# U.T. 4: Recuperación y utilización de información. (Parte 2)

# Ficheros y Directorios

- Un fichero
  - Es un almacén de datos en dispositivos externos
  - Convierte los datos en persistentes al final de los programas
- Tipos de ficheros:
  - TEXTOS:
    - La información se almacena en ASCII, es legible mediante editores, los datos requieren separadores (‘ ’, ‘\n’, ‘\t’,’\r\n’)
  - BINARIOS:
    - La información es binaria, NO es legible mediante editores, los datos NO requieren separadores

# Ficheros y Directorios

- Las operaciones sobre ficheros suelen constar de tres fases:
  - Apertura del fichero
    - 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*** que se usará en el resto de funciones.
  - Procesamiento del fichero
    - Lectura
    - Escritura.
  - Cierre del fichero.

# Ficheros y Directorios: Apertura y cierre.

- **Apertura fopen()**

```
fopen( nombre_fichero, modo, include_path)
```

  - nombre_fichero:
    - local o remoto (“http://” o “ftp://”)
  - modo:
    - ‘r’ Sólo lectura. Puntero al inicio.
    - ‘r+’ Lectura/escritura. Puntero al inicio.
    - ‘w’ Sólo escritura. Puntero al inicio. Si existe el fichero borra lo que había, sino lo intenta crear.
    - ‘w+’ Lectura/escritura. Puntero al principio. Si existe el fichero borra lo que había, sino lo intenta crear.
    - ‘a’ Sólo escritura. Puntero al final. Si no existe lo intenta crear.
    - ‘a+’ Lectura/escritura. Puntero al final.
    - ‘x’, ‘x+’ Igual que w, w+ pero evitando la creación si el fichero existe.
    - ‘c’, ‘c+’ Igual que w , w+ pero sin truncar el fichero cuando ya existe.

# Ficheros y Directorios: Apertura y cierre.

- **Apertura fopen()**
  - Devuelve un **identificador** que se emplea en el resto de funciones (o FALSE en caso de error)
  - **Include_path** Especifica la lista de directorios donde las funciones require, include, fopen(), file(), readfile() y file_get_contents() buscarán ficheros.
  - **Include_path** = true -> el fichero debe buscarse en las rutas establecidoas en la directiva *include_path* de php.ini.
    - Ejemplo en Windows: include_path=".;c:\php\includes"
    - Ejemplo en Unix: include_path=".:/php/includes“
  - Por portabilidad, se recomienda encarecidamente que siempre use la bandera 'b' cuando se abran ficheros binarios con ***fopen()***.
- **Cierre fclose()**

```
fclose(identifcador)
```

# Ficheros y Directorios: Apertura y cierre.

- ¿Para qué sirve el **puntero** que devuelve *fopen()*?
  - Define un canal a través del cual se accede al fichero.
  - Desde que el fichero está abierto se trabaja con el puntero.
  - Cuando se abre el fichero, el puntero se coloca al principio del fichero para esperar instrucciones.
- Ejemplos:
  - Apertura para lectura

```
$fichero = fopen("datos.txt", 'r');
```

  - Apertura para escritura silenciando errores con @

```
$fichero = @fopen("datos.txt", 'w');
```

  - Apertura para añadir, silenciando errores

```
$fichero = @fopen("datos.txt", 'a');
```

# Ficheros y Directorios: Apertura y cierre.

- Más ejemplos:
  - $gestor = fopen("/home/rasmus/fichero.txt", "r");
  - $gestor = fopen("/home/rasmus/fichero.gif", "wb");
  - $gestor = fopen("**http://**www.example.com/", "r");
  - $gestor = fopen("**ftp://**user:password@example.com/fichero.txt", "w");
- PHP incorpora de serie envolturas para distintos protocolos tipo URL para trabajar junto con funciones del sistema de ficheros, como fopen(), copy(), file_exists() y filesize()
- La sintaxis de URL que se utiliza para describir una envoltura es:

***scheme://....***

# Ficheros y Directorios: Apertura y cierre.

- Para verificar que la operación *fopen()* ha tenido éxito:
  - Apertura para lectura silenciando errores con el operador @

    $fichero = @fopen("datos.txt", 'r');
  - Si no se ha podido abrir el fichero finaliza la ejecución del script devolviendo un mensaje de error:

```
if (!$fichero)
        die("ERROR: no se ha podido abrir el fichero de datos");
```

  - die() -> Provoca la finalización de la ejecución del script mostrando el mensaje recibido como parámetro
  - En caso contrario el script continúa ejecutándose.

# Ficheros y Directorios: Procesamiento de ficheros.

- Leer
  - **string fgets ($puntero, [bytes])**
    - Obtiene una **línea** desde el puntero del fichero
    - Se termina de leer cuando llega al final de línea, final del fichero o el último byte de datos
    - byte indica cuantos bytes (caracteres) queremos leer del fichero (opcional)
  - **string fread ($puntero, longitud)**
    - Es similar a fgets( ), pero se **lee todo el fichero (**o hasta el carácter longitud), no va línea por línea como fgets( ).

# Ficheros y Directorios: Procesamiento de ficheros.

- Leer todo el contenido y almacenar cada línea en una posición del array que devuelve.
  - **file (nombre_fichero)**

- Leer todo el contenido y devolverlo en un string.
  - **file_get_contents (nombre_fichero)**

# Ficheros y Directorios: Procesamiento de ficheros.

- Ejemplo: leer el contenido de un fichero línea a línea:

```
$a = fopen('datos.txt', 'r');
while(!feof($a)){
echo fgets($a) . '<br>';}
fclose($a);
```

- Ejemplo: leer el contenido de un fichero de una vez

```
$a = file('datos.txt');
foreach($a as $linea)
echo $linea . '<br>';
```

- Ejemplo: mostrar una página web de otro sitio (las imágenes fallan si las URL son relativas)

```
$a =file('http://www.upm.es/index.html');
foreach($a as $linea)
echo $linea;
```

# Ficheros y Directorios: Procesamiento de ficheros.

- Ejercicio 1:

  Realiza una página llamada lectura.php en la que lea el contenido de una de las páginas web hechas hasta ahora, lo muestre por pantalla y escriba el número total de bytes escritos.

# Ficheros y Directorios: Procesamiento de ficheros.

- Escribir (w ó a )
  - **fwrite( $fichero, cadena, longitud)**
    - Escribe en *$fichero* los caracteres de *cadena*.
    - Si se añade el parámetro longitud, escribe hasta que termine la cadena o hasta que se alcancen los caracteres indicados en este parámetro, lo que antes ocurra.
    - Devuelve el número de caracteres escrito
  - **fputs** es una alias de fwrite.

```
$a = fopen('datos.txt', 'w+');
fwrite($a,"nueva linea\r\n");
fputs($a,"otra linea\r\n");
fclose($a);
```

- Verificar el final de fichero
  - **feof(identificador)** (TRUE -> fin, FALSE -> no fin)
- **rewind( identificador)**
  - permite colocar el puntero al principio del fichero.

# Ficheros y Directorios: Procesamiento de ficheros.

- Otras operaciones:
  - **file_exists()**
    - Determina si existe un archivo o directorio
  - **fgetss()**
    - Idéntica a *fgets* con la diferencia de que los tags html son eliminados del archivo a medida que se lee el mismo. Obsoleto desde PHP 7.3.
    - Opcionalmente puede pasarse una lista de tags que no deben ser eliminados.
    - Ejemplo:

```
$string=fgetss($des,999999,"<b> <i> <table> <tr> <td>");
```

    - Lee una línea (de cualquier longitud) eliminando los tags html excepto los indicados como segundo parámetro. Los tags que cierran éstos tampoco son eliminados. (</td>,..)
  - **readfile(path)**
    - Lee e imprime un archivo

# Ficheros y Directorios: Procesamiento de ficheros.

- Otras operaciones:
  - **copy (origen, destino)**
    - Copiar un fichero
  - **rename (nombre_original, nombre_final)**
    - Renombrar (o mover) un fichero
  - **unlink(fichero)**
    - Borrar un fichero
  - **ftruncate (descriptor, longitud)**
    - Trunca el archivo a la longitud en bytes dada.
  - **filesize(path)**
    - Tamaño de un fichero en bytes
  - **filemtime(path)**
    - Devuelve marca de tiempo en que se modificó por última vez el fichero.
    - Es necesario formatearla con date

# Ficheros y Directorios: Procesamiento de ficheros.

- Otras operaciones .
  - **chgrp** : Cambia el grupo de un archivo.
  - **chmod** : Cambia permisos de un archivo
  - **chown** : Cambia el propietario de un archivo
  - **is_executable** : Indica si el archivo es ejecutable
  - **is _file** :Indica si el archivo es un archivo regular
  - **is_link** : Indica si el archivo es un enlace simbólico
  - **is_readable** : Indica si es posible leer el archivo
  - **is_uploaded_file** Indica si un archivo fue cargado a través de HTTP POST
  - **is_writable** : Indica si el nombre de archivo es escribible
  - **is_writeable** : Alias de is_writable
  - **filetype(path)** : Devuelve el tipo de un archivo.

Algunas no funcionarán en ficheros remotos ya que el fichero debe ser accesible vía el sistema de ficheros del servidor para poder ser examinado

# Ficheros y Directorios: Procesamiento de ficheros.

- Otras operaciones .
  - **int exif_imagetype ( string $filename ):** Lee los primeros bytes de una imagen y comprueba su firma.
  - devuelve el valor de la constante apropiada al tipo o FALSE si no se reconoce su tipo
  - Algunas constantes utilizadas por la función:

| 1 | IMAGETYPE_GIF |
|---|---|
| 2 | IMAGETYPE_JPEG |
| 3 | IMAGETYPE_PNG |
| 4 | IMAGETYPE_SWF |
| 5 | IMAGETYPE_PSD |
| 6 | IMAGETYPE_BMP |

# Ficheros y Directorios: Procesamiento de ficheros.

- ***int exif_imagetype ( string $filename )***

```
<?php
    if (exif_imagetype('imagen.gif') != IMAGETYPE_GIF) {
        echo 'La imagen no es gif';
}
?>
```

# Ficheros y Directorios: Manejo de directorios.

- Similar al procesamiento de ficheros secuenciales:
  - Se abre directorio
  - Después se procesa cada entrada del mismo
  - Se cierra el directorio

# Ficheros y Directorios: Manejo de directorios.

- Funciones:
  - Determinar la existencia
    - **is_dir( directorio)**: Determina si existe y es un directorio
  - Apertura
    - **opendir( directorio)**: Abre directorio y devuelve un descriptor
  - Lectura
    - **readdir($descriptor)**:
      - Devuelve el nombre del siguiente fichero en el directorio.
      - Los nombres de archivo son devueltos en el orden en que están en el sistema de archivos. ¡Ojo! los primeros “.” y “..”
  - Cierre
    - **closedir( $descriptor):** Cierra el recurso *$descriptor*

# Ficheros y Directorios: Manejo de directorios.

- **Ejemplo:** Mostrar los ficheros de un directorio con su tamaño:

```
$dir = opendir('.'); //abre el directorio actual
If  ($descriptor = opendir($dir)){
while(false !== ($fichero = readdir($descriptor))){
                echo "$fichero: " .filesize($dir. $fichero).'bytes <br>';}
closedir($dir);}
```

- **chdir(directorio)** : Cambia de directorio
- **getcwd(directorio) :**Devuelve el directorio actual
- **mkdir(directorio)** : Crea un directorio
- **rmdir(directorio)** : Elimina un directorio
- **basename(path)** : Devuelve la parte del path correspondiente al nombre del archivo
- **rewinddir(directorio)**: Rebobina el puntero interno de un directorio para que apunte nuevamente al comienzo del mismo.

# Incluir ficheros en PHP

- ***require*** e ***include*** nos permiten incluir en un código PHP el contenido de otro archivo.
- ***require_once*** e ***include_once*** se aseguran de que el archivo no se incluya más de una vez (para evitar problemas en la definición de constantes y funciones).
- Todas estas funciones reciben como único argumento el nombre del archivo a incluir.
- Si require(_once) no puede encontrar el archivo se interrumpe la ejecución, pero include(_once) intentaría continuar.
- El archivo requerido se interpretará como HTML a menos que contenga las etiquetas <?php y ?>.

# Incluir ficheros en PHP

*vars.php*

```
<?php
    $color = 'verde';
    $fruta = 'manzana';
    ?>
```

*test.php*

```
<?php
  echo "Una $fruta $color"; // Una
  include 'vars.php';
  echo "Una $fruta $color"; // Una manzana verde
  ?>
```