**Ficheros y Almacenamiento de Datos 1**

1 Abrir el Fichero

**fopen()** (booleano - false si no puede abrir el fichero)

**r** (solo lectura),  
**r+** (lectura y escritura, el contenido del fichero no se borra sino que se escribe a continuacion),  
**w** (solo escritura, sobreescribe cualquier fichero afectado),  
**w+** (escritura y lectura, si el fichero no existe intenta crearlo), **a** (escribe siempre al final del fichero, si existe o si no),  
**a+** (escribe al final del fichero y permite leer)

//apertura de fichero guardado en ordenador  
$descriptor = fopen ("../noticias.txt", "a+");  
//FTP permite abrir ficheros de lectura y escritura  
$descriptor2 = fopen ("ftp://usuario:contraseña@luisyfernanda.net/noticias.txt", "r")  
//Los ficheros HTTP solo pueen abrirse de lectura  
$descriptor3 = fopen("http://www.luisyfernanda.net/noticias.txt", "r")

2 Lectura del Fichero

**fread()** tiene como parametro un descriptor de fichero devuelto por la funcion **fopen()** y el tamaño del bloque de datos que queremos leer.

$variable = **fread($descriptor, 4000)**;

Normalmente no se conocera el tamaño del fichero entonces usamos **filesize()** asi,

$variable = **fread($descriptor, filesize("noticias.txt"))**;

Ejemplo práctico

$descriptor = **fopen ("libro.txt", "a+")**;  
$linea\_numero = 1;  
while (!feof($descriptor)) {  
 $linea = **fgets ($descriptor, 4096)**;  
 echo "linea numero: $numero\_linea es: $linea";  
 $linea\_numero++;

}  
**fclose ($descriptor)**

El bucle anterior actua hasta que la función **feof()** encuentra el final del fichero apuntado por el descriptor. Mientras que no sea el fin del fichero, la función **fgets()** irá leyendo linea a linea todo el contenido e imprimiendolo en pantalla.

Existe una funcion que nos permite almacenar todo el contenido de un fichero en un array para su posterior proceso con las funciones propias de las colecciones de datos. La funcion **explode ()** toma como parametros un caracter por donde cortar la cadena y el nombre de la variable que contiene los datos, Ejemplo

$descriptor = **fopen ("libro.txt", "a+")**;  
$linea\_numero = 1;  
$archivo = "";  
while (!feof($descriptor)) {  
 $linea = **fgets ($descriptor, 4096)**;  
 $archivo = $archivo. "linea numero: $numero\_linea es: $linea";  
 $linea\_numero ++;

}  
**fclose ($descriptor)**;  
$lista = **explode (" ", $archivo)**

En este caso **explode ()** busca en la variable $archivo espacios en blanco y va almacenando en el array caracteres que esten entre dos espacios.

Por último la función **fget()** permite leer un fichero caracter a caracter

3 Escritura de Ficheros

**fwrite()** escribe en un fichero una cadena de caracteres

Ver ejemplo libro de visitas pgs. 185-187

A. Sistema de ficheros y directorios

Copiar, borrar, y renombrar ficheros y directorios

**copy()** copia un fichero, parametros, el fichero a copiar, y el destino donde guardar la copia  
**unlink()** toma como parametro el nombre de un fichero y lo borra  
**rename()** toma como parametros el nombre original, y el nuevo nombre a darle  
  
Antes de copiar borrar o renombrar sera conveniente saber si el fichero existe con la funcion

**file\_exists()** booleano

ejemplo

public function borrar\_libro()  
{  
 if (**file\_exists($this->nombre)**) {  
 **unlink($this->nombre)**;  
 }  
}

La funcion **filetype()** devuelve un valor que indica el tipo de fichro que corresponde al valor pasado como parametro

Valores: **fifo** (de tipo FIFO), **char** (dispositivo de caracter), **dir** (directorio), **block** (dispositivo especial de bloques), **link** (enlace), **file** (fichero), **unknown** (desconocido)

Otras funciones de comprobación booleanas son

**is\_dir(), is\_\_executable(), is\_file(), y is\_link()**

**Ficheros y Almacenamiento de Datos 2**

TRABAJAR CON DIRECTORIOS

**opendir(), readdir(), closedir(),**

ejemplo

$directorio = "./";  
$descriptor = **opendir($directorio)**;  
while ($entrada = **readdir($descriptor)**) {  
 if (**is\_dir($directorio.$entrada)**) {  
 echo "[directorio]".$entrada."<br>"  
 } elseif (**is\_file ($directorio.$entrada)**) {  
 echo " [Fichero] ". $entrada."<br>";  
 }  
}

Lo primero que se hace en el ejemplo previo es crear un puntero hacia el directorio que se quiere observar y gaurdarlo en $descriptor. El bucle va leyendo entrada por entrada con la función readdir() y dependiendo si es file o dir se va imprimiendo el resultado pertinente.

En OOP, existen 3 metodos analogos **dir(), read(), y close()**  
  
ejemplo OOP

$directorio = "./";  
$fichero = **dir($directorio)**;  
while ($entrada = $fichero -> **read ()**) {  
 if (**is\_dir($directorio.(entrada)**) {  
 echo "[directorio]".$entrada."<br>"  
 } elseif (**is\_file ($directorio.$entrada)**) {  
 echo " [Fichero] ". $entrada."<br>";  
 }  
}

FICHEROS DE CONFIGURACION

Tecnicas para almacenar datos de configuración en un fichero

$servidor = "localhost";  
$usuario = "luis";  
$password = "secreta";  
$base de datos = "anaya";

Otra alternativa de configuración para conectar a una base de datos seria la siguiente, que utiliza la misma estructura que un fichero php.ini

[Base\_\_datos]  
servidor = localhost  
usuario = luis  
password = secreta  
base de datos = anaya

[Preferencias]  
color = rojo  
tamanio = medio  
fuente = verdana

Este archivo alamacenaría las variables que necesita el programa, para leerlo existe una sentencia especial que almacena el resultado en un array

$configuracion = parse\_ini\_file("configuracion.ini", TRUE);  
//Ejemplos de Variables  
echo $configuración ["Base\_\_datos"] ["servidor"];  
echo $configuracion ["Preferencias"] ["fuente"];

MANEJO DE FICHEROS EN EL SERVIDOR

SUBIDA DE FICHEROS

Ejemplo

<html>  
<body>  
<form action="subir.php" method = "POST" enctype = "multipart/form-data">  
Introduce el fichero: <input type ="file" name="fichero"><br>  
<input type = "submit">  
</form>  
<?php  
$fichero = $\_FILES["fichero"];  
foreach ($\_FILES["fichero"] as $indice => $valor) {  
 echo "$indice: $valor<br>";  
}  
if ($\_POST["tamanio"]<$\_FILES["fichero"] ["size"]) {  
 echo "<br> El fichero tiene un tamaño adecuado";  
 copy ($\_FILES["fichero"]["tmp\_name"],"archivo.txt");  
} else {  
 echo "<br> El fichero es demasiado grande";  
}  
?>

Este codigo muestra un formulario en donde se puede subir un fichero. Cuando activa el boton de subida la pagina vuelve a cargar, pero esta vez muestra información sobre el fichero y si el tamaño es adecuado o no para su almacenamiento. Si el tamaño es adecuado usamos la funcion copy para guardarlo en el servidor.

DESCARGA DE FICHEROS

Normalmente esta tarea cae en manos de html simplemente

<html>  
<body>  
<a href="ejemplo.zip"> Pincha aqui para descargar </a>  
</body>  
</html>  
  
Necesitamos php cuando queremos que los archivos a descargar se generen al instante, por ejemplo teniendo en cuenta los tipos MIME

<?php  
$fichero = "ejemplo.zip";  
$mimeType = "application/zip";  
//Si el navegador es Explorer u Opera cambiamos el tipo de cabecera  
if (strpos ($\_SERVER['HTTP\_USER\_AGENT'],'MSIE 5' I[  
 strpos ($\_SERVER['HTTP\_USER\_AGENT'], 'Opera 7')) {  
 $mimeType = "application/x-download";  
}  
//Se genera la cabecera  
header ("content-disposition: attachment;  
filename = ejemplo\_personal.zip");  
header ("content-type: " . $mimeType);  
header ("content-lenght: " . filesize($fichero));  
//Se envía el archivo al navegador  
readfile($fichero);  
?>

Los tipos MIME identifican el tipo de fichero y lo asocian a una aplicación determinada en un sistema operativo determinado. Si utiliza el tipo MIME de PDF en Windows, la aplicación asociada puede ser Acrobat REader de Adobe. CAsi todos los archivos llevan asociado un tipo MIME y sirve entre otras cosas para que el navegador reconozca el tipo de archivo que se esta descargando y optar por mostrarlo en pantalla directamente o guardarlo en alguna ubicación del disco duro. Las cabeceras HTTP deben generarse con la función header() y debe ser lo primero que se envia al navegador. Si envía algun dato o etiqueta antes que la cabecera obtendrá un error.