Utilización de ficheros externos

PHP dispone de funciones mediante las cuales se pueden **crear**, **modificar**, **borrar** y **leer** *ficheros* de cualquier tipo así como extraer información sobre ellos y sus contenidos.

Abrir o crear ficheros

Para crear o modificar ficheros se utiliza la instrucción:

$f1=fopen(*fichero,modo***)**

$f1 es una variable que recoge *el identificador del recurso*, un valor importante (será utilizado para referirnos a este fichero en instrucciones posteriores), **fichero** es el nombre (con extensión) del fichero a abrir o crear y deberá escribirse entre comillas, y **modo**, que es una cadena que debemos poner entre comillas, el indicador del *modo de apertura* elegido.

En la tabla de la derecha hemos enumerado las opciones de ese parámetro.

Si el fichero que pretendemos abrir está en un directorio distinto al del script, debe incluirse el *path* completo delante del nombre del fichero y la cadena resultante debe ir entre comillas.

¡Cuidado!

Si incluimos, junto con el nombre del fichero, **un path** hay que tener muy presente que bajo **Windows** hemos de utilizar siempre el separador *anti-slash* (\).

Cerrar ficheros

Una vez finalizado el uso de un fichero es necesario **cerrarlo**. Para ello PHP dispone de la siguiente intrucción:

fclose(*$f1***)**

Esta función - que devuelve un valor *booleano*- permite cerrar el fichero especificado en **$f1** que, como recordarás, es el valor del **identificador de recurso** que le fue asignado automaticamente por PHP en el momento de la apertura.

Punteros internos

PHP dispone de funciones para

Función fopen()

En esta tabla tienes los parámetros *modo* de la función **fopen** de apertura de ficheros.

Valores del parámetro *modo* de la función fopen	
Valor	*Funcionalidad*
r	Abre el fichero en modo **lectura** y coloca el **puntero** al comienzo del fichero
r+	Abre el fichero en modo **lectura** y **escritura** y coloca el **puntero** al comienzo del fichero
w	Abre el fichero en modo **escritura** y coloca el **puntero** al comienzo del fichero, reduce su tamaño a **cero** y si **el fichero no existe intenta crearlo**
w+	Abre el fichero en modo **lectura** y **escritura** y coloca el **puntero** al comienzo del fichero, reduce su tamaño a **cero** y si **el fichero no existe intenta crearlo**
a	Abre el fichero en modo **escritura** y coloca el **puntero** al **final** del fichero y si **no existe intenta crearlo**
a+	Abre el fichero en modo **lectura** y **escritura** y coloca el **puntero** al **final** del fichero y si **no existe intenta crearlo**

Un fichero de pruebas

Hemos creado un fichero llamado **domingo.txt** para poder utilizarlo en los ejemplos. Su contenido es exactamente el siguiente (incluidos los saltos de línea):

```
Esto es un ejemplo para comprobar
si funcionan o no los saltos de linea
en un documento de texto
que será leido desde php
```

Un ejemplo de algunas funciones sobre ficheros

```
<?
/* abrimos con w+ con lo cual borramos el contenido
  y creamos el fichero en el caso de que no existiera */
$f1=fopen("sabado.txt","w+");
# escribimos en el fichero vacio
fwrite($f1,"Esta es la primera linea que escribimos en el fichero<br>");
#cerramos el fichero
fclose($f1);
echo "<H2>Este es el resultado después del primer fwrite</H2><br>";
include("sabado.txt");
# abrimos con r+ con lo cual sobreescribiremos
# en el fichero preexistente
$f1=fopen("sabado.txt","r+");
# escribimos en al principio del fichero preexistente
# ya que al abrir un fichero en este modo el puntero
# se situa al comienzo del fichero
fputs($f1,"Esto se sobreescribe");
#cerramos el fichero
fclose($f1);
echo "<H2>Este es el resultado después del segundo fwrite</H2><br>";
include("sabado.txt");
# abrimos con a+ con lo cual AÑADIREMOS
# al fichero preexistente ya que el modo de apertura
# situa el puntero al final del fichero
$f1=fopen("sabado.txt","a+");
# escribimos  al final del fichero preexistente
fputs($f1," Esto se añadirá al final<br>");
#cerramos el fichero
fclose($f1);
echo "<H2>Este es el resultado después del tercer fwrite</H2><br>";
include("sabado.txt");
echo "<h2>Leyendo con fgetc</h2><br>";
# abrimos con r+ con lo cual podemos LEER y AÑADIR
# al fichero preexistente
```

situar sus *punteros internos* y también para determinar la posición a la que **apuntan** en un momento determinado.

Se trata de las siguientes:

feof(*$f1*)

Es un operador *booleano* que devuelve CIERTO (1) si el puntero señala el final del fichero y FALSO si no lo hace.

rewind(*$f1*)

Coloca el *puntero interno* al **comienzo** del fichero indicado por *el identificador del recurso* **$f1**.

fseek(*$f1, posición*)

Sitúa el apuntador del fichero señalado por *el identificador del recurso* **$f1** en la posición (expresada en bytes) señalada por **posición**.

ftell(*$f1*)

Devuelve (expresada en bytes) la **posición actual** del puntero interno del fichero.

¡Cuidado!

Antes de utilizar funciones es necesario que el fichero que señala el *identificador de recursos* haya sido **abierto**.

Lectura de ficheros

La lectura de los contenidos de un fichero puede hacerse de dos maneras: *sin apertura previa* o *con apertura previa* del mismo.

Lectura de ficheros sin apertura previa

Las funciones que permiten la **lectura** de ficheros *sin haber sido abiertos* previamente son las siguientes:

readfile(*fichero*)

Escribe directamente en el punto de inserción del *script* el *contenido completo* del fichero.

¡Atención!

La función *readfile* escribe el contenido del fichero *sin necesidad* de ir precedido por *echo* ni *print*.

Si se pone **echo** o se recoge en una variable, *además de su contenido* añadirá un **número** que indica el **tamaño del fichero** expresado en *bytes*.

$var=file(*fichero*)

Crea **$var** –un *array escalar*– cuyos elementos tienen como valores los

```
# leemos el primer caracter del fichero
# ya que el apuntador esta el principio
$z=fgetc($f1);
# imprimimos el primer carácter
echo "He leido el primer carácter: ",$z,"<br>";
/* leemos el segundo caracter del fichero
ya que el apuntador se ha movido a esa posición al leer
anteriormente el primer carácter.
OBSERVA que NO HEMOS CERRADO AUN EL FICHERO */
$z=fgetc($f1);
# este es el nuevo valor de la variable $z
echo "He leido el segundo carácter: ",$z,"<br>";
/* leemos el siguiente caracter del fichero
ya que el apuntador se ha movido a una nueva posición
Recuerda que NO HEMOS CERRADO AUN EL FICHERO */
$z=fgetc($f1);
# este es ahora el valor de la variable $z
echo "He leido el tercer carácter: ",$z,"<br>"
echo "<h2>Ahora el puntero está en el tercer caracter<br>";
echo "fgets empezará a leer a partir de el</H2>";;
$z=fgets($f1,200);
echo "Con fgets he leido esto: ",$z,"<br>";
#Ahora cerramos el fichero
fclose($f1);
echo "<br><H2>Al abrir el fichero de nuevo fgets
 comienza desde el principio</h2><br>";
#Abrimos de nuevo el fichero
$f1=fopen("sabado.txt","r");
#Leemos su contenido
$za=fgets($f1,5000);
#Presentamos el contenido
echo $za;
#Ahora cerramos el fichero
fclose($f1);
echo "-----------------------------------------------------------------<br>";
?>
<h2>Aqui veremos el contenido (sin etiquetas HTML)de una pagina web</H2>
<?
# Escribimos la dirección completa de la página que puede ser
# el resultado de unir el valor de la variable $_SERVER['DOCUMENT_ROOT']
#  (ruta completa deldirectorio raiz de servidor)
#   con el nombre del directorio que la contiene y nombre del fichero
# la abrimos en modo solo lectura
$f1=fopen($_SERVER['DOCUMENT_ROOT']."/cursophp/php53.php","r");
# Escribimos un bucle para que vaya leyendo
# cada una de las líneas hasta llegar al final del fichero
while (!feof($f1)) {
    $z = fgetss($f1, 1024);
    echo $z,"<br>";
  }
#Cerramos el fichero
fclose($f1);
#Borramos el fichero antes de salir
unlink("sabado.txt");
?>
```

Ver ejemplo83.php

Otro ejemplo completando el anterior

```
<?
# Abrimos el fichero en modo lectura
$f1=fopen("domingo.txt","r");
# Al anteponer echo a fpassthru
# NOS APARECERA AL FINAL EL TAMAÑO DEL FICHERO
echo fpassthru($f1),"<br>";
/* Abrimos de nuevo el fichero
RECUERDA QUE FPASSHRU LO CIERRA AUTOMATICAMENTE
DESPUES DE EJECUTAR LA INSTRUCCIÓN */
$f1=fopen("domingo.txt","r");
# Este bucle nos escribirá cada una de las
#lineas del fichero
while(!feof($f1)){
$z=fgets($f1,4000);
echo $z,"<br>";
```

contenidos de cada una de las **líneas del fichero**.

Una línea termina allí donde se haya insertado **un salto de línea** en el fichero original.

Lectura de ficheros con apertura previa

Para la utilización de estas funciones los ficheros **han de ser abiertos** en un **modo** que permita la **lectura**.

La función que permite la lectura completa del fichero es:

fpassthru(*$f1*)

Esta función presenta algunas peculiaridades importantes:

• **Cierra** el fichero de forma automática después de la lectura. Por esa razón, si se escribe la función **fclose** a continuación de *fpassthru*, se produce un error.

• Si el resultado se recoge en una variable, o si va precedido de **echo**, además de escribir el contenido del mismo, añadirá el *número de bytes* que indican su **tamaño**.

fgets(*$f1,long*)

Extrae del fichero señalado por el **$f1** una cadena –que **comienza** en la *posición actual* del puntero– y cuya **longitud** está limitada por el **menor** de estos tres valores:

• El valor (en bytes) indicado en **long**.

• La *distancia* (también en bytes) desde la *posición actual del puntero* hasta el **final** del fichero.

• La *distancia* que hay entre la *posición actual* del puntero y el **primer** *salto de línea*.

fgetc(*$f1*)

Extrae el caracter **siguiente** al señalado por la *posición actual del puntero*.

Escribir en un fichero

Una vez abierto un fichero -en modo que permita escritura- la función PHP que nos permite *escribir* en el es la siguiente:

fwrite(*$f1,"texto",long*)

donde: **$f1** sigue siendo el *identificador de recurso*, **texto** la cadena de texto a insertar en el fichero y **long** el número máximo de caracteres que han de insertarse.

Si la cadena de texto tiene *menor o igual longitud* que el parámetro **long** la escribirá en su totalidad, en caso

```
}
# Situamos el puntero
#al comienzo del fichero
rewind($f1);
# Reescribimos el fichero
while(!feof($f1)){
$z=fgets($f1,4000);
echo $z,"<br>";
}
# Situamos de nuevo el puntero
#al comienzo del fichero
rewind($f1);
# Situamos el puntero
#señalando el byte número 15 del fichero
fseek($f1,15);
# Releemos el fichero
#ahora la primera linea estar incompleta
#LE FALTARAN LOS 15 PRIMEROS CARACTERES
while(!feof($f1)){
$z=fgets($f1,4000);
echo $z,"<br>";
}
# volvemos el puntero al comienzo del fichero
rewind($f1);
#leemos la primera linea
$z=fgets($f1,4000);
echo $z,"<br>";
# Determinamos LA POSICION ACTUAL DEL PUNTERO
echo ftell($f1),"<br>";
# Cerramos el fichero
fclose($f1);
echo "_____<br>";
# leemos el fichero y lo presentamos
# en diferentes modalidades
$pepe=readfile("domingo.txt");
 readfile("domingo.txt");
echo $pepe, "<br>";
#leemos el fichero y lo recogemos
#en un array
$z=file("domingo.txt");
#Al presentar la variable solo
#nos aparecerá la palabra array
echo $z,"<br>";
#  presentamos el contenido del array
foreach($z as $linea=>$texto) {
echo "Linea: ",$linea,"  Texto: ",$texto,"<br>";
};
# copiamos el fichero con mensaje de resultado
if (!copy("domingo.txt", "otrodomingo.txt")) {
    print("Error en el proceso de copia<br>\n");
}else{
 print "<br>Fichero copiado con exito";
}
# renombramos un fichero con mensaje de resultado
if (!rename("otrodomingo.txt", "otrolunes.txt")) {
    print("Error en el proceso de renombrado<br>");
}else{
 print "<br>Fichero renombrado con exito";
}
unlink("otrolunes.txt");
echo "Ultima modificación a las: ",date("h:i:s A",
filemtime ("domingo.txt")),"  del día  ",
date("j-n-Y", filemtime ("domingo.txt"));
echo "<br>El tamaño del fichero es: ", filesize("domingo.txt"),"
bytes<br>";
echo "<br>El  fichero es tipo: ", filetype("domingo.txt")," <br>";
echo "<br>Saldrá un 1 si el fichero existe: ",file_exists("domingo.txt");
?>
```

Ver ejemplo84.php

Los valores del array devuelto por la función stat

Indice	Significado	Sintaxis	Resultado
0	Dispositivo	**<? echo $d[0] ?>**	**2**

contrario sólo escribirá el número de caracteres indicados.

También es posible utilizar:

fputs(*$f1,"texto",long*)

que en realidad es un *alias* de la función anterior.

¡Atención!

Estas funciones realizan la **inserción** de la cadena **a partir** de la *posición a la que apunte el puntero* en el momento de ser invocadas.

Si el fichero ya existiera y contuviera datos los nuevos datos **se sobrescribirían** sobre el contenido anterior.

Para poder **añadir** contenidos a un fichero el *puntero* deberá *apuntar* el **final** del fichero preexistente y estar *abierto* en un **modo** que permita añadir contenidos.

Borrado de ficheros

Para borrar ficheros se utiliza la siguiente instrucción:

unlink(*fichero*)

fichero ha de ser una cadena que contenga el nombre y la extensión del fichero y, en su caso, también el path.

Duplicado de ficheros

La función:

copy(*fich1, fich2*)

Copia el fichero **fich1** (debe indicarse nombre y extensión) en otro fichero cuyo nombre y extensión se establecen en la cadena **fich2**.

Esta función devuelve un valor *booleano* indicando si la copia se ha realizado con éxito TRUE (1) o FALSE (nul) si por alguna razón no ha podido realizarse la copia.

Renombrar ficheros

La función:

rename(*fich1, fich2*)

cambia el nombre del fichero **fich1** (hay que poner nombre y extensión) por el indicado en la cadena **fich2**.

También devuelve TRUE o FALSE.

Si tratamos de cambiar el nombre a un fichero inexistente nos dará error.

Funciones informativas

1	I node	<? echo $d[1] ?>	0
2	Modo de protección de I node	<? echo $d[2] ?>	33206
3	Número de enlaces	<? echo $d[3] ?>	1
4	Id de usuario del propietario	<? echo $d[4] ?>	0
5	Id de grupo del propietario	<? echo $d[5] ?>	0
6	tipo de dispositivo si es un inode device *	<? echo $d[6] ?>	2
7	Tamaño en bytes	<? echo $d[7] ?>	126
8	Fecha del último acceso	<? echo $d[8] ?>	1243021087
9	Fecha de la última modificación	<? echo $d[9] ?>	1243002280
10	Fecha del último cambio	<? echo $d[10] ?>	1243021087
11	Tamaño del bloque para el sistema I/O *	<? echo $d[11] ?>	-1
12	Número de bloques ocupados *	<? echo $d[12] ?>	-1
Los valores señalados con * devuelven -1 en algunos sistemas operativos, entre ellos Windows>			

Ejemplo de un contador de visitas

```
<?
/* comprobamos si existe el fichero contador. Si existe
   leemos las visitas registradas e incrementamos su valor en una unidad
   Si no existe, registramos 1 como valor de número de visitas*/
if(file_exists("contador.txt")){
/* abrimos el fichero en modo lectura y escritura (r+) con lo que
   el puntero se colocará al comienzo del fichero */
       $f1=fopen("contador.txt","r+");
        # leemos el contenido del fichero
       $visitas=(int)(fgets($f1,10));
        # lo aumentamos en una unidad
          $visitas++;
        # colocamos el puntero al comienzo del fichero para que
        # al guardar en nuevo valor sobreescriba el anterior
            rewind($f1);
   }else{
/*abrimos el fichero en modo lectura y escritura con (w+)
  de modo que se cree automaticamente al no existir*/
       $f1=fopen("contador.txt","w+");
#asignamos uno como valor a número de visitas
       $visitas=1;
}
/* escribimos el número de visitas en el fichero. En cualquiera
   de los casos el puntero estará al comienzo del fichero, por tanto
   cuando existan valores serán sobreescritos */
 fwrite($f1,$visitas,10);
 print("Esta página ha sido visitada ".$visitas." veces");
 fclose($f1);
?>
```

Ver contador

Guardar y leer datos transferidos mediante un formulario

Aunque el modo más habitual de guardar información suele ser los servidores de bases de datos (MySQL, por ejemplo) la utilización de ficheros ofrece interesantes posibilidades de almacenamiento de información.

Este es un ejemplo muy sencillo, en el que mediante un formulario tal como el que aparece en el recuadro puede transferirse y almacenarse la información en un fichero.

```
<form name="fichero" method="post" action="escribe.php">
<input type="text" name="nombre">
<input type="text" name="apellido">
<input type="edad" name="edad">
<input type="submit" value="enviar">
</form>
```

PHP dispone de funciones que nos facilitan información sobre ficheros. Algunas de ellas son las siguientes:

file_exists(*fichero*)

Esta función devuelve TRUE si el fichero existe, en caso contrario devuelve FALSE.

filesize(*fichero*)

Devuelve el *tamaño* del fichero expresándolo en *bytes*. En caso de que el fichero no existiera nos dará un error.

filetype(*fichero*)

Devuelve una *cadena* en la que se indica el *tipo* del *fichero*. En caso de que el fichero no existiera nos dará un error.

filemtime(*fichero*)

Devuelve –en *tiempo Unix*– la *fecha de la última modificación* del fichero.

stat(*fichero*)

Devuelve un *array* que contiene información sobre el fichero.

En la tabla de la derecha puedes ver los contenidos asociados a cada uno de sus índices.

Recogeremos en un array, que llamaremos **$d**, el resultado de la función **stat** aplicada sobre el fichero **domingo.txt**.

Para ello vamos a utilizar la siguiente sintaxis:

$d=stat("domingo.txt")

El contenido y significado de los valores asociados a los índices de array **$d** son los que tenemos en la tabla de la derecha.

Otras funciones

Existen otras muchas funciones relacionadas con el manejo de ficheros y directorios, así como con los permisos de acceso, etcétera.

Hemos resumido únicamente las de mayor interés.

Si quieres profundizar en este tema a través de este enlace podrás acceder al capítulo del *Manual de Referencia oficial de PHP*, en el que se describen las funciones relacionadas con el manejo de ficheros.

Los datos transferidos mediante un formulario como el anterior podrían ser registrados y visualizados mediante un script como este:

```
<?
/*abrimos el fichero en modo a+ para permitir que
  se cree en caso de no existir, que permita los modos lectura
  y escritura y que escriba al final del fichero */
$f1=fopen("escribiente.txt","a+");
    # hacemos un bucle para leer los valores transferidos
        # desde el formulario y recogidos en el array $_POST
       foreach($_POST as $v){
        /* añadimos "\r\n" a cada valor para que se inserte
           un salto de línea y que cada valor sea recogido en
           una línea distinta en el fichero
           Limitamos las entradas a 150 caracteres*/
            fwrite($f1,$v."\r\n",150);
      }
/* para comprobar que los nuevos datos han sido agregados
   y visualizar el contenido integro del fichero situamos el
   puntero interno al comienzo del mismo */
rewind($f1);
/* creamos un bucle que vaya leyendo todas las líneas
   hasta encontrar el final del fichero */
   while (!feof($f1)) {
        /* vamos leyendo el contenido de cada línea del fichero
        y aunque establecemos en 250 el número de carácteres
        dado que los saltos del línea aparecerán antes
        serán ellos los puntos de interrupción de cada lectura*/
                $z = fgets($f1,250);
        #presentamos el resultado de las lecturas
                echo $z,"<br>";
     }
  # cerramos el fichero
 fclose($f1);
?>
```

Ejercicio nº 28

Crea un script -puedes llamarlo **ejercicio28.php**- de modo que al ejecutarlo escriba en un fichero la fecha y hora en que se produjo el acceso.

Ese mismo script deberá presentar en la página la fecha y hora de la visita anterior y si es la primera vez que se accede deberá aparecer un mensaje diciendo: «Esta es la primera vez que accedes a esta página».

Para obtener los valores de fecha y hora, deberás utilizar las funciones **adodb** a las que se alude en la página en la que se trata la opción **Include**.