PHP 3.2

FORMULARIOS. MANEJO DE FICHEROS. PASO DE VARIABLES. SERVIDOR

1. MANEJO DE FICHEROS.

Ver todos los ejemplos en:

ejemplos_archivosDirectorios.php

PHP dispone de una serie de funciones que nos permiten manejar ficheros.

- **`fopen(archivo,modo):`**
  - Con esta función abrimos un fichero, bien sea local o una dirección de internet (http:// o ftp://).
  - Nos devuelve un valor numérico (indicador de archivo) de tipo integer que nos servirá para hacer referencia al archivo abierto.
  - En caso de no poder abrir el archivo, devuelve FALSE

  Podemos abrir un archivo de los siguientes maneras:

  - **r** sólo lectura
  - **r+** lectura y escritura
  - **w** solo escritura. Si no existe el archivo lo crea, si ya existe lo machaca.
  - **w+** lectura y escritura. Si no existe el archivo lo crea, si ya existe lo machaca.
  - **a** solo lectura. Si no existe el archivo lo crea, si ya existe empieza a escribir al final del archivo.
  - **a+** lectura y escritura. Si no existe el archivo lo crea, si ya existe empieza a escribir al final del archivo.
    - Los modos **r**, **r+** , **w** , **w+** colocan el puntero de lectura/escritura a **principio** del fichero ,
    - Los modos **a** , **a +** lo colocan al final

ejemplo: fopen con una URL:

```php
<?php
//ejemplo de fopen
//abre un archivo utilizando el protocolo HTTP
$url="http://www.ciberaula.com/";
if (!fopen($url, "r"))
{
echo "El archivo no se puede abrir\n";
}
?>
```

- **`feof(indicador_archivo):`**

  Esta función devuelve TRUE si el cabezal de lectura/escritura es encuentra al final del fichero, y FALSE en caso contrario.

- **`fgets(indicador_archivo,[longitud]):`**

  - Nos devuelve una cadena con la *longitud* específica del fichero al cual apunta el indicador de archivo .
  - El parámetro *longitud* es opcional.
  - La lectura acaba cuando se han leído *length - 1* bytes, o una nueva línea (la cual está incluida en el valor de retorno), o un *EOF* (el que suceda primero).
  - Si no se especifica una longitud, se continuará leyendo datos del flujo hasta que alcance el final de la línea.

```php
<?php
// Ejemplo de fgets
//abre un archivo e imprime cada linea
$nomArchivo="ARCHIVOS\\ej_gets.txt";
$fp = fopen( $nomArchivo, "r");
if ($fp)
{
    while (!feof($fp))
    {
    $linea = fgets($fp, 255);
    echo $linea;
    echo '<br>';
    }
fclose ($fp);
}
?>
```

- **`fgetc(resource $handle):`**
  - Obtiene un carácter de un archivo.
  - El fichero al cual se apunta tiene que ser válido, y tiene que estar abierto por fopen() .
  - Devuelve una cadena que contiene un solo carácter leído del archivo apuntado por handle.
  - Devuelve FALSE sobre EOF.

Ejemplo1:

```php
<?php
// Ejemplo de fgetc
$nomArchivo="ARCHIVOS\\ej_getc.txt";
$file = fopen($nomArchivo,"r");

while (! feof ($file))
  {
  echo fgetc($file);
  }
fclose($file);
?>
```

Ejemplo2:

Sobre la entrada standard (es a decir, **teclado**)

```
<?php

echo  'Seguro que desea abandonar? (s/n) ';

$input = fgetc(STDIN);

if ($input == 's')

exit(0);}

?>
```

- **`readfile(fichero):`**

función que visualiza el contenido de un fichero por pantalla

Ejemplo:

```php
<?php
// Ejemplo Readfile
$nomArchivo="ARCHIVOS\\ej_readfile.txt";
if(file_exists($nomArchivo)) readfile($nomArchivo);
else
    echo "El archivo no existe";
?>
```

- **`fwrite(indicador_archivo, cadena, [length]):`**
  - Escribe una cadena en el fichero indicado, el cual tiene que haber sido previamente abierto
  - La función **fwrite** devuelve el nombre de bytes escritos o FALSE, en caso que se haya producido un error.
  - El argumento *length* es opcional. Si es da, la escritura se detendrá después de que se consiga *length* nombre de bytes o que se llegue al final del string, la que cosa suceda primero

```php
<?php
// ejemplo fwrite
$nomArchivo="ARCHIVOS\\ej_fwrite.txt";
$fd1=fopen($nomArchivo,"w");
$res=fwrite($fd1,"Hola mundo!");
fclose($fd1);
?>
```

- **`fputs(indicador_archivo, cadena):`**
  - Escribe una cadena en el fichero indicado, el cual tiene que haber sido previamente abierto.
  - La función **fputs** devuelve TRUE si se ha escrito con éxito, en caso contrario devuelve FALSE.

```php
<?php
// Ejemplo fputs
//abre un archivo y escribe en él
$nomArchivo="ARCHIVOS\\ej_fputs.txt";
$archivo = fopen($nomArchivo , "w");
if ($archivo)
{
fputs ($archivo, "Hola Mundo");
}
fclose ($archivo);
?>
```

- **`filesize(String $filename)`**
  - Devuelve el tamaño en bytes de un archivo.
  - Recibe el nombre del archivo como argumento
  - A veces es usa como segundo argumento de la función *fread*

```php
<?php
//Ejemplo  filesize
$nomArchivo="ARCHIVOS\\ej_filesize.txt";
$tamanyo_fichero=  filesize($nomArchivo);
echo "<br>El tamaño del archivo <i><b>".$nomArchivo."</i></b> es:<br>";
echo "$tamanyo_fichero caracteres";
?>
```

- **`fread(resource $handle , int $length)`**
  - Lee hasta *length* bytes desde el puntero al fichero referenciado por handle
  - La lectura acaba tan pronto como es encuentre una de los siguientes condiciones:
    - *length* bytes han sido leídos
    - *EOF* (fin de fichero) es conseguido

Nota:

Normalmente como segundo argumento de *fread* es usa ***filesize ($nombre_fichero)***

Ejemplo:

```php
<?php
// ejemplo fread
// poner el contenido de un fichero en una cadena
$nomArchivo="ARCHIVOS\\ej_fread.txt";
$gestor = fopen($nomArchivo, "r");
$contenido = fread($gestor, filesize($nomArchivo));
fclose($gestor);
?>
```

- **`fclose (indicador_archivo):`**
  - Nos cierra el fichero el indicador del cual le decimos .
  - Devuelve TRUE si el fichero es cierra correctamente y FALSE si no se ha podido cerrar.
- **`file_exists (fichero):`**

Devuelve TRUE si el archivo especificado existe, y FALSE en caso contrario.

```php
<?php
// ejemplo file_exists
$nomArchivo="ARCHIVOS\\ej_file_exists.txt";
if (file_exists($nomArchivo))
{
    echo "<br>El fichero existe";
}
else
{
    echo "<br>El fichero NO existe";
}
?>
```

- **`copy (origen, destino):`**

  Copia un fichero de un lugar (origen) a un otro (destino), devuelve TRUE si la copia ha tenido éxito y FALSE en caso contrario.

```php
<?php
// ejemplo copy
$origen="ARCHIVOS\\ej_copyOrigen.txt";
$destino="ARCHIVOS\\ej_copyDestino.txt";
if (copy($origen, $destino))
{
    echo "<br>El fichero ha sido copiado con éxito";
}
else
{
    echo "<br>El fichero NO se ha podido copiar";
}
?>
```

- **`rename($fich1,$fich2):`**

  Cambia el nombre del archivo de fich1 a fich2.

  Ejemplo:

```php
<?php
// ejemplo rename
$nomInicial="ARCHIVOS\\ej_renameInicial.txt";
$nomFinal="ARCHIVOS\\ej_renameFinal.txt";
rename($nomInicial,$nomFinal);
?>
```

- **`unlink($archivo):`**

  Borra el archivo

  Ejemplo:

```php
<?php
    $nomArchivo="ARCHIVOS\\ej_unlink.txt";
    unlink($nomArchivo);
?>
```

- **`fseek (resource $handle , int $offset [, int $whence = SEEK_SIETE ]):`**
  - Establece el indicador de posición de fichero para el fichero referenciado por handle
  - La nueva posición, medida en bytes desde el inicio del fichero, se obtiene añadiendo offset a la posición especificada por whence
  - Avanza o retrocede el puntero del archivo un cierto número de posiciones
  - Si tiene éxito, devuelve 0; en caso contrario devuelve -1.

  Ver enlace:

  http://php.net/manual/es/function.fseek.php

```php
<?php
    $fp = fopen('fichero.txt','r');
    //leer alguna información
    $data= fgets($fp,4096);
    //volver al principio del fichero
    // igual que rewind($fp);
    fseek($fp,0);
?>
```

- **`fpassthru($id):`**
  - Lee completamente el archivo y lo muestra.
  - En realidad fpassthru lee el archivo indicado desde la posición actual hasta EOF y escribe los resultados en el buffer de salida (salida standard).
  - Actúa sobre un archivo que se haya abierto previamente

Nota: Puede necesitar llamar a *rewind()* para reiniciar el puntero al archivo al principio del archivo si ya se ha escrito información en el archivo.

```php
<?php
    // ejemplo fpassthru
    $nomArchivo="ARCHIVOS\\ej_fpassthru.txt";
    $fp = fopen($nomArchivo, 'r');

    // leemos la primera linea
    $primLinea=fgets($fp);
    echo "<br>Contenido de la primera línea:<br>";
    echo $primLinea;

    echo "<br>Contenido del resto del archivo:<br>";
    fpassthru($fp);
?>
```

- **`rewind ( resource $handle )`**
  - Establece el indicador de posición de archivo de handle en el principio del flujo del archivo.
  - Devuelve TRUE en caso de éxito. FALSE en caso contrario

```php
<?php
    $nomArchivo="ARCHIVOS\\ej_rewind.txt";
    $fp = fopen($nomArchivo, 'r');
    // leemos la primera linea
    $primLinea=fgets($fp);
    echo "<br>Contenido de la primera línea:<br>";
    echo $primLinea;
    // leemos la segunda linea
    $segundaLinea=fgets($fp);
    echo "<br>Contenido de la segunda línea:<br>";
    echo $segundaLinea;
    // volver al principio del fichero
    rewind($fp);
    echo "Volvemos a leer de primera línea<br>";
    $Linea=fgets($fp);
    echo "<br>Contenido de la primera línea:<br>";
    echo $Linea;
?>
```

- **`file($fichero):`**

Vuelca el contenido de un fichero en un array de tantos elementos como líneas tenga el fichero.
Ejemplo:

Pasa el contenido del archivo especificado a un array. Cuenta el nombre de elementos del array generado y muestra su contenido por pantalla.

Ver *ejemplo_fich2.php:*

```php
<?php
    $arr=file('name.txt');
    // To check the number of lines
    echo count($arr).'<br>';
    foreach($arr as $name)
    {
       echo $name.'<br>';
    }
?>
```

Ejemplo completo 1:

Ver *ejemplo_fich1.php*

- Se comprueba si existe un archivo.
- Se intenta abrir. En caso afirmativo se extrae su contenido. En otro caso se emite mensaje.
- Finalmente, se cierra

```php
<?php
if(file_exists($nombre))
{
    if($ref=fopen($nombre,"r"))
    {
        while(!feof($ref))
        {
        $linea=fgets($ref,11);
        echo $linea;
        }
                fclose($ref);
    }
    else
        echo "Error al abrir archivo";
}
else
    echo "No existe archivo";
?>
```

Ejemplo completo 2:

Ver ejemplo_fich2.php

- Se comprueba si existe el fichero, en caso afirmativo se extraen los elementos de un archivo y se vuelcan en un array con función ***file***
- En el bucle *while* extraemos uno a uno, en cada iteración, los elementos del array *$texto* mediante la función **each**.
- A continuación , la función **list** se encarga de separar , por un lado , el índice del elemento , el cual es guardado en la variable $línea, y, por otro, el contenido del elemento , que es guardado en la variable $contenido . El contenido de las dos variables, $línea y $contenido , son mostrados en el navegador.

```php
<?php
    $nombre="prueba.txt";
    if(file_exists($nombre))
    {
        // Con "file" creamos array "texto".
        //Cada línea del fich será un elem del array
        $texto=file($nombre);
        // Mostramos cada una de las líneas del array 'list' y 'each'
        while(list($linea,$contenido)=each($texto))
        {
            echo ($linea+1)." --> ".$contenido."<br>";
        }
    }
    else
        echo "El archivo no existe";
?>
```

2. MANEJO DE DIRECTORIOS.

Veremos algunas funciones que nos pueden ser muy útiles en la navegación por directorios e incluso para crear exploradores de archivos en nuestro navegador.

Podemos distinguir **tres** acciones elementales:

- Apertura del directorio , asignándole un identificador.
- Realización de las tareas necesarias en relación con ese directorio
- Cierre del directorio.

FUNCIONES PARA GESTIÓN DE ARCHIVOS/CARPETAS DEL SISTEMA OPERATIVO:

ejemplos_*comandos_directorios.php*

Funciones de gestión de directorios		
Función	**Descripción**	**Sintaxis**
opendir	Abre un directorio situado en $path y le asigna un identificador $dir	$dir = opendir ($path)
readdir	Lee un elemento del directorio $dir abierto previamente con opendir y desplaza el puntero al elemento siguiente	readdir ($dir)
rmdir	Elimina el directorio $dir	rmdir($dir)
mkdir	Crea un directorio situado en $path con los derechos de acceso $derechos (entero)	mkdir ($path, $derechos)
rewinddir	Vuelve el puntero de lectura del directorio $dir al primer elemento	rewinddir ($dir)
closedir	Cierra el directorio $dir abierto previamente con opendir	closedir ($dir)

https://www.tutorialspoint.com/php/php_directory_functions.htm

- **`resource opendir ( string $path [, resource $contexto] );`**
  - Intenta abrir la carpeta que se le indica $path
  - Abre un gestor de directorio para ser usado en llamadas posteriores como closedir(), readdir(), y rewinddir().
  - El parámetro $contexto no es obligatorio e indica varias opciones que pueden modificar el comportamiento.

Ejemplo:

```php
<?php
$ruta='D:\\projects_PHP_IAW\\UPLOADS2/';
$des=  @opendir($ruta); // incluimos @ para evitar que
                        // aparezca error cuando no existe
if ($des==true) {
echo "Existe la carpeta";
}
else
{
    echo "No Existe";
}
?>
```

- **`String readdir ( [ resource $decir_handle] );`**
  - Devuelve el nombre de la siguiente entrada del directorio. Las entradas son devueltas en el orden en que fueron almacenadas por el sistema de ficheros.
  - Devuelve el nombre de la entrada en caso de éxito o FALSE en caso de error.
  - El gestor de directorio resource previamente abierto por opendir() . Si el gestor de directorio no se especifica, la ultima conexión abierta por *opendir()* es asumida.

Ejemplo:

```php
//readdir()
$ruta='D:\\projects_PHP_IAW\\UPLOADS';
$gestor = opendir($ruta);
if ($gestor) {
echo "Identificador de gestor de directorio: <b>".$gestor.'</b><br>';
echo "<b>Contenido de la carpeta: </b><br>";

// Esta es la forma correcta de iterar sobre el directorio.
$entrada = readdir($gestor);
while ($entrada !== false )
{
    echo $entrada;
    echo '<br>';
    $entrada = readdir($gestor);
}
closedir($gestor);
}
```

- **array** scandir **(string $directory [, int $sorting_order = SCANDIR_SORT_ASCENDING [, resource $contexto]])**
  - Recibe como argumento $directory, la ruta y la forma de ordenación,
  - Enumera los ficheros y directorios situados en la ruta especificada
  - Devuelve un array ***asociativo*** en el que en cada posición está un string con el nombre de cada archivo.

```php
//scandir()
 $ruta='D:\\projects_PHP_IAW\\UPLOADS';
$ficheros1  = scandir($ruta); // sólo visualiza los nombres de archivo
$ficheros2  = scandir($ruta , 1);  // tiene en cuenta los ficheros . y ..

print_r($ficheros1);
echo '<br>';
print_r($ficheros2);
echo '<br>';

// los podemos recorrer con un foreach
foreach ($ficheros1 as $key => $value)
{
echo $key.'====>'.$value.'<br>';
}
```

- **bool** chdir **(string $directory)**
  - Cambia de directorio
  - Recibe como argumento *$directory* la ruta

```php
//chdir()
 $ruta='D:\\projects_PHP_IAW\\UPLOADS';

// directorio actual
$dirActual=getcwd();
echo "El directorio actual es: <b>".$dirActual.'</b><br>';

echo "Cambiamos de directorio....: <br>";
chdir($ruta);

// directorio actual
$dirNuevo=getcwd();
echo "El nuevo directorio es: <b>".$dirNuevo.'</b><br>';
```

- bool **chroot** (string $directory)
  - Cambia el directorio raíz del proceso actual a directorio, y cambia el directorio de trabajo actual a "/".
  - Esta función sólo se encuentra disponible en sistemas GNU y BSD, y es encuentra usando un entorno CLI, CGI o SAPI embebido. Así mismo, esta función requiere privilegios de administrador (root).

```php
//rchroot()
chroot("/ruta/al/chroot/");
```

- **bool** mkdir **(string $pathname [, int $modo = 0777 [, bool $recursive = false [, resource $contexto]]])**

  - Crea un directorio.
  - Devuelve TRUE en caso de éxito y FALSE en caso de error
  - Argumentos, solo el nombre de la carpeta a crear es obligatorio, es resto son opcionales

    pathname: La ruta del directorio.

    modo: El modo predeterminat es 0777, que significa el acceso más amplio posible.

    Para más información sobre los modos, lea los detalles en la página de *chmod().*

    Nota: modo es ignorado en Windows.

    Observa que probablemente se quiere especificar el modo como un número octal, cosa que significa que tendría que haber- un cero inicial. El modo es modificado también por la actual máscara de usuario, la cual es puede cambiar usando *umask().*

    Recursive: permite la creación de directorios anidados especificado en el parámetro *pathname*.

Ejemplo:

```php
//mkdir()
$ruta='D:\\projects_PHP_IAW\\UPLOADS\\NIVEL1';
echo "Creamos la carpeta<br>";
mkdir($ruta, 0777, true);
echo "Asignamos permisos<br>";
chmod($ruta, 0777);
```

- `bool` **`rmdir`** `( string $dirname [, resource $contexto ] )`
  - Borra el directorio especificado
  - Devuelve TRUE en caso de éxito y FALSE en caso de error

```php
//rmdir()
$ruta='D:\\projects_PHP_IAW\\UPLOADS\\NIVEL1';
echo "Vamos a borrar la carpeta<br>";

if (!is_dir('NIVEL1'))
{
    echo "Se trata de una carpeta.<br>";
    $res=rmdir($ruta);
    if ($res==true)
        echo "Carpeta Borrada<br>";
    else
        echo "No se ha borrado la carpeta<br>";
}
```

- **void** rewinddir **([resource $dir_handle])**

  Volver el gestor de directorio

  Restablece la secuencia de directorio indicada por **$dir_handle** al principio del directorio.

```php
//rewinddir()
 $ruta='D:\\projects_PHP_IAW\\UPLOADS';
```

  Ejemplo completo:

```php
<?php
$dir = "/images/";

// Open a directory, and read its contents
if (is_dir($dir)){
  if ($dh = opendir($dir)){
    // List files in images directory
    while (($file = readdir($dh)) !== false){
      echo "filename:" . $file . "<br>";
    }
    rewinddir();
    // List once again files in images directory
    while (($file = readdir($dh)) !== false){
      echo "filename:" . $file . "<br>";
    }
    closedir($dh);
  }
}
?>
```

- `bool` **`is_dir ( string $filename )`**

  Devuelve TRUE si *$filename* es un directorio y FALSE en caso contrario

- `bool` **`is_file`** `( string $filename )`

  Devuelve TRUE si *$filename* es un archivo y FALSE en caso contrario

- `void` **`closedir`** `([ resource $decir_handle ] )`

  Cierra el directorio que se indica con ese handler

  El gestor de directorio tipo *resource*, previamente abierto con *opendir()*. Si el gestor de directorio no se especifica, la última conexión abierta por *opendir()* es asumida.

```php
$ruta='D:\\projects_PHP_IAW\\UPLOADS';
$gestor = opendir($ruta);

// tratamiento del directorio
// cierre
closedir($gestor);
```

Nota 1:

Para que un directorio pueda ser borrado tiene que estar vacío

Nota 2:

`disk_total_space()`, `disk_free_space()`**:** podemos obtener el espacio total y libre, en bytes, que corresponde a un cierto directorio, la ruta del cual indicaríamos como argumento .

Nota 3:

La constante `_FILE_` contiene el camino completo y el nombre del archivo donde está contenido el código que está ejecutándose.

Funciones útiles para manejo de ficheros:

- **dirname()** ➔ toma como argumento una ruta completa de un fichero y devuelve la parte correspondiente al directorio.

- **basename**() ➔ toma como argumento una ruta completa de un fichero y devuelve nombre y extensión del archivo

Ejemplo:

```php
<?php
    // $c contiene el número de bytes disponibles en C
    $C=disk_free_space("C:");
    $T=disk_total_space("C:");
    echo "Bytes libres en la unidad c: ".$C."<br>";
    echo "Bytes totales en la unidad c: ".$T."<br>";
?>
```

Ejemplo:

```php
<?php
    $directorio = opendir("../../Imagenes/");
    while ($fichero=readdir($directorio))
    {
        if (is_dir($fichero))
            print "<b>$fichero</b> es un directorio<br>";
        else
            print "<b>$fichero</b> es un fichero<br>";
    }
?>
```

3. EJECUCIÓN DE COMANDOS DEL SISTEMA OPERATIVO PHP

FORMA 1: FICHEROS .PHP QUE CONTIENEN COMANDOS LINUX

Desde un programa PHP se pueden invocar comandos del sistema operativo.

Se trata de ficheros .php que contienen comandos linux

Ver manual en:

http://php.net/manual/es/book.exec.php

Es puede hacer de varias formas:

- exec
- system
- shell_exec
- comillas invertidas

Los vemos en la siguiente imagen y posteriormente se detalla cada una de ellas:

```
<?php
    $cmd = "whoami";
    echo `$cmd`;
    echo shell_exec($cmd);
    system($cmd);
?>
```

- `string` **`exec`** `( string $command [, array &$output [, int &$return_var ]] )`

Ejecuta el comando dado.

Parámetros

command

El comando que será ejecutado.

output

Si el argumento **output** está presente, entonces el array especificado será llenado con cada línea de la salida del comando . El espacio en blanco extra, como \n , no es incluido en este array.

Nota que si el array ya contiene algunos elementos, **exec()** anexará sus resultados al final del array. Si no desea que la función anexe los elementos, use unset() sobre el array antes de pasarlo a exec() .

return_var

Si el argumento **return_var** está presente juntamente con el argumento output, entonces el estado de retorno del comando ejecutado será escrito en esta variable.

Ejemplo:

```php
$ruta='D:\\projects_PHP_IAW\\UPLOADS/';
chdir($ruta); // Cambiaría al directorio que quiero visualizar

// versión exec
echo "<strong>versión mediante exec</strong>";
echo '<pre>';
exec ('dir',$result);
//en el vector se se indica en el segundo argumento.
//Por ello habrá que recorrer dicho vector

foreach($result as $line)
                echo $line."<br>";
echo '</pre>';
```

- `string` **`system`** `( string $command [, int &$return_var ] )`

Ejecuta un programa externo y muestra su salida

Parámetros

comando

El comando que será ejecutado.

return_var

Si el argumento ***return_var*** se encuentra presente, entonces el estado devuelto por el comando ejecutado será almacenado en esta variable.

Valores devueltos

Devuelve la última línea de la salida del comando en caso de tener éxito, y **FALSE** si ocurre un error.

Ejemplo:

```php
// versión system

//Vuelca la salida línea a línea a la pantalla
//del navegador
echo "<strong>Versión mediante system</strong>";
echo '<pre>';
$result=system('dir');
echo '</pre>';
```

- **`string` `shell_exec` `( string $cmd )`**

  Parámetros

  cmd: El comando que será ejecutado.

  Valores devueltos

  La salida del comando ejecutado o NULL si ocurre un error o el comando no produce ninguna salida.

```php
<?php
$salida = shell_exec('ls -lart');
echo "<pre>$salida</pre>";
?>
```

- **Operador de comilla invertida**

PHP soporta un operador de ejecución: las comillas invertidas (` `).

PHP intentará ejecutar el contenido entre las comillas invertidas como sí se tratara de un comando del shell;

La salida será devuelta (es a decir, no será simplemente volcada como salida; puede ser asignada a una variable).

El uso del operador de las comillas invertidas es idéntico al de shell_exec() .

Ejemplo:

```
// versión con comillas invertidas
echo "<strong>Versión con comillas invertidas</strong>";
echo '<pre>';
$result=`dir`;
echo $result;
echo '</pre>';
```

FORMA 2: FICHEROS .PHP QUE INVOCAN A ARCHIVO .SH QUE CONTIENE COMANDOS LINUX

Ahora tenemos dos archivos:

- un script linux (***shell script***)
- un ***programa php***. Este es el encargado de invocar al ***shell script***

comandosLin0.php:

```php
<?php
    $comando = shell_exec('sh comandosLin0.sh');

    echo "<pre>$comando</pre>";
?>
```

comandoslin0.sh:

```
#########
#!/bin/sh
#########

echo 'LISTADO CARPETA ACTUAL:<br>';
ls
```

Forma de invocación:

```php
// Forma1: Con shell_exec
    $comando=shell_exec('sh comandosLin0.sh');
    echo "<pre>$comando</pre>";
```