destructores tanto a la lista persistente como a la no persistente. El destructor de la lista no persistente no deberá hacer nada. El de la lista persistente deberá liberar adecuadamente los recursos obtenidos por dicho tipo (por ejemplo, memoria, enlaces SQL, etc.). Tal y como pasa para los recursos no persistentes, DEBERÁ añadir destructores para cada recurso aunque no sean necesarios y estén vacíos. Recuerde que como no se pueden usar emalloc() y similares en conjunción con la lista persistente, tampoco podrá utilizar efree() aquí.

Añadiendo directivas de configuración en tiempo de ejecución

Muchas de las características del PHP3 pueden ser configuradas en tiempo de ejecución. Estas directivas de configuración pueden aparecer tanto en el fichero php3.ini o, en el caso de la versión de módulo del Apache, en los archivos .conf del propio Apache. La ventaja de tenerlos en los archivos .conf del Apache es que se pueden configurar directorio por directorio. Esto quiere decir que cada uno puede tener un cierto safemodeexecdir, por ejemplo, mientras otro directorio puede tener otro. Esta granularidad en la configuración es especialmente útil cuando un servidor soporta múltiples servidores virtuales.

Los pasos necesarios para añadir una nueva directiva:

1. Añada la directiva a la estructura php3_ini_structure en mod_php3.h.
2. En main.c, edite la función php3_module_startup y añada la llamada a cfg_get_string() o a cfg_get_long() según se requiera.
3. Añada la directiva, las restricciones y un comentario a la estructura php3_commands en mod_php3.c. Cuidado con la parte de restricciones. Las de tipo RSRC_CONF sólo puede aparecer en los archivos .conf del Apache. Las directivas de tipo OR_OPTIONS pueden aparecer en cualquier parte, incluso en los habituales archivos .htaccess.
4. Añada el elemento apropiado para su directiva, bien en php3take1handler(), bien en php3flaghandler().
5. Necesita añadir su nueva directiva a la sección de configuración de la función _php3_info() en functions/info.c.
6. Y finalmente, por supuesto, deberá utilizar su nueva directiva en algún sitio. Estará accesible como php3_ini.directive.

Llamando a Funciones del Usuario

Para llamar a funciones del usuario desde una función interna, deberá usar la función **call_user_function()**.

call_user_function() devuelve SUCCESS si tiene éxito y FAILURE en caso de que la función no sea hallada. ¡Deberá comprobar ese valor de retorno! Si devuelve SUCCESS, debe usted ocuparse de destruir el pval devuelto (o devolverlo como el valor de retorno de su función). Si devuelve FAILURE, el valor de valret no está definido y no debe tocarlo.

Todas las funciones internas que llaman a funciones de usuario *deben* ser reentrantes. Entre otras cosas, esto quiere decir que no se utilicen variables globales ni estáticas.

call_user_function() lleva 6 argumentos:

HashTable *tabla_funciones

Esta es la matriz asociativa en la que se buscará la función.

pval *objeto

Este es un puntero a un objeto sobre el que se invoca la función. Deberá valer NULL si se llama a una función global. Si no es NULL (es decir, si apunta a un objeto), el argumento tabla_funciones se ignora y se toma su valor a partir de la codificación del objeto. El objeto **puede** ser modificado por dicha función (ésta accede a él a través de \$this). Si por alguna razón no desea que eso ocurra, envíe entonces una copia del objeto.

pval *nombre_func

El nombre de la función a llamar. Debe ser un pval de tipo IS_STRING, con nombre_func.str.val y nombre_func.str.len fijados a los valores apropiados. El nombre_func es modificado por call_user_function(), que lo convierte a minúsculas. Si necesita preservar el nombre, envíe una copia del mismo.

pval *valret

Un puntero a una estructura pval, en la que se guarda el valor de retorno de la función. Hay que asignar espacio a la estructura previamente, porque la función call_user_function() NO lo asigna por sí misma.

int num_params

El número de parámetros que se pasan a la función.

pval *params[]

Una matriz de punteros a los valores que se pasarán como argumentos a la función. El primer argumento está en el elemento 0, el segundo en el elemento 1, etc. La matriz es una matriz de punteros a pval. Los punteros se envían tal cual a la función, lo que quiere decir que si la función modifica sus argumentos, se modifican los valores originales (paso por referencia). Si no desea ese comportamiento, pase una copia.

Informando de errores

Para informar de errores desde una función interna, deberá llamar la función **php3_error()**. Esta lleva al menos dos parámetros: el primero el es nivel del error, y el segundo es la cadena de formato para el mensaje de error (como en una llamada estándar a **printf()**). Cualquiera de los argumentos siguientes son para la cadena de formato. Los niveles de error son:

E_NOTICE

Por defecto se visualizan las noticas, e indican que el guión encontró algo que podría indicar un error, pero que también podría ocurrir durante el curso normal de la ejecución del mismo. Por ejemplo, al intentar acceder al valor de una variable que no ha sido fijado, o llamar a **stat()** sobre un fichero que no existe.

E_WARNING

Por defecto se visualizan los avisos, pero no se interrumpe la ejecución del guión. Estos indican un problema que debiera haberse interceptado en el guión antes de la llamada. Por ejemplo, llamar **ereg()** con una expresión regular no válida.

E_ERROR

Por defecto se visualizan los errores y la ejecución del guión se detiene cuando la función retorna. Estos indican errores irrecuperables, como un problema de asignación de memoria.

E_PARSE

Los errores de troceo sólo debe generarlos el troceador. El código se lista aquí con objeto de ser exhaustivos.

E_CORE_ERROR

Este es similar al E_ERROR, pero generado por el núcleo del PHP. Las funciones no deben generar este tipo de error.

E_CORE_WARNING

Este es similar al E_WARNING, pero generado por el núcleo del PHP. Las funciones no deben generar este tipo de error.

Nota: "N.D.:" significa Nota del Documentador y es un texto interno

Notas

Tenga cuidado aquí. El valor a usar se debe asignar dinámicamente y de forma manual, pues el código de manejo de memoria intentará liberar este puntero más adelante. Nunca pase memoria asignada de forma estática a SET_VAR_STRING.

Apéndice D. El debugger de PHP

Usando el Debugger

El debugger interno de PHP es útil para localizar fallos que se resisten. El debugger funciona conectándose a un puerto TCP cada vez que PHP comienza. Todos los mensajes de error de esa petición serán enviados a esta conexión TCP. Esta información se entiende que es para un "servidor de debugger" que puede ejecutarse en un IDE o en un editor programable (como Emacs).

Como poner en marcha el debugger:

1. Establezca un puerto TCP para el debugger en el [archivo de configuración \(debugger.port\)](#) y activelo (`debugger.enabled`).
2. Ponga en marcha un módulo de escucha de TCP en algún sitio (por ejemplo `socket -l -s 1400` en UNIX).
3. En su código, ejecute "`debugger_on(host)`", donde `host` es la dirección IP o el nombre de el host ejecutando un módulo de escucha de TCP.

Ahora, todos los avisos, notificaciones, etc. se mostrarán en ese módulo de escucha, *incluso si lo ha desactivado con `error_reporting()`*.

Protocolo del debugger

El protocolo del debugger está basado en líneas. Cada línea tiene un *tipo*, y varias líneas componen un *mensaje*. Cada mensaje comienza con una línea del tipo `start` y termina con una línea del tipo `end`. PHP puede enviar líneas para diferentes mensajes simultáneamente.

Una línea tiene este formato:

fecha hora host(pid) tipo: datos del mensaje

fecha

Fecha en formato ISO 8601 (`aaaa-mm-dd`)

hora

Hora incluyendo microsegundos: `hh:mm:uuuuuu`

host

Nombre DNS o dirección IP del host donde el script de error fue generado.

pid

PID (id proceso) en el `host` del proceso en que el script de PHP generó este error.

tipo

Tipo de la línea. Dice al programa que recibe que debe considerar los datos siguientes como:

Tabla D-1. Tipos de línea del debugger

Nombre	Significado
<code>start</code>	Informa al programa que recibe que un mensaje del debugger comienza aquí. El contenido de <code>data</code> será el tipo del mensaje de error, listados debajo.
<code>message</code>	El mensaje de error de PHP.

Nombre	Significado
location	Nombre del fichero y número de línea donde ocurrió el error. La primera línea con location siempre contendrá la localización de mayor nivel. data contendrá fichero:línea. Siempre habrá una línea de tipo location después de message y después de cada function.
frames	Número de marcos en la pila. Si hay cuatro marcos, espere información sobre los cuatro niveles de las funciones llamadas. Si no hay una línea de tipo "frames", la profundidad se asume que es 0 (el error ocurrió en el nivel superior).
function	Nombre de la función donde ocurrió el error. Será repetida una vez por cada nivel en la pila de funciones.
end	Informa al programa que recibe que el mensaje del debugger termina aquí.

data

Línea de datos.

Tabla D-2. Tipos de error del debugger

Debugger	PHP Internal
warning	E_WARNING
error	E_ERROR
parse	E_PARSE
notice	E_NOTICE
core-error	E_CORE_ERROR
core-warning	E_CORE_WARNING
unknown	(any other)

Ejemplo D-1. Ejemplo de mensaje del debugger

```

1998-04-05 23:27:400966 lucifer.guardian.no(20481) start: notice
1998-04-05 23:27:400966 lucifer.guardian.no(20481) message: Uninitialized variable
1998-04-05 23:27:400966 lucifer.guardian.no(20481) location: (null):7
1998-04-05 23:27:400966 lucifer.guardian.no(20481) frames: 1
1998-04-05 23:27:400966 lucifer.guardian.no(20481) function: display
1998-04-05 23:27:400966 lucifer.guardian.no(20481) location: /home/ssb/public_html/test.php3:10
1998-04-05 23:27:400966 lucifer.guardian.no(20481) end: notice

```

