Apuntes de PHP

Salvador Ramirez Flandes

22 de noviembre de 2000

Índice General

1 Introduccion

Es un lenguaje script que se usa principalmente dentro del HTML y se ejecuta en el servidor. En el servidor se puede ejecutar como modulo del web server o como un script externo tal como perl por ejemplo. Ejemplo:
<html><head><title>Prueba</title></head>
<body>
<?php echo “hello world”; ?>
</body>
</html>

PHP puede hacer todo lo un CGI hacer dado que puede ser considerado como un script CGI que se ejecuta en el servidor.

El fuerte de PHP son las bases de datos. PHP actualmente soporta: Oracle (7 y 8), Sybase, Adabas D, Informix, dBase, mSQL, mySQL, Unix dbm, etc.

PHP tambien soporta protocolos Internet tales como: IMAP, SNMP, NNTP, POP3, HTTP, etc.

Con PHP se pueden crear socket de red.

2 Sintaxis

Hay 4 formas de escribir codigo PHP dentro de HTML:

1. <? echo ("this is the simplest, an SGML processing instruction\n"); ?>
2. <?php echo("if you want to serve XML documents, do like this\n"); ?>

3. <script language="php">
   echo ("some editors (like FrontPage) don't like processing instructions");
   </script>
4. <% echo ("You may optionally use ASP-style tags"); %>
   <%= $variable; # This is a shortcut for "<%echo .." %>

La forma 2 es la mas usada. La primera debe ser habilitada desde la configuracion del web server (configure –enable-short-tags) al igual que la 4.

Todos los estamentos deben ser terminados en ;

Existen 3 tipos de comentarios: /* */ (de C) , // (de C++) y # (de unix shell).

2.1 Tipos de datos

Los tipos de dato de PHP son: array, floating point, integer, object y string. El tipo de dato no es explicitamente escrito por el programador sino que en tiempo de ejecucion PHP decide el tipo de dato que mas se ajusta al dato en uso. Sin embargo tambien existe el casting en PHP:
$a = (double)$b;, $a = (int)$b;, $a = (real)$b;, $a = (string)$b;, $a = (object)$b;
Ejemplos:
Enteros: $a = 1234; (decimal), $a = 0x12; (hex equivale a 18 decimal)
Floating point (doubles): $a = 1.234;, $a = 1.2e3;

2.1.1 Strings

Existen 3 formas de crear strings en PHP.

$a = “Probando $b”;
En este caso las variables dentro del string seran expandidas a sus valores. Se puede usar \ para poner caracteres especiales tales como \n, \r, \t, \\, \$, \”, etc.

$a = ’Probando $b’;
En este caso las variables dentro del string no seran expandidas. Ademas los unicos caracteres especiales que se tomaran en cuenta son \\ y \’.

```
$a = <<<EOD
Hola “$nombre”\n
EOD;
```

En este caso se expanden variables y no es necesarios usar \ para cada caracter especial. Esta forma se agrego en PHP4.

(gettype($a) en PHP se puede usar para obtener el tipo de dato de un variable).

Los strings en PHP se concatenan con un punto “.”. Los caracteres de un string se pueden acceder con los indices del string en cuestion. A continuacion algunos ejemplos:

```
<?php
/* Assigning a string. */
$str = "This is a string";

/* Appending to it. */
$str = $str . " with some more text";

/* Another way to append, includes an escaped newline. */
$str .= " and a newline at the end.\n";

/* This string will end up being '<p>Number: 9</p>' */
$num = 9;
$str = "<p>Number: $num</p>";

/* This one will be '<p>Number: $num</p>' */
$num = 9;
$str = '<p>Number: $num</p>';

/* Get the first character of a string */
$str = 'This is a test.';
$first = $str[0];

/* Get the last character of a string. */
$str = 'This is still a test.';
$last = $str[strlen($str)-1];
```

?>

Un string en PHP se puede evaluar y sus reglas para estos estan regidas por las mismas que stdtod(3).

2.1.2 Arrays

Los arreglos se pueden crear con list(), array() o como en los siguientes ejemplos:

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

$a[] = "hello"; // $a[2] == "hello"

$a[] = "world"; // $a[3] == "world"

Los arreglos se pueden ordenar usando: asort(), arsort(), ksort(), rsort(), sort(), uasort(), usort(), y uksort() dependiendo en el orden que se desee. Se pueden contar los elementos de un arreglo con count(). Se puede recorrer un arreglo usando next(), prev() o each().

Ejemplo de arreglos multidimensionales:

$a["foo"][2] = $f; # (you can mix numeric and associative indices)

En PHP3 no es posible referenciar arreglos multidimensionales dentro de los strings. Por ejemplo lo siguiente no funciona:

echo "This won't work: $a[3][bar]";

Esto se arregla con concatenacion:

echo "This will work: " . $a[3][bar];

En PHP4 sin embargo se puede usar lo siguiente:

echo "This will work: {$a[3][bar]}";

Ejemplo de uso de **array**(). Las dos siguientes formas llevan al mismo array:

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

```
3 => 4 );
```
Para crear array multidimensionales array() se puede usar como sigue:
```
$a = array(
    "apple" => array(
        "color" => "red",
        "taste" => "sweet",
        "shape" => "round"
    ),
    "orange" => array(
        "color" => "orange",
        "taste" => "tart",
        "shape" => "round"
    ),
    "banana" => array(
        "color" => "yellow",
        "taste" => "paste-y",
        "shape" => "banana-shaped"
    )
);

 echo $a["apple"]["taste"];   # will output "sweet"
```

2.2 Variables

En PHP una variable se representa por un signo peso seguido por el nombre de la variable.

En PHP3 las variables solo podian ser asignadas por valor. En PHP4 tambien es posible la asignacion por referencia (lo que permite asignaciones mas rapidas). La asignacion por referencia es como en C, ejemplo: $a = &$b.

PHP crea variables con el tipo de dato mas apropiado de acuerdo al valor que se le asigne en un momento dado. El tipo de dato para una variable en PHP puede cambiar durante el tiempo de ejecucion. Para saber el tipo de una variable en un momento determinado se pueden usar las funciones: gettype(), is_long(), is_double(), is_string(), is_array(), e is_object().

2.2.1 Variables predefinidas

Dado que PHP se puede ejecutar como modulo de un web server debe ser posible el acceso a algunas variables que maneja el web server. Estas variables predefinidas no son accesibles cuando se ejecuta php como script externo al web server. Se puede usar phpinfo() para obtener una lista de las variables predefinidas. A continuacion una lista de las mas importantes variables predefinidas para Apache web server:

SERVER_NAME: el nombre del servidor.

REQUEST_METHOD: GET, HEAD, POST o PUT.

QUERY_STRING: El string con los datos de un formulario.

HTTP_REFERER: La direccion de la pagina (si existe) desde la cual se acceso la actual[1].

HTTP_USER_AGENT: Contenido del campo User_Agent: del header del requerimiento.

REMOTE_ADDR: La direccion IP desde la cual se acceso la pagina actual.

REMOTE_PORT: El puerto que esta siendo usado por el cliente para accesar la pagina actual.

etc.

Tambien PHP crea automatica las variables argc y argv para los mismos propositos que en C. PHP ademas crea otras variables como el array GLOBALS entre otras.

2.2.2 Alcance de variables

El alcance de las variables es como en la mayoria de los lenguajes script. Si algo se declara en el nivel mayor del script entonces se supone global. En el siguiente ejemplo, dentro del script b.inc tambien existira $a a menos que este script declare $a tambien:

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

[1]Esta variable es modificada por el browser y no todos los browsers lo soportan

En el siguiente ejemplo se muestra este caso:

```
$a = 1; /* global scope */
Function Test () {
echo $a; /* reference to local scope variable */
}
Test ();
```

Sin embargo dentro de una funcion se puede usar la palabra global para indicar que las variables son las globales. En el siguiente script se imprimiria 3 al final:

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

Lo anterior tambien es posible asi:

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

Tambien es posible el uso de static (como en C) para que las variables locales de una funcion mantengan sus valores entre llamadas y llamadas a tal funcion.

2.2.3 Variables variables

Esto se entendera mejor con la ayuda de un ejemplo:

```
$a = “hello”;
$$a = “world”;
```

Despues de estos dos estamentos existen dos variables: $a y $hello. Asi las siguientes dos lineas imprimen el mismo texto:

```
echo “$a ${$a}”;
echo “$a $hello”;
```

2.2.4 Variables externas en PHP

Cuando un formulario se envia a un script PHP entonces cada nombre de variable dentro del formulario pasa a ser una variable dentro del script PHP. Por ejemplo:

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

Cuando se ejecute el codigo foo.php3, este tendra el contenido de lo ingresado en esa entrada en una variable llamada $name.

PHP tambien soporta arreglos en los nombres de un formulario. Por ejemplo:

```
<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 alguna variable externa a PHP contiene un '.' entonces este sera reemplazado por un '_'.

Todas las variables del ambiente sobre el que se ejecute un script PHP seran accesibles al script como variables con el mismo nombre de la variable de ambiente.

Para trabajar con cookies PHP ofrece la funcion SetCookie()

2.2.5 Constantes

Una constante es casi lo mismo que una variable con la salvedad de que una constante se define con define() y su valor no puede ser modificado despues. Algunas constantes predefinidas son:

__FILE__ : El nombre del script que se esta interpretando. Si se incluye otro script entonces es el nombre de este ultimo el que contiene esta constante.

__LINE__ : La linea actual del script __FILE__

PHP_VERSION : version del PHP

PHP_OS : El sistema operativo sobre el cual se esta interpretando el script

TRUE : Un valor siempre verdadero

FALSE : Un valor siempre falso

Ejemplo definiendo una constante:

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

2.3 Expresiones

Casi todo lo que se escribe en un programa en un lenguaje de programacion es una expresion. Todo lo que tiene un valor es expresion. Una asignacion, una comparacion, una funcion, etc son ejemplos de expresiones.

2.3.1 Operadores

Operadores Aritmeticos:

Ejemplo	Nombre	Resultado
$a + $b	Adicion	Suma de $a y $b
$a - $b	Substraccion	Diferencia entre $a y $b
$a * $b	Multiplicacion	Producto de $a y $b
$a / $b	Division	Division de $a y $b
$a % $b	Modulo	Resto de $a dividido por $b

Operadores de asignacion: La asignacion es con '='. El valor de una asignacion es el valor asignado. Por ejemplo: $a = ($b = 4) + 5; $a termina valiendo 9 al final de esta asignacion. Tambien existen las combinaciones de operadores como:

$b = "Hola "; $b .= "Mundo!"; ($b termina con el valor "Hola Mundo!")

Dentro de las asignaciones tambien en PHP4 esta la asignacion de un valor por referencia, esto es $a = &$b.. en donde $a queda apuntando a la direccion de

memoria de $b.

Operadores a nivel de bits:

Ejemplo	Nombre	Resultado
$a & $b	y (and)	Bits activos tanto en $a como en $b se activan
$a | $b	o (or)	Bits activos en $a o en $b se activan
$a ^ $b	o exclusivo (Xor)	Bits activos en $a o $b pero no en ambos se activan
~ $a	no (not)	Bits no activos en $a se hacen activos y viceversa
$a << $b	corrimiento izq (shift left)	Se corren $b veces los bits de $a a la izq (multiplicar por 2)
$a >> $b	corrimiento der (shit right)	Se corren $b veces los bits de $a a la der (dividir por 2)

Operadores de comparacion:

Ejemplo	Nombre	Resultado
$a == $b	Igual	V si $a es igual a $b
$a === $b	Identico	V si $a == $b y ellos son del mismo tipo de dato (PHP4)
$a != $b	No igual	V si $a no es igual a $b
$a !== $b	No identico	V si $a != $b o ellos no son del mismo tipo de dato (PHP4)
$a < $b	menor que	V si $a es estrictamente menor que $b
$a > $b	mayor que	V si $a es estrictamente mayor que $b
$a <= $b	menor o igual que	V si $a es menor o igual a $b
$a >= $b	mayor o igual que	V si $a es mayor o igual a $b

Otro operador de comparacion es ?. (expr1) ? (expr2) : (expr3) . Se evalue expr2 si expr1 es verdadero y expr3 en el otro caso.

Operadores de reporte de error: El caracter @ se usa delante de expresiones para ignorar cualquier mensaje de error que pueda generar esa expresion. Si la caracteristica track_errors fue habilitada al configurar/instalar PHP entonces los mensajes de error generados por la expresion son grabados en la variable global $php_errormsg. Esta variable se sobre-escribe con cada error por lo que se debe leer su mensaje inmediatamente despues de interpretada la expresion en cuestion. Por ejemplo:

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

Operadores de ejecucion shell: PHP soporta la ejecucion de comandos shell con el comando entre comillas como ‘. Por ejemplo:

$output = ‘ls -l‘;

echo “<pre>$output</pre>”;

Funciones relacionadas: system(), passthru(), exec(), popen() y escapeshell-cmd().

Operadores de incremento/decremento:

Ejemplo	Nombre	Efecto
++$a	pre-incremento	Se incrementa $a en 1 y luego retorna $a
$a++	post-incremento	Se retorna $a y luego se incrementa $a en 1
–$a	pre-decremento	Se decrementa $a en 1 y luego se retorna $a
$a–	post-decremento	Se retorna $a y luego se decrementa $a en 1

Operadores logicos:

Ejemplo	Nombre	Resultado
$a and $b	And	V si $a y $b son V
$a or $b	Or	V si $a o $b son V
$a xor $b	Xor	V si $a o $b es V pero no ambos a la vez
! $a	Not	V si $a no es V
$a && $b	And	V si $a y $b son V
$a || $b	Or	V si $a o $b son V

and con && y or con || tienen distinta prioridad en la precedencia de los operadores. && y || tienen mayor prioridad que and y or en la evaluacion de expresiones.

Operadores de String: Existen dos operadores para strings. El primero es ’.’ que concatena dos strings y el otro es ’.=’ que concatena y asigna a la vez. Ejemplo:

$a = “Hola “;

$b = $a . “Mundo!”; // $b contiene “Hola Mundo!”

$a = “Hola “;

$a .= “Mundo!”; // $a contiene “Hola Mundo!” sin hacer uso de otra variable.

2.4 Estructuras de control

Estas se componen por if, while, for y switch. Cada una de estas se empiezan y terminan como en C con { y }. Sin embargo existe una sintaxis alternativa en PHP. Esta consiste en reemplazar el { por : y los } por endif, endwhile, endfor y endswitch respectivamente. Otra estructura de control pero que no posee la sintaxis alternativa es do-while.

2.4.1 if-else-elseif

Ejemplo:

```
if ($a > $b) {
  print "a is bigger than b";
} elseif ($a == $b) {
  print "a is equal to b";
} else {
  print "a is smaller than b";
}
```

Ejemplo con la sintaxis alternativa:

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

2.4.2 while

Como los loops en C, while se puede terminar dentro del loop con break.
Ejemplo:

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

2.4.3 do-while

Como los loops en C, do-while se puede terminar dentro del loop con break.
Ejemplo:

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

2.4.4 for

Como los loops en C, for se puede terminar dentro del loop con break.
Ejemplo:

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

2.4.5 foreach

Esta funcion solo existe en PHP4. En PHP se debe usar while con list() y each() para simularlo.

foreach sirve para recorrer arreglos. Cada vez que se llama foreach implicitamente hace un reset() sobre el arreglo y por tanto comienza del principio. foreach trabaja con una copia del arreglo por lo que el puntero al arreglo no es modificado como con each().

Los dos siguientes codigos tienen el mismo efecto:

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

```
}
```
_____

Los dos siguientes tambien:

_____

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

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

Otros ejemplos:

```
/* solo valor */
$a = array (1, 2, 3, 17);
foreach ($a as $v) {
  print "Current value of \$a: $v.\n";
}
/* ahora se imprime la clave */
$a = array (1, 2, 3, 17);
$i = 0; /* for illustrative purposes only */
foreach($a as $v) {
  print "\$a[$i] => $k.\n";
}
/* ahora clave y valor se imprimen */
$a = array (
  "one" => 1,
  "two" => 2,
  "three" => 3,
  "seventeen" => 17
);
foreach($a as $k => $v) {
  print "\$a[$k] => $v.\n";
}
```

2.4.6 switch

Ejemplo:

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

2.4.7 break

Como ya se ha dicho, break termina la ejecucion de un loop for, while, do-while o switch. En PHP break acepta un argumento numerico opcional que indica cuantos loops deben ser terminados. Esto evita el uso de goto o longjmp de C. Ejemplo:

```
$i = 0;
while (++$i) {
  switch ($i) {
  case 5:
    echo "En 5<br>\n";
    break 1; /* Sale solo del switch. */
  case 10:
    echo "En 10; Saliendo<br>\n";
    break 2; /* Sale del switch y del while. */
  default: break;
  }
}
```

2.4.8 continue

Como en C, continue es usado para saltarse el resto de la ejecucion de un loop e ir directamente a la siguiente iteracion. Como break continue tambien admite un argumento numerico opcional que indica los loops que deben saltarse.

2.4.9 require(), include()

require() e include() son muy parecidos a #include de C.

Funciones relacionadas: require_once(), include_once, get_required_files(), get_included_files(), readfile() y virtual().

2.5 Funciones

En PHP3 las funciones deben ser definidas antes de referenciadas. En PHP4 esto no es necesario. La sintaxis general de una funcion en PHP es:

```
function prueba ($arg_1, $arg_2, ..., $arg_n) {
  echo "Ejemplo de funcion.\n";
  return $valor_retornado;
}
```

El pasaje por valor es lo normal en PHP. Para pasar un valor por referencia se debe llamar a la funcion asi: prueba(&$arg1); . Para forzar que una funcion siempre se le pase un valor por referencia entonces se puede definir asi:

```
function prueba_ref( &$arg1, &$arg2, $arg3 ) {
...
}
```

Como en C++ se pueden definir valores por defecto. Ejemplo:

```
function makecoffee ($type = "cappucino") {
  return "Making a cup of $type.\n";
}
```

y luego se puede llamar a esta funcion con o sin argumentos como sigue:

```
echo makecoffee ();
echo makecoffee ("espresso");
```

En el caso de tener argumentos con y sin valores por omision, se deben poner los argumentos con valores por omision a la derecha de los primeros:

```
function makeyogurt( $type = "acidophilus", $flavour) /* Mal.. pues makeyo-
```

gurt(“algo”) generara un error al no poner un argumento obligatorio como es $flavour */

debe ser entonces:

```
function makeyogurt ($flavour, $type = "acidophilus")
```

De las funciones se puede retornar cualquier tipo de dato, un array, una suma de algo, una asignacion, un variable por referencia, etc.

PHP soporta variables funciones. Esto es, si una variable tiene parentesis () al final de ella, entonces PHP tratara de ejecutar una funcion con el nombre del valor de la variable en ese momento. Por ejemplo:

```
<?php
function foo() {
  echo "In foo()<br>\n";
}
function bar( $arg = ” ) {
  echo "In bar(); argument was ’$arg’.<br>\n";
}
$func = ’foo’;
$func();
$func = ’bar’;
$func( ’test’ );
?>
```

2.6 Clases y Objetos

Una clase es una coleccion de variables y funciones trabajando con esas variables. Un ejemplo de clase se muestra a continuacion:

```
<?php
class Cart {
  var $items; // Items en el carro de compras
  // Se agregan $num articulos de $artnr al carro
  function add_item ($artnr, $num) {
    $this->items[$artnr] += $num;
  }
  // Se sacan $num articulos de $artnr del carro
  function remove_item ($artnr, $num) {
    if ($this->items[$artnr] > $num) {
      $this->items[$artnr] -= $num;
      return true;
    } else {
      return false;
    }
  }
}
?>
```

Para crear un objeto de una clase se usa el operador new como sigue:

```
$cart = new Cart;
$cart->add_item(“10”, 1); // se agrega un item del articulo 10 al carro
```

La herencia se puede implementar como sigue:

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

Constructores son funciones en una clase que son automaticamente ejecutadas cuando se crea un objeto. Una funcion es un constructor cuando tiene el mismo nombre de la clase. En el siguiente ejemplo los argumentos para el constructor son opcionales como se vio anteriormente:

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

3 Uso de LDAP en PHP

PHP soporta Lightweight Directory Access Protocol (LDAP). LDAP es un protocolo para acceder a directorios que son un tipo de base de datos en donde la informacion es organizada de jerarquicamente y las operaciones sobre esta base de datos son principalmente busquedas y lecturas. A continuacion una lista de las funciones LDAP en PHP:

ldap_add - Add entries to LDAP directory
ldap_bind - Bind to LDAP directory
ldap_close - Close link to LDAP server
ldap_compare - Compare value of attribute found in entry specified with DN
ldap_connect - Connect to an LDAP server
ldap_count_entries - Count the number of entries in a search
ldap_delete - Delete an entry from a directory
ldap_dn2ufn - Convert DN to User Friendly Naming format
ldap_err2str - Convert LDAP error number into string error message

ldap_errno - Return the LDAP error number of the last LDAP command
ldap_error - Return the LDAP error message of the last LDAP command
ldap_explode_dn - Splits DN into its component parts
ldap_first_attribute - Return first attribute
ldap_first_entry - Return first result id
ldap_free_result - Free result memory
ldap_get_attributes - Get attributes from a search result entry
ldap_get_dn - Get the DN of a result entry
ldap_get_entries - Get all result entries
ldap_get_values - Get all values from a result entry
ldap_get_values_len - Get all binary values from a result entry
ldap_list - Single-level search
ldap_modify - Modify an LDAP entry
ldap_mod_add - Add attribute values to current attributes
ldap_mod_del - Delete attribute values from current attributes
ldap_mod_replace - Replace attribute values with new ones
ldap_next_attribute - Get the next attribute in result
ldap_next_entry - Get next result entry
ldap_read - Read an entry
ldap_search - Search LDAP tree
ldap_unbind - Unbind from LDAP directory

La secuencia tipica de una aplicacion que accesa un servidor LDAP es la siguiente:

ldap_connect() /* se establece la coneccion al servidor LDAP */
ldap_bind() /* autentificacion, ya sea anonymous o un “login” provisto */
/* efectuar una busqueda o actualizacion al directorio, desplegar resultados, etc */
ldap_close() /* desconeccion del servidor LDAP */

A continuacion un ejemplo de busqueda en el directorio de la entrada en el directorio cuyo uid sea sram, se despliega el campo email y el gecos:

```
<?php
 // Secuencia basica: connect, bind, buscar, interpretar resultado de busqueda,
 // mostrar algun resultado y cerrar coneccion con ldap_close().

 echo "<h3>Probando coneccion LDAP desde PHP</h3>";
 echo "Conectandose ...";
```

```
$ds=ldap_connect("localhost"); // debe ser un servidor LDAP
echo "Resultado de la coneccion: ".$ds."<p>";

if ($ds) {
  echo "Binding ...";
  $r=ldap_bind($ds); // anonymous bind, tipicamente acceso read-only
  echo "Resultado del bind(): ".$r."<p>";

  echo "Buscando (uid=sram) ...";
  $sr=ldap_search($ds,"dc=Udec,dc=CL", "uid=sram");
  echo "Resultado de busqueda: ".$sr."<p>";

  // Lo siguiente es util cuando la busqueda arroja multiples entradas
  echo "Numero de entradas retornadas: ".ldap_count_entries($ds,$sr)."<p>";

  echo "Obteniendo las entradas ...<p>";
  $info = ldap_get_entries($ds, $sr);
  echo "Valor para: ".$info["count"]." itemes retornados:<p>";

  for ($i=0; $i<$info["count"]; $i++) {
     echo "uid es: ". $info[$i]["uid"] ."<br>";
     echo "Entrada gecos: ". $info[$i]["gecos"][0] ."<br>";
     echo "Entrada email: ". $info[$i]["email"][0] ."<p>";
  }

  echo "Cerrando coneccion";
  ldap_close($ds);

} else {
  echo "<h4>No se puede conectar al servidor LDAP</h4>";
}
?>
```

Ahora un ejemplo para agregar una entrada:

```
<?php
$ds=ldap_connect("localhost"); // assuming the LDAP server is on this host
```

```
if ($ds) {
  // bind with appropriate dn to give update access
  $r=ldap_bind($ds,"cn=root, o=My Company, c=US", "secret");

  // prepare data
  $info["cn"]="John Jones";
  $info["sn"]="Jones";
  $info["mail"]="jonj@here.and.now";
  $info["objectclass"]="person";

  // add data to directory
  $r=ldap_add($ds, "cn=John Jones, o=My Company, c=US", $info);

  ldap_close($ds);
} else {
  echo "Unable to connect to LDAP server";
}
?>
```

4 Coneccion con Bases de Datos Oracle

Las funciones Oracle de PHP son las siguientes:

Ora_Bind (int cursor, string PHP variable name, string SQL parameter name, int length [, int type]) - bind a PHP variable to an Oracle parameter
Ora_Close (int cursor) - close an Oracle cursor
Ora_ColumnName (int cursor, int column) - get name of Oracle result column
Ora_ColumnSize (int cursor, int column) - get size of Oracle result column
Ora_ColumnType (int cursor, int column) - get type of Oracle result column[2]
Ora_Commit (int conn) - commit an Oracle transaction
Ora_CommitOff (int conn) - disable automatic commit
Ora_CommitOn (int conn) - enable automatic commit
Ora_Do (int conn, string query) - Parse, Exec, Fetch
Ora_Error (int cursor_or_connection) - get Oracle error message

[2]Valores retornados: VARCHAR2, VARCHAR, CHAR, NUMBER, LONG, LONG RAW, ROWID, DATE, CURSOR

Ora_ErrorCode **(int cursor_or_connection)** - get Oracle error code
Ora_Exec **(int cursor)** - execute parsed statement on an Oracle cursor
Ora_Fetch **(int cursor)** - fetch a row of data from a cursor
Ora_Fetch_Into **(int cursor, array result [, int flags])** - Fetch a row into the specified result array
Ora_GetColumn **(int cursor, mixed column)** - get data from a fetched column
Ora_Logoff **(int connection)** - close an Oracle connection
Ora_Logon **(string user, string pass)** - open an Oracle connection
Ora_pLogon **(string user, string pass)** - Open a persistent Oracle connection
Ora_Numcols **(int cursor_ind)** - Returns the number of columns
Ora_Numrows **(int cursor_ind)** - Returns the number of rows
Ora_Open **(int connection)** - open an Oracle cursor
Ora_Parse **(int cursor_ind, string sql_statement, int defer)** - parse an SQL statement
Ora_Rollback **(int connection)** - roll back transaction

Las variables de ambientes que se pueden configurar para conectarse a una base de datos ORACLE son: ORACLE_SID (instacia), ORACLE_HOME (oracle home), TNS_ADMIN (directorio que contiene el archivo tnsnames.ora).

A continuacion un ejemplo de codigo PHP para conectarse a una base de datos oracle:

```
<?
 putenv("ORACLE_SID=alumnos");
 putenv("ORACLE_HOME=/opt/oracle");

 $conn = ora_logon("scott@nombreTNS", "tiger");
 $curs = ora_open($conn);

 ora_commitoff($conn);

 $query = sprintf("select * from cat");

 ora_parse($curs, $query);
 ora_exec($curs);
 ora_fetch($curs);

 $ncols = ora_numcols($curs);
 $nrows = ora_numrows($curs);
```

```
printf(“Result is $ncols cols by $nrows rows”);

for ($i=0; $i<$ncols; $i++) {
  printf("col[%s] = %s type[%d] = %s”, $i, ora_columnname($curs, $i),
    $i, ora_columntype($curs, $i));
}
for ($j=0; $j<$nrows; $j++) {
  for ($i=0; $i<$ncols; $i++) {
    $col = ora_getcolumn($curs, $i);
    printf(&quotval[%d, %d] = %s * ", $j, $i, ora_getcolumn($curs, $i);
  }
  printf(“\n”);
}
ora_close($curs);
ora_logoff($conn);
?>
```