

Las operaciones de manipulación de ficheros tienen una gran importancia en prácticamente todos los lenguajes de programación, ya que, generalmente, no tiene sentido una aplicación que no sea capaz de mostrar resultados en pantalla, leer o escribir en un fichero o manipular una base de datos.

Las operaciones sobre ficheros suelen constar de tres fases:

- Fase 1: Apertura del fichero

En esta primera fase se abre el fichero, indicando si se realizarán operaciones para leer, escribir o añadir al final del mismo. La operación devuelve un descriptor de fichero.

- Fase 2: Procesamiento del fichero

Dentro de esta fase se distinguen dos posibilidades, que se hubiera abierto para lectura o para escritura.

- Fase 3: Cierre del fichero

Una vez realizadas las correspondientes operaciones sobre el fichero, se debe proceder a su cierre.

Veamos cada una de estas fases, indicando las funciones disponibles en PHP para realizar cada tarea.

FASE 1 : APERTURA DEL FICHERO

La operación de apertura del fichero se realiza con la función **fopen()**, cuya sintaxis es:

```
resource fopen( string fichero, string modo [, bool include_path = false ] )
```

La función devuelve un identificador de recurso o FALSE si la operación no pudo realizarse correctamente.

PARÁMETRO	DESCRIPCION
fichero	Nombre del fichero a abrir. En Windows, hay que escapar cualquier barra invertida usada en la ruta de fichero, o usar barras hacia delante. Ej. \$f1 = fopen("d:\\misapuntes\\php.txt", "r")
modo	Modo de apertura. r Solo Lectura (coloca puntero al fichero al principio del fichero) w Solo Escritura (coloca puntero al fichero al principio del fichero y si existe contenido lo elimina) a Añadir al final (solo Escritura, coloca puntero al fichero al final del fichero y si existe contenido lo mantiene) r+ , w+ y a+ actúan igual que r , w y a con la diferencia que abren para Lectura y Escritura
include_path	Con valor true indica que también se debe buscar el fichero en la lista de directorios de la directiva include_path (php.ini)

Si falla la apertura se genera un error de nivel E_WARNING. Se puede usar @ para suprimir su visualización

Ejemplos:

```
$fichero = @fopen("/datos/apuntes.txt", 'r');           // Abre para lectura silenciando errores
$fichero = fopen("http://www.misitio.com/fichero.gif", 'rb'); // Abre para lectura en binario
$fichero = fopen("datos.txt", 'a', true);              // Abre para añadir, ampliando búsqueda del fichero en el include_path
```

Realizada la operación de apertura, y antes de realizar cualquier tipo de proceso, se debe verificar que la operación **fopen()** ha tenido éxito, comprobando si el recurso devuelto se evalúa como verdadero (la operación se ha realizado correctamente) o como falso (se ha producido un error al intentar abrir el fichero).

Un ejemplo de esta comprobación podría ser:

```
$fichero = @fopen("/datos/datos.txt", 'r'); // Apertura para lectura silenciando errores
if ( !$fichero )                          // Si no se ha podido abrir el fichero
    die("Error al abrir fichero");          // finaliza la ejecución del script devolviendo un mensaje de error
.....                                     // En caso contrario el script continúa ejecutándose.
```

La función **die()** finaliza la ejecución del script mostrando el mensaje recibido como parámetro.

FASE 2 : PROCESAMIENTO DEL FICHERO

Dentro de esta segunda fase se distinguen dos posibilidades, que se hubiera abierto para lectura o para escritura. Las funciones que suelen emplearse más habitualmente son:

MODOS	FUNCIÓN UTILIZADA
r (fichero de texto)	<pre>string fgets(resource <i>fichero</i> [, int <i>longitud</i>])</pre> <p>Lee de <i>fichero</i> el número de caracteres indicado en el parámetro <i>longitud</i>, o hasta un carácter de fin de línea (que es incluido), o hasta fin de fichero (lo que antes ocurra)</p>
r (ficheros binarios)	<pre>string fread(resource <i>fichero</i>, int <i>longitud</i>)</pre> <p>Lee de <i>fichero</i> el número de bytes indicado en el parámetro <i>longitud</i> (o hasta fin de fichero) NOTA: en sistemas Windows el fichero debe abrirse en modo 'rb'</p>
w o a	<pre>int fwrite(resource <i>fichero</i>, string <i>cadena</i> [, int <i>longitud</i>])</pre> <p>Escribe en <i>fichero</i> los caracteres proporcionados en el parámetro <i>cadena</i>. Si se añade el parámetro <i>longitud</i>, escribe hasta que termine la cadena o hasta que se alcancen los caracteres indicados en este último parámetro, lo que antes ocurra. Devuelve el número de caracteres escrito</p>

Otras funciones útiles en el proceso de ficheros son

- ✓ **file_get_contents** string **file_get_contents** (string *filename* [, bool *include_path* =false])

Obtiene en una cadena el contenido de un fichero

```
// Obtiene y muestra código fuente de pag de inicio de un sitio web
$pag_ini = file_get_contents('http://www.example.com/');
echo $pag_ini;
```

- ✓ **file** : Obtiene en un array el contenido de un fichero. Cada elemento del array guarda una línea del fichero (con su \n)

array **file** (string *filename* [, int *flag* =0])

valores de flag: **FILE_USE_INCLUDE_PATH**
FILE_IGNORE_NEW_LINES
FILE_SKIP_EMPTY_LINES

```
// Obtiene y muestra código fuente de pag de inicio de un sitio web
$lineas = file ('http://www.example.com/');
foreach( $lineas as $n=>$dato)
    echo "línea $n : " . htmlspecialchars($dato) . "<br>";
```

- ✓ **file_put_contents** : Escribe una cadena o array en un fichero y devuelve nº de bytes escritos o FALSE en caso de error

int **file_put_contents** (string *filename* , mixed *dato* [, int *flag* =0])

valores de flag: **FILE_USE_INCLUDE_PATH**
FILE_APPEND

```
$modulo ='redes locales \n' ;
file_put_contents('materias.txt' , $modulo , FILE_APPEND);
```

- ✓ **feof** bool **feof** (resource *fichero*)

Comprueba si es final de fichero . Devuelve TRUE cuando puntero alcanza la marca de EOF y FALSE en otro caso

```
$f= fopen('elfichero.txt', 'r') ;
While ( !feof($f) )
    $r= fgets($f);
```

- ✓ **fsize** int **fsize** (string *filename*)

obtiene tamaño en bytes de un fichero

```
$f= fopen('elfichero.txt', 'r') ;
$contenido= fread( $f , filesize('elfichero.txt') );
```

FASE 3 : CIERRE DEL FICHERO

Para cerrar el fichero, simplemente se precisa la función **fclose()**, indicándole el recurso que apunta al fichero. Devuelve **true** si la operación tiene éxito, **false** en caso contrario. La sintaxis de la función es:

```
bool fclose(resource fichero)
```

EJEMPLOS DE UTILIZACIÓN

Vistas las fases más importantes y las funciones que habitualmente se emplean en cada una de ellas, mostraremos ejemplos de las operaciones más frecuentes sobre ficheros, como son la escritura de un fichero nuevo, la lectura de ficheros buscando alguna información, o añadir nuevos datos a un fichero ya existente.

1. LECTURA Y ESCRITURA DE FICHEROS

El ejemplo abre un fichero (remoto), lo lee y escribe en un fichero local su contenido.

```
<html> <head> <title> PHP: Lectura y Escritura de Ficheros </title> </head>
<body>
<?php
// Definimos el nombre del recurso remoto del que se va a leer y el nombre del fichero local donde se va a escribir
define("FICH_REMOTO", 'http://x.y.w.z/f_remoto.php'); // x.y.w.z es la IP del servidor remoto
define("FICH_LOCAL", 'fich_salida.txt');
// Abrimos los ficheros y comprobamos resultados
$fich_remoto = @fopen(FICH_REMOTO, 'r') or die("ERROR al abrir el recurso remoto");
$fich_local = @fopen(FICH_LOCAL, 'w') or die("ERROR al abrir el recurso local");
$contador_bytes = 0; // inicializamos a 0 el contador de bytes procesados
// Procesamos cada línea del recurso remoto hasta que alcance el final de fichero
while ($linea = fgets($fich_remoto))
{
    $nbytes = fwrite($fich_local, $linea); // escribimos la línea leída en el fichero local
    $contador_bytes = $contador_bytes + $nbytes; // Incrementamos el contador con el nº de bytes procesados
}
fclose($fich_remoto); // cerramos los ficheros
fclose($fich_local);
echo "Total de bytes procesados: $contador_bytes<br />"; // Mostramos el número de bytes leídos (y escritos)
?>
</body>
</html>
```

2. AÑADIR DATOS A UN FICHERO

Cuando el método de la solicitud es **GET** lee el fichero local **usuarios.txt**, muestra sus datos y un formulario que permita añadir nuevos usuarios. Cuando el método de solicitud es **POST** y se reciben datos, se encarga de añadir los datos al final del fichero **usuarios.txt**

fichero2.php

```
<html> <head> <title>PHP: Añadir datos a Ficheros</title></head>
<body>
<?php
// Definimos el nombre del fichero de usuarios La estructura de cada línea es:  USUARIO|PASSWORD
define("FICH_DATOS", 'usuarios.txt');
// Si el método es GET o falta el nombre de usuario -> mostramos datos
if (($_SERVER['REQUEST_METHOD'] == 'GET') || empty($_POST['user']))
{ $fich = @fopen(FICH_DATOS, 'r') or die("ERROR al abrir fichero de usuarios"); // abrimos el fichero para lectura
echo "<table align='center' border='1'>\n"; // Generamos la cabecera de la tabla
echo "<tr><th>usuario</th><th>password</th></tr>\n";
// Procesamos cada línea del fichero mostrando la información
while ($linea = fgets($fich))
{ $partes = explode('|', trim($linea)); // "Limpiamos" la línea y la troceamos obteniendo sus componentes
echo "<tr><td>$partes[0]</td><td>$partes[1]</td>"; // Escribimos la correspondiente fila de la tabla
}
fclose($fich); // Cerramos el fichero y la tabla
echo "</table><br>\n";
echo <<< Marca_Fin // Mostramos el formulario de introducción de datos
<form name='f1' action='fichero2.php' method='POST'>
Usuario: <input type='text' name='user' size='10' > &nbsp;
Palabra Clave: <input type='password' name='pclave' size='10' ><br>
<input type='submit' value='Enviar' >
</form>
Marca_Fin;
}
else // Si el metodo de envio es POST y se reciben datos, añadimos datos al final del fichero
{ $fich = @fopen(FICH_DATOS, 'a') or die("ERROR al abrir fichero"); // abrimos el fichero para añadir al final
$cadena = $_POST['user'] . '|' . $_POST['pclave'] . "\n"; // Creamos cadena a grabar -con su separador-
$fok = fwrite($fich, $cadena); // escribimos datos en el fichero
echo ($fok) ? "Datos añadidos al fichero" : "Error al añadir datos";
fclose($fich); // Cerramos el fichero
echo "<br>Pulsa <a href='fichero2.php'>aquí</a> para continuar..."; // Mostramos un enlace al propio script
}
?>
</body>
</html>
```

- La función **trim()** elimina caracteres *no útiles* de la línea recibida (espacios iniciales y finales, salto de línea,...)
- La función **explode()** trocea la línea en función del separador proporcionado, devolviendo los distintos trozos en un array.
- Para que el script funcione, previamente debe crearse el fichero *usuarios.txt*.

usuario	p. clave
usuario1	p_clave1
usuario2	p_clave2
usuario3	p_clave3

Usuario:
 Palabra Clave:

Datos añadidos al fichero
Pulsa **aquí** para continuar...

Presentaciones en pantalla al ejecutar el script anterior

DIRECTORIOS

La forma de acceder al contenido de un directorio es similar al procesamiento de ficheros secuenciales: se abre el directorio, después se procesa cada entrada del mismo, y finalmente se cierra el directorio.

Para realizar estas operaciones PHP proporciona las siguientes funciones:

OBJETIVO	FUNCIÓN
Determinar existencia	<code>bool is_dir(string <i>directorio</i>)</code> Determina si <i>directorio</i> existe
Apertura	<code>resource opendir(string <i>directorio</i>)</code> Abre <i>directorio</i> y devuelve un cursor para acceder a su contenido
Lectura	<code>string readdir(resource <i>dir_cursor</i>)</code> Devuelve la entrada apuntada por <i>dir_cursor</i> y avanza a la siguiente
Cierre	<code>void closedir(resource <i>dir_cursor</i>)</code> Cierra el recurso <i>dir_cursor</i>

Ejemplo: Script que muestre en pantalla el contenido del directorio especificado

```
<html> <head> <title>PHP: Listado de Directorios</title></head>
<body>
<?php
define('DIRECTORIO', '/PHP-5.0.4'); // Define el directorio que se va a procesar
if ( !is_dir(DIRECTORIO) ) // Comprueba que realmente existe el directorio
    die("No existe el directorio " . DIRECTORIO);
// Abrimos el directorio
$dir_cursor = @opendir(DIRECTORIO) or die("Error al abrir el directorio");
// Mostramos cada entrada del directorio
echo "<pre>\n";
$entrada = readdir($dir_cursor); // lee primera entrada
while ($entrada !== false ) // mientras haya datos
{
    echo " $entrada\n";
    $entrada = readdir($dir_cursor); // lee siguiente entrada
}
echo "</pre>\n";
closedir($dir_cursor); // cerramos el directorio
?>
</body>
</html>
```

OTRAS OPERACIONES SOBRE DIRECTORIOS Y FICHEROS

Crear directorio bool **mkdir** (string **directorio** [, int **modo** = 0777 [, bool **recursivo** = false]])

Intenta crear el directorio especificado devolviendo TRUE si tiene éxito y FALSE en caso de error

Directorio ruta y nombre de directorio a crear

Modo el predeterminado da el acceso mas amplio. En Windows se ignora

Recursivo Con valor true permite crear estructura de directorios anidados especificado en directorio

```
Ejemplo  <?php
          mkdir("dir1");    // lo crea en el directorio que contiene el script
          mkdir("../dir2"); // lo crea en el directorio padre del que contiene el script

          $estructura = './nivel1/nivel2/nivel3/'; // Estructura de directorios a crear

          // Para crear directorios anidados se ha de dar valor true al parámetro de recursividad

          if( !mkdir($estructura, 0777, true)) die('Fallo al crear las carpetas...');
          ?>
```

Eliminar directorio bool **rmdir**(string **directorio**)

Intenta eliminar el directorio indicado. Para ello el directorio debe existir , ha de estar vacio y se debe tener permisos para hacerlo. Devuelve TRUE o FALSE en función de si se tiene éxito o no en la acción.

```
Ejemplo  <?php
          if ( is_dir( 'ejemplos' ) ) rmdir('ejemplos'); // Comprobada existencia de directorio lo manda eliminar
          ?>
```

Eliminar ficheros bool **unlink** (string **fichero**)

El parámetro será la ruta y nombre del fichero a eliminar. Devuelve TRUE o FALSE en función de si se tiene éxito o no en la acción.

```
Ejemplo  <?php
          if ( is_file( './datos/ejemplos.txt' ) ) rmdir('./datos/ejemplos.txt'); // Comprobada existencia de fichero lo elimina
          ?>
```

Renombrar ficheros y directories bool **rename** (string **oldname** , string **newname**)

Intenta renombrar el elemento dado en oldname a newname. Si newname ya existía lo sobrescribe. Devuelve TRUE o FALSE en función de si se tiene éxito o no en la acción.

```
Ejemplo  <?php
          rename( "/tmp/fichero_tmp.txt" , "/home/user/login/docs/mi_fichero.txt" );
          ?>
```

Copiar ficheros bool **copy** (string **fuentes** , string **destino**)

Realiza una copia del fichero fuente en destino. Devuelve TRUE o FALSE en función de si se tiene éxito o no en la acción.

Para mover un fichero, usar la función rename().

```
Ejemplo  <?php
          copy( "/tmp/fichero_tmp.txt" , "/home/user/login/docs/mi_fichero.txt" );
          ?>
```