# TEMA 9

# Ficheros

Dentro de lo que es el tema de ficheros en PHP podemos destacar dos grandes ámbitos:

- File System, o sistema de ficheros
- Upload

El primero consiste en como crear, modificar y borrar ficheros y carpetas dentro del servidor en tiempo de ejecución de la aplicación web.

El segundo trata sobre como podemos realizar la subida de archivos (de texto, imagen o cualquier otro) a partir de un formulario hacia el servidor por parte del cliente.

# Índice

# Permisos de Archivos

Desde PHP vamos a interactuar con el sistema de archivos del servidor, por lo que debemos saber que tipo de servidor tenemos y sus características.

Como normalmente los servidores están montados sobre un S.O. Linux, debemos recordar cómo se manejan los archivos desde este S.O.

Cada fichero y carpeta dispone de unas propiedades que determinan quien y que acciones se pueden realizar sobre el mismo. Nos estamos refiriendo a acciones de tipo CRUD.

Los permisos están configurados según la conversión de binario a decimal, por lo que, p. e., 111 es 7.

# Permisos de Archivos

Recordad que un archivo tiene siempre permisos de esta forma:

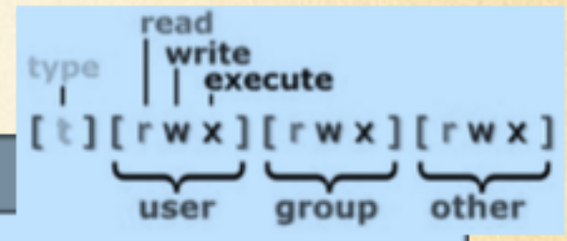
**[t] type [r-w-x] user [r-w-x] group [r-w-x] other**

Entonces, si un fichero o carpeta tiene permisos 777 significará que tiene para el usuario, grupo y otros todos los permisos de r (read) w (write) y x (execution). Se cambian mediante **chmod**

Por seguridad **evitaremos en todo momento poner a algún archivo o carpeta los permisos** 777, porque eso es tener una puerta abierta a cualquiera que quiera entrar a nuestro sistema.

# Permisos de Archivos

## "Chuleta" para chmod




| Permisos | Valor | Descripción |
|---|---|---|
| rw------- | 600 | El propietario tiene permisos de lectura y escritura. |
| rwx--x--x | 711 | El propietario lectura, escritura y ejecución, el grupo y otros solo ejecución. |
| rwxr-xr-x | 755 | El propietario lectura, escritura y ejecución, el grupo y otros pueden leer y ejecutar el archivo. |
| rwxrwxrwx | 777 | El archivo puede ser leido, escrito y ejecutado por quien sea. |
| r-------- | 400 | Solo el propietario puede leer el archivo, pero ni el mismo puede modificarlo o ejecutarlo y por supuesto ni elgrupo ni otros pueden hacer nada en el. |
| rw-r----- | 640 | El usuario propietario puede leer y escribir, el grupo puede leer el archivo y otros no pueden hacer nada. |

# Procesos CRUD

- `fopen()`. Abre un puntero a un archivo.
- `fclose()`. Cierra el puntero a un archivo.
- `fgets()`. Obtiene una línea desde el puntero a un archivo.
- `fread()`. Lectura de un fichero en modo binario seguro.
- `fwrite()`. Escritura de un archivo en modo binario seguro.

# Procesos

- flock(). Bloque de un archivo (_SH, _EX, _UN y _NB).
- file(). Transfiere un fichero completo a un array.
- file_exist(). Comprueba si existe un fichero.
- is_readable(). Indica si existe y es legible.
- stream_set_timeout(). Establece un periodo de tiempo de espera en un flujo.
- …

http://www.php.net/manual/es/book.filesystem.php

# Escribir un archivo en 3 pasos

```
<?php
 //Abrimos un fichero, escribimos en él y lo cerramos
 $fuente = fopen("lista.txt", "a+");
 fwrite($fuente, "Hola Mundo\n");
 fclose($fuente);
?>
```

Como podemos ver, los pasos son los indicados en el comentario. Ver ejemplo 1.

fopen tiene como segundo parámetro los modos de apertura del archivo, que son los siguientes:

| | |
|---|---|
| r | Apertura para sólo lectura, coloca el puntero al principio. |
| r+ | Apertura para lectura y escritura, coloca el puntero al principio. |
| w | Apertura para escritura, coloca el puntero al principio y trunca el fichero a longitud 0. Si el fichero no existe lo intenta crear. |
| w+ | Apertura para lectura y escritura, coloca el puntero al principio y trunca el fichero a longitud 0. Si el fichero no existe lo crea. |
| a | Apertura para sólo escritura, coloca el puntero al final. Si no existe lo crea. |
| a+ | Apertura para lectura y escritura, coloca el puntero al final. Si no existe lo crea. |
| x | Crea y abre para sólo escritura y coloca el puntero al principio. Si el fichero existe devuelve FALSE y genera un error |
| x+ | Crea y abre para lectura y escritura y coloca el puntero al principio. Si el fichero existe devuelve FALSE y genera un error |
| c | Abre para sólo escritura y no trunca el fichero, coloca el puntero al principio. Si no existe se crea y si existe no genera error. |
| c+ | Abre para lectura y escritura y no trunca el fichero, coloca el puntero al principio. Si no existe se crea y si existe no genera error. |

# Ejercicio

Vamos a crear ahora un script PHP que contenga un formulario de citas para el médico.

El formulario pedirá el nombre del paciente y la fecha y lo guardará en un fichero llamado "citas.txt".

Modificar el script anterior para que lo guarde en un fichero llamado "citas.html".

Los archivos contendrán una cabecera "CITAS" y otra con los nombres de las dos columnas "Nombre" y "Fecha".

Ver ejemplo 2.

# Bloqueo de Archivos

Tipos de bloqueos:

- LOCK_SH. Para solicitar un bloqueo compartido (lectura).
- LOCK_EX. Para solicitar un bloqueo exclusivo (escritura).
- LOCK_UN. Para solicitar un desbloqueo (compartido o exclusivo).

# Bloqueo de Archivos

Ejemplo con bloqueo:

```
<?php
 $fp = fopen("/tmp/lock.txt", "r+");
 if (flock($fp, LOCK_EX)) {  //Adquirir un bloqueo exclusivo
  ftruncate($fp, 0);    //Truncar el archivo
  fwrite($fp, "Escribo algo aquí\n");
  fflush($fp);  //Volcar la salida antes de liberar el bloqueo
  flock($fp, LOCK_UN);  //Liberamos el bloqueo
 } else {
  echo "¡No he podido obtener el bloqueo!";
 }
 fclose($fp);
?>
```

# file_put_contents()

A partir de PHP5 entra en juego **file_put_contents()** que sería el 3 en 1, ya que viene a sustituir a **fopen()**, **fwrite()** y **fclose()**.

Con este comando ahorramos mucho tiempo. Su sintaxis es:

```
file_put_contents($fichero,$datos,FILE_APPEND|LOCK_EX)
```

Va asociado al comando:

```
file_get_contents($fichero)
```

que lo que hace es leer el fichero y se puede asignar a una variable que posteriormente imprimimos.

Ver ejercicio 3.

# Borrado de Archivos

Si queremos borrar un archivo, podemos usar la función **unlink()** cuya sintaxis es:

```
unlink($archivo);
```

siendo **$archivo** el nombre (con ruta) del archivo que queremos eliminar. Devuelve TRUE si lo elimina y FALSE en caso contrario, generando un error.

Si lo que queremos es eliminar un directorio, podemos usar:

```
rmdir($directorio);
```

siendo **$directorio** la carpeta a eliminar, que debe estar vacía.

# Listado de Elementos

**file()** nos permite pasarle una ruta de un fichero y convertirlo a un array. Cada línea del fichero sería un elemento del array.

```
$array = file('ruta_archivo.txt');
```

Ahora podemos leer el fichero con un bucle **foreach**.

Recordemos que **is_readable()** nos permite saber si el fichero se puede leer o no.

Ver ejemplo 4.

# Navegar por Directorios

- scandir() lista los ficheros de un directorio.
- is_dir(), indica si el elemento es un directorio
- is_file(), indica si el elemento es un fichero
- is_writable(), indica si un archivo existe y se puede escribir.
- is_readable(), indica si el fichero existe y se puede leer
- filesize(), devuelve el tamaño del fichero
- opendir(), closedir(), readdir(), abre un directorio, lo cierra y devuelve el siguiente elemento del mismo.
- fileperms(), obtiene los permisos de un archivo
- fileowner(), obtiene el propietario de un archivo

# Navegar por Directorios

```
<?php
$dir = "/fotos/";
//Abre un directorio conocido y procede a leer el contenido
if(is_dir($dir)) {
 if($dh = opendir($dir)) {
  while(($file = readdir($dh)) !== false) {
   echo "Nombre de archivo: $file, Tipo de archivo: " .
   filetype($dir.$type) . PHP_EOL;
  }
  closedir($dh);
 }
}
?>
```