Anexo I - Trabajando con ficheros externos

1.1 Abrir o crear ficheros

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

- **$f1**: variable que recoge *el identificador del recurso*, (para referirnos al fichero en instrucciones posteriores).
- **Fichero:** Nombre (con extensión) del fichero, va entre comillas, si está fuera del directorio en que se encuentra el script se debe poner ruta completa (usando \).
- **Modo**: entre comillas, indica el *modo de apertura* elegido.

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

1.2 Cerrar ficheros

Una vez finalizado el uso de un fichero es necesario **cerrarlo**.

```
fclose($f1)
```

Esta función (devuelve un valor *booleano*) permite cerrar el fichero especificado en **$f1**.

1.3 Punteros internos

- **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.
- **fseek(*$f1, posición*)**: Sitúa el puntero en la posición (en bytes) **posición**.
- **ftell(*$f1*)**: Da (en bytes) la **posición actual** del puntero interno del fichero.

1.4 Lectura de ficheros

Puede hacerse de dos formas: *sin apertura previa* o *con apertura previa* del mismo.

1.4.1 Lectura de ficheros sin apertura previa

- **readfile(*fichero*):** Escribe el fichero completo. Si se pone **echo** o se recoge en una variable, también añade un **número** con el **tamaño del fichero** (en *bytes)*.
- **$var=file(*fichero*)**: Crea un *array escalar* cuyos elementos son cada una de las líneas del fichero (las líneas terminan con los salto de línea en el original).

1.4.2 *Lectura de ficheros con apertura previa*

- **fpassthru(***$f1***):** Permite la lectura completa del fichero, presenta algunas peculiaridades importantes:
  - **Cierra** el fichero después de la lectura (si se añade **fclose** da error).
  - Si el resultado se recoge en una variable, o va precedido de **echo**, además de escribir el contenido, añade el *número de bytes* de su **tamaño**.
- **fgets(***$f1,long***): Extrae** del fichero señalado una cadena (desde la *posición actual* del puntero) y de **longitud** el **menor** de tres valores:
  - El valor (en bytes) indicado en **long**.
  - D*istancia* (bytes) desde la *posición del puntero* hasta el **final** del fichero.
  - D*istancia* entre la *posición* del puntero y el **primer** *salto de línea*.
- **fgetc(***$f1,long***)**: Extrae el caracter **siguiente** al señalado por la *posición actual del puntero*.
- **fgetss(***$f1,long***)**: Extrae el texto tal cual en el ejemplo del fichero….

1.5 Escribir en un fichero

Una vez abierto (en modo que permita escritura) escribe en la posición indicada por el puntero, tiene dos formas:

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

$f1 es el *identificador de recurso*, **texto** la cadena a insertar (puede ser una variable) y **long** el número máximo de caracteres a insertarse (si texto es más larga, se cortará).

Se pueden encadenar varias cadenas de texto uniéndolas con punto (.) y para hacer un retorno de carro .”\n”

1.6 Operaciones concretas

1.6.1 *Borrado de ficheros*

unlink(*fichero*)

fichero contiene el nombre y extensión del fichero (en su caso, también el path).

1.6.2 *Duplicado de ficheros*

copy(fich1, fich2**)**

Copia el fichero **fich1** (debe indicarse nombre y extensión) en otro fichero con nombre y extensión dados en **fich2**. Da como resultado un valor *booleano*, TRUE (1) si la copia se ha realizado con éxito o FALSE (nul) si no ha podido realizarse la copia.

1.6.3 *Renombrar ficheros*

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.

1.6.4 *Funciones informativas*

- **file_exists(***fichero***):** Devuelve TRUE si el fichero existe, sino FALSE.
- **filesize(***fichero***)**: Devuelve el *tamaño* del fichero 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. Para almacenar en el array podríamos poner: **$d=stat("domingo.txt")**

1.7 Transferencia de ficheros

Proceso mediante el cual un usuario manda un fichero al servidor.

Lo primero es comprobar la configuración del **php.ini**. Es necesario que file_uploads=On (dice que PHP permite subir ficheros al servidor) y es muy útil conocer el valor de upload_max_filesize (tamaño máximo (en Mbytes) de los ficheros que pueden subidos). De no estar así hay que abrir **php.ini** y modificarlo.

La transferencia de un fichero requiere dos documentos: un *formulario* que la inicie y un *script* que la recoja.

1.7.1 El formulario

Tiene ciertas peculiaridades:

- En la etiqueta **<form>** hemos de incluir –obligatoriamente– el **method='POST'** y la encriptación **ENCTYPE = "multipart/form-data"**
- El cuerpo del *formulario* colocamos un nuevo tipo de input:

```
<input type='file' name='nm'>
```

1.7.2 La tranferencia

Una vez enviado el formulario, *el fichero transferido* se guarda en un *directorio temporal* del servidor (salvo que **php.ini** diga otra cosa). Las *variables predefinidas* **$_FILES** (*superglobales*) y **$HTTP_POST_FILES** (*arrays bidimensionales*) se recogen datos relativos al contenido del fichero y a los resultados de la transferencia.

El *primero de sus índices* es el nombre asignado al fichero en el form (name).

Los segundos índices son:

- *name*: Nombre original de fichero
- *type*: Formato
- *tmp_name:* Nombre con el que se ha guardado en el directorio temporal
- *error*: Error de transferencia (CERO para transferencia realizada con éxito, o UNO)
- *size*: Tamaño del archivo

1.7.3 Copia del fichero

El fichero transferido se encuentra de momento en un directorio temporal, es preciso hacer una copia en nuestro espacio de servidor. Utilizaremos una función ya vista:

```
copy(fich1, fich2)
```

fich1: valor contenido en, **$_FILES['nm']['tmp_name']** donde **nm** es el valor incluido como name en el formulario usado para la transferencia y **tmp_name** es una *palabra reservada* que debe escribirse exactamente con esa sintaxis (en versión de PHP anterior a 4.1.0 $HTTP_POST_FILES).

fich2: Puede ser un nombre cualquiera asignado en el propio script o el nombre original del fichero transferido. En este caso habría que recogerlo del elemento del array anterior cuyo segundo índice es **name**. Podría incluirse (entre comillas) un **path** señalando el directorio o subdirectorio donde queremos que guarde la copia. Si no, se copia en el directorio desde el que se está ejecutando el script.

1.7.4 Sintaxis alternativa

Igual de eficiente y mucho más segura. Se trata de:

```
move_uploaded_file(fich1, fich2)
```

Añade algunas ventajas:

- Comprueba que el fichero ha sido transferido mediante el método POST e impide que se *copie* en el servidor en caso de no cumplirse esa condición.
- Si la opción **safe mode=on** (configuración de php.ini en modo seguro) –por defecto está Off– comprueba, que la transferencia del fichero ha sido realizada por el mismo usuario que hace la petición de ejecución del script, evitando que alguien, *maliciosamente*, pueda *mover* ficheros desde el directorio temporal hasta el espacio de servidor.

Al usar esta función bajo Windows conviene indicar en el parámetro **fich2** la ruta absoluta completa junto con el nombre del fichero ya que –de no hacerlo así– en algunas ocasiones la imagen no será transferida al directorio desde el que se ejecuta el script.

1.7.5 Posibilidad añadida con register_globals = ON

En ese caso, la sintaxis podría ser más simple ya que, en el script indicado en la *action* del formulario, se crearía una variable cuyo nombre coincide con el valor de **name** en el **<INPUT type="FILE">** del formulario.

Esta variable (supongamos que su nombre es $archivo) tiene una importante peculiaridad, ya que, además de guardar el fichero transferido, contiene (con el formato que tenemos a continuación) tres valores muy importantes:

• **$archivo_name** (se añade **_name** al nombre de la variable) y recogerá el *nombre del fichero transferido*.

• **$archivo_size** que contiene el *tamaño del fichero*.

• **$archivo_type** que recoge el *tipo de fichero*