

fundacionunirioja.adrformacion.com © ADR Infor SL  
Héctor García González

## **Subida de archivos y manejo de ficheros © ADR Infor SL**

fundacionunirioja.adrformacion.com © ADR Infor SL  
Héctor García González

fundacionunirioja.adrformacion.com © ADR Infor SL  
Héctor García González

# Indice

<b>Subida de archivos y manejo de ficheros</b>	<b>3</b>
Subir archivos al servidor desde nuestra web	3
Formulario de envío de archivos	4
Recuperación y tratamiento del archivo enviado	5
Limitaciones en la subida de archivos	6
Aspectos de seguridad	6
Manejar archivos en el servidor	7
Operaciones con archivos	8
Comprobar si existe un archivo	8
Leer un archivo	9
Crear o escribir un archivo	10
Mover o renombrar un archivo	11
Eliminar un archivo	11
Operaciones con directorios	12
Comprobar si existe un directorio	12
Recorrer los archivos de un directorio	13
Crear un directorio	15
Mover o renombrar un directorio	15
Eliminar un directorio	16
Gestionar los permisos	16
Hemos aprendido	18
<b>Ejercicios</b>	<b>20</b>
Ejercicio 1: Gestor de imágenes	20
Lo necesario para comenzar	20
Pasos a seguir	20
Solución	20
Ejercicio 2: Árbol de carpetas y archivos	20
Lo necesario para comenzar	21
Pasos a seguir	21
Solución	21
<b>Recursos</b>	<b>22</b>
Enlaces de Interés	22
Preguntas Frecuentes	22
Glosario.	22

## Subida de archivos y manejo de ficheros



Al finalizar esta unidad el alumno será capaz de desarrollar formularios de subida de archivos para almacenarlos en el servidor y realizar las operaciones básicas con ficheros y directorios desde PHP.

### Subir archivos al servidor desde nuestra web

Permitir que los usuarios de nuestra web puedan enviar archivos directamente al servidor es una utilidad muy versátil que puede ser clave en multitud de procesos.

Algunas de las funcionalidades que podemos implementar gracias a la subida de ficheros por parte de los usuarios son:

1

Incluir imágenes en artículos o comentarios generados por los usuarios.

2

Subida de archivos CSV para suministrar y procesar gran cantidad de datos.

3

Alojar en el servidor archivos del usuario como currículums y documentación de todo tipo.

4

Personalización de páginas de perfil mediante imágenes.

5

Gestores de información en los que sea necesario indicar una imagen.

EJEMPLO: Gestor de productos de una tienda online.



#### Ejemplo: Envío de imágenes al servidor

Un usuario quiere cambiar la foto de perfil en una red social.

Para ello deberá enviar un archivo .jpg al servidor que la almacenará para poder mostrarla en la web.

Del mismo modo que los parámetros GET y POST son un fundamento básico de la **web 2.0**, ya que permiten al usuario la creación de contenido, la subida de archivos, al ser un suministro de información por parte de los usuarios a la web, también lo es.

Redes sociales como **Facebook**, **Instagram** o **YouTube**, entre muchas otras, no podrían existir sin la posibilidad de que el usuario subiese archivos.



La subida de archivos, no es una característica exclusiva de PHP, todos los lenguajes de programación web dispondrán de métodos para tratar las subidas de archivos.

La forma de envío de archivos desde el lado del cliente, será idéntica indiferentemente del lenguaje que usemos en el servidor. Si aprendemos un lenguaje de programación web distinto a PHP, simplemente deberemos aprender cómo recuperar los archivos enviados.

## Formulario de envío de archivos

Como sucede con el envío de parámetros, para que un usuario pueda subir archivos al servidor, necesitamos crear un formulario HTML en nuestra página web.

Este formulario tiene que contar con tres requisitos para que se pueda realizar el envío de archivos:

### input

En HTML, disponemos de un tipo de campo especialmente diseñado para el envío de archivos. Su sintaxis es la siguiente:

```
<input type="file" name="nombre_campo">
```

Su apariencia por pantalla es similar a la siguiente:

No se ha seleccionado ningún archivo.

Este tipo de campo, **permitirá al usuario seleccionar un archivo de su ordenador** o dispositivo.

Para el envío de archivos, evidentemente **debemos tener al menos un campo** de este tipo.

### method

El método de envío del formulario cuando queremos subir archivos debe ser obligatoriamente **POST**:

```
<form method="post" action="index.php">
```

### enctype

Para habilitar el envío de archivos, debemos establecer un atributo adicional en la etiqueta form:

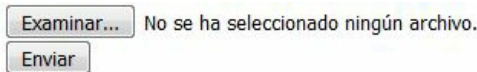
```
<form method="post" action="index.php" enctype="multipart/form-data">
```

Al indicar este atributo, establecemos que el campo de tipo **file**, debe enviar el archivo al servidor. Si no lo establecemos, enviará únicamente el nombre del archivo y no podremos recuperar su contenido.

Vamos a ver un ejemplo completo de formulario de envío de archivos:

**El siguiente fragmento de código HTML**

```
<form method="post" action="index" enctype="multipart/form-data">
<input type="file" name="archivo"><br>
<button type="submit">Enviar</button>
</form>
```

**Se visualiza en el navegador de la siguiente forma:**



Formulario de envío de archivos

## Recuperación y tratamiento del archivo enviado

Una vez que el usuario ha enviado el archivo al servidor, utilizaremos un array especial de PHP para recuperar el archivo y su información. Su sintaxis es la siguiente:

```
$_FILES['nombre_campo']
```

Este array es multidimensional y tiene varias componentes con información del archivo:

<code>\$_FILES['nombre_campo']['name']</code>	Nombre del archivo enviado.
<code>\$_FILES['nombre_campo']['tmp_name']</code>	Ruta temporal donde se ha almacenado el archivo.
<code>\$_FILES['nombre_campo']['type']</code>	Tipo MIME del archivo. Lo utilizaremos para identificar el tipo del archivo.
<code>\$_FILES['nombre_campo']['size']</code>	Tamaño en bytes del archivo.



### Ejemplo: print\_r de \$\_FILES

Si hacemos un **print\_r** de **\$\_FILES** tras el envío de un archivo, nos encontraremos una estructura como la siguiente:

```
Array
(
    [archivo1] => Array
        (
            [name] => factura.pdf
            [type] => application/pdf
            [tmp_name] => C:\xampp\tmp\php6851.tmp
            [error] => 0
            [size] => 4386
        )
)
```

Por defecto, el archivo se sube a un directorio temporal, pero es muy frecuente que queramos guardar este archivo de forma permanente. Para ello utilizaremos la siguiente sintaxis:

```
move_uploaded_file($_FILES['nombre_campo']['tmp_name'], ruta_destino);
```

`ruta_destino` será la ruta donde se guardará el archivo. Podemos utilizar tanto rutas relativas (partiendo del directorio en el que se encuentra el script de PHP) como absolutas.



Recuperación y tratamiento del archivo enviado

## Limitaciones en la subida de archivos

Ya que el usuario podría subir archivos de gran tamaño, haciendo que nuestro servidor se colapsase si no dispone de los recursos adecuados, existen varios parámetros de configuración de PHP que limitan el tamaño y el tiempo de subida.

Podemos modificar los valores de estos parámetros desde nuestro **php.ini**.

Directiva	Valor habitual	Descripción
<code>max_file_uploads</code>	20	Número máximo de ficheros que se pueden subir de forma simultánea.
<code>upload_max_filesize</code>	2M	Tamaño máximo de cada fichero subido.
<code>post_max_size</code>	8M	Tamaño máximo del conjunto de datos enviados mediante POST.
<code>max_input_time</code>	60	Tiempo máximo en segundos de envío de datos al servidor.

## Aspectos de seguridad

Debemos tener especial cuidado a la hora de gestionar los archivos que un usuario sube a nuestro servidor. Si permitimos que los usuarios suban cualquier tipo de archivo al servidor, podrían hacernos llegar archivos ejecutables con **virus o códigos malintencionados**.

Si necesitásemos guardar archivos de este tipo, siempre debemos hacerlo en una carpeta que **no sea pública**, ya que de lo contrario, el usuario **podría lanzar la ejecución de cualquier programa**.



### Ejemplo: Ataque clásico mediante envío de archivos

- Tenemos un formulario en nuestra web **<http://www.example.com/>** con una subida de archivos.
- Un usuario nos envía un archivo llamado **hack.php** y nosotros lo guardamos en la carpeta pública "**archivos**".
- El usuario llama a la siguiente dirección **<http://www.example.com/archivos/hack.php>** y ejecuta todas las instrucciones de PHP contenidas en él.
- El archivo PHP lee el contenido de todos nuestros archivos PHP y muestra el código por pantalla exponiendo la contraseña de nuestra base de datos.
- El usuario puede acceder ahora a nuestra base de datos y recuperar o modificar toda la información que desee.

Al ejecutar un código PHP, el usuario puede realizar cualquier acción, pudiendo incluso borrar o modificar nuestro código fuente.

Por lo tanto, las precauciones que siempre deberemos tomar a la hora de hacer una subida de archivos son:

1

Validar todas las extensiones de los archivos subidos y permitir únicamente las que vayamos a tratar.

**EJEMPLO:** Si el usuario va a subir una imagen, solo permitiremos archivos con extensión **jpg, jpeg, gif** o **.png**.

2

Si vamos a alojar ficheros con todo tipo de extensiones, guardar los archivos en un directorio que no esté en la carpeta pública.

**EJEMPLO:** Creamos una carpeta al lado de la carpeta pública (carpeta `htdocs` en XAMPP) llamada **archivos** y referenciamos las rutas de **`move_uploaded_file`** a dicha carpeta.

## Manejar archivos en el servidor

En PHP existen una serie de funciones que nos permiten trabajar cómodamente con los archivos del servidor, permitiéndonos un control total sobre los mismos.

El manejo del sistema de ficheros del servidor resulta muy interesante para gestionar información como imágenes, PDF u otros ficheros que se almacenan en el disco, en vez de en la base de datos.



## Enlaces: Referencia completa de funciones de manejo del sistema de ficheros

<http://php.net/manual/es/book.filesystem.php>

Vamos a ver las distintas operaciones que podemos hacer con el sistema de ficheros del servidor:

## Operaciones con archivos

### Comprobar si existe un archivo

Una de las operaciones básicas que necesitaremos realizar con nuestro sistema de ficheros es comprobar si un determinado archivo existe, o no.

Para ello utilizaremos la siguiente sintaxis:

```
if (file_exists(ruta)) {  
    // Acciones a realizar si el archivo existe.  
}
```

`ruta` será la ruta del archivo. Podemos utilizar tanto rutas relativas (partiendo del directorio en el que se encuentra el script de PHP) como absolutas.

Vamos a ver un ejemplo:

**Tenemos el archivo `index.php` en la carpeta pública del dominio `www.example.com` con el siguiente código:**

```
<?php  
    if (file_exists('archivos/foto.jpg')) {  
?>  
  
<?php  
    } else {  
?>  
    No existe la foto  
<?php  
    }  
?>
```



Si abrimos <http://www.example.com/> desde nuestro navegador y existe <http://www.example.com/archivos/foto.jpg>, mostrará:



Si abrimos <http://www.example.com/> desde nuestro navegador y no existe <http://www.example.com/archivos/foto.jpg>, mostrará:

No existe la foto

## Leer un archivo

Para leer el contenido de un archivo, utilizaremos la siguiente sintaxis:

```
$contenido = file_get_contents(ruta);
```

`ruta` será la ruta del archivo. Podemos utilizar tanto rutas relativas (partiendo del directorio en el que se encuentra el script de PHP) como absolutas.

La función introducirá dentro de la variable `$contenido` el contenido completo del archivo.

Vamos a ver un ejemplo:

**En la carpeta pública de nuestro servidor tenemos el archivo texto.txt con el siguiente contenido:**

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis pellentesque efficitur porta. Phasellus dapibus leo diam, ut molestie sem posuere ac. Ut eget aliquam orci, viverra blandit ante. Suspendisse eget nisl nisl. Fusce luctus sapien sed efficitur fermentum. Nullam a iaculis orci, id vehicula eros. Etiam sodales felis quis egestas facilisis. Duis ut dui hendrerit ligula vehicula lobortis.

**También tenemos el archivo index.php en la carpeta pública con el siguiente código:**

```
<?php
    echo file_get_contents('texto.txt');
?>
```

**Al llamar a index.php desde nuestro navegador mostrará lo siguiente:**

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis pellentesque efficitur porta. Phasellus dapibus leo diam, ut molestie sem posuere ac. Ut eget aliquam orci, viverra blandit ante. Suspendisse eget nisl nisl. Fusce luctus sapien sed efficitur fermentum. Nullam a iaculis orci, id vehicula eros. Etiam sodales felis quis egestas facilisis. Duis ut dui hendrerit ligula vehicula lobortis.



### Truco: Leer archivos de otro servidor

Con esta misma función podemos leer archivos que no se encuentren en nuestro propio servidor.

Por ejemplo podríamos ejecutar la siguiente línea de código y obtener un archivo alojado en el servidor de Google:

```
$contenido = file_get_contents('https://www.google.es/robots.txt');
```

Para poder recuperar archivos que se encuentren en otros servidores, es necesario que especificar la directiva **allow\_url\_fopen** a **On** en nuestro **php.ini**

Aunque esta directiva suele venir por defecto activada, muchas empresas de hosting la desactivan por seguridad en su configuración por defecto.

## Crear o escribir un archivo

Para crear un archivo, o sobrescribir el contenido de un archivo, utilizaremos la siguiente sintaxis:

```
file_put_contents(ruta,$contenido);
```

`ruta` será la ruta del archivo. Podemos utilizar tanto rutas relativas (partiendo del directorio en el que se encuentra el script de PHP) como absolutas.

La función escribirá el contenido de `$contenido` en el archivo especificado.

Vamos a ver un ejemplo:

**Tenemos el archivo `index.php` en la carpeta pública con el siguiente código:**

```
<?php
    file_put_contents('texto.txt','Hola mundo');
?>
```

**Al llamar a `index.php` desde nuestro navegador se creará el archivo `texto.txt` en la carpeta pública con el siguiente contenido:**

Hola mundo

## Mover o renombrar un archivo

Para cambiar de nombre un archivo o moverlo, utilizaremos la siguiente sintaxis:

```
rename(ruta_actual,ruta_nueva);
```

`ruta_actual` será la ruta donde se encuentra actualmente el archivo. Podemos utilizar tanto rutas relativas (partiendo del directorio en el que se encuentra el script de PHP) como absolutas.

`ruta_nueva` será la nueva ruta del archivo.

Con la misma instrucción, podemos indicar tanto un cambio de nombre, como un cambio de directorio, ya que indicaremos la ruta completa, incluido el nombre del archivo.



```
rename('archivo.txt','documentos/info.csv');
```

Con esta instrucción, hemos cambiado el nombre de `archivo.txt` a `info.csv` y lo hemos movido a la carpeta **documentos**.

## Eliminar un archivo

Para eliminar un archivo, utilizaremos la siguiente sintaxis:

```
unlink(ruta);
```

`ruta` será la ruta del archivo. Podemos utilizar tanto rutas relativas (partiendo del directorio en el que se encuentra el script de PHP) como absolutas.



```
unlink('archivo.txt');
```

Con esta instrucción eliminaremos el fichero **archivo.txt**



Operaciones con archivos

## Operaciones con directorios

### Comprobar si existe un directorio

Para comprobar si un directorio existe, utilizaremos la misma sintaxis que para comprobar si existe un archivo. La función **file\_exists** puede utilizarse tanto para archivos como para carpetas:

```
if (file_exists(ruta)) {
    // Acciones a realizar si el directorio existe.
}
```

**ruta** será la ruta del directorio. Podemos utilizar tanto rutas relativas (partiendo del directorio en el que se encuentra el script de PHP) como absolutas.

Vamos a ver un ejemplo:

**Tenemos el archivo index.php en la carpeta pública con el siguiente código:**

```
<?php
    if (file_exists('documentos')) {
?>
Existe el directorio documentos.
<?php
    } else {
?>
No existe el directorio documentos.
<?php
    }
?>
```

Si existe el directorio "documentos" el navegador mostrará:

Existe el directorio documentos.

Si no existe el directorio "documentos" el navegador mostrará:

No existe el directorio documentos.

## Recorrer los archivos de un directorio

Para obtener todos los archivos y subdirectorios de un directorio, utilizaremos la siguiente sintaxis:

```
$directorio = opendir(ruta);  
while ($entrada = readdir($directorio)) {  
    // Acciones a realizar para cada elemento del directorio  
}
```

`ruta` será la ruta del directorio. Podemos utilizar tanto rutas relativas (partiendo del directorio en el que se encuentra el script de PHP) como absolutas.

El bucle recorrerá todos los elementos que se encuentren dentro del directorio y guardará en `$entrada` el nombre del archivo o subdirectorio actual.



### Importante: Directorio . y ..

Este conjunto de instrucciones también recuperará los directorios especiales. y .. que hacen referencia al directorio actual y al directorio superior respectivamente.

Suele ser habitual que antes de realizar el proceso que necesitemos realizar para cada elemento del directorio, comprobemos que el elemento actual no es ninguno de estos dos directorios.

### EJEMPLO:

```
while ($entrada = readdir($directorio )) {
    if (($entrada != '.') && ($entrada != '..')) {
        // Acciones a realizar para cada elemento del directorio
    }
}
```

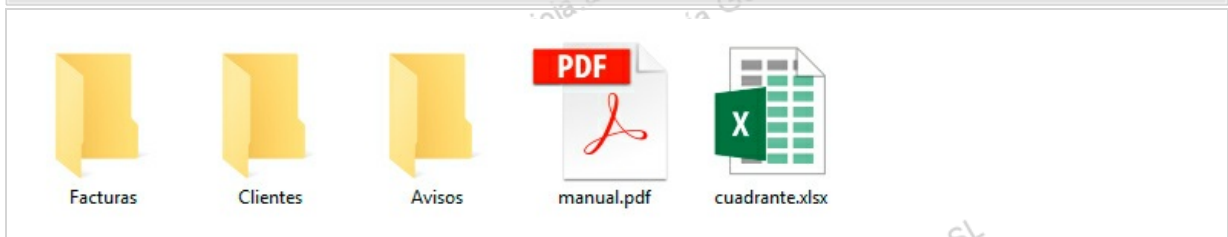
Cuando recuperamos los elementos de un directorio, necesitaremos saber si el elemento recuperado es **un archivo o un directorio**. Lo más frecuente es que no realicemos el mismo proceso para los archivos que para los directorios.

Contamos con un par de funciones que nos indicarán si un elemento es un archivo, o un directorio:

Función	Descripción
is_dir(ruta)	Devuelve true si la ruta indicada pertenece a un directorio.
is_file(ruta)	Devuelve true si la ruta indicada pertenece a un archivo.

Vamos a ver un ejemplo completo:

**Tenemos una carpeta dentro de la carpeta pública llamada "ejemplo" que contiene los siguientes elementos:**



En la carpeta pública tendremos el archivo index.php con el siguiente código:

```
<?php
$directorio = opendir('ejemplo');
while ($entrada = readdir($directorio)) {
    if (($entrada != '.') && ($entrada != '..')) {
        if (is_dir('ejemplo/' . $entrada)) {
            echo "Directorio: $entrada<br>";
        } else {
            echo "Archivo: $entrada<br>";
        }
    }
}
?>
```

Al llamar a index.php desde el navegador, mostrará lo siguiente:

```
Directorio: Avisos
Directorio: Clientes
Archivo: cuadrante.xlsx
Directorio: Facturas
Archivo: manual.pdf
```

## Crear un directorio

Para crear un directorio utilizaremos la siguiente sintaxis:

```
mkdir(ruta);
```

*ruta* será la ruta del directorio. Podemos utilizar tanto rutas relativas (partiendo del directorio en el que se encuentra el script de PHP) como absolutas.



```
mkdir('informes');
```

La instrucción anterior creará la carpeta **informes**.

## Mover o renombrar un directorio

Para mover o renombrar un directorio, utilizaremos la misma sintaxis que para mover o renombrar un archivo. La función **rename** puede utilizarse tanto para archivos como para carpetas:

```
rename(ruta_actual,ruta_nueva);
```

`ruta_actual` será la ruta donde se encuentra actualmente el directorio. Podemos utilizar tanto rutas relativas (partiendo del directorio en el que se encuentra el script de PHP) como absolutas.

`ruta_nueva` será la nueva ruta del directorio.



```
rename('informes','documentos/reportes');
```

Con esta instrucción, hemos cambiado el nombre del directorio **informes** a **reportes** y lo hemos movido a la carpeta **documentos**.

## Eliminar un directorio

Para eliminar un directorio utilizaremos la siguiente sintaxis:

```
rmdir(ruta);
```

`ruta` será la ruta donde se encuentra el directorio. Podemos utilizar tanto rutas relativas (partiendo del directorio en el que se encuentra el script de PHP) como absolutas.



```
rmdir('informes');
```

Con esta instrucción, hemos eliminado el directorio **informes**.



## Operaciones con directorios

## Gestionar los permisos

Un aspecto que debemos tener en cuenta a la hora de trabajar con el sistema de ficheros desde PHP son los permisos de los archivos y carpetas.



Sólo podremos realizar operaciones de modificación o lectura sobre los elementos en los que tengamos permiso para realizar dichas operaciones.

Será especialmente relevante si estamos utilizando un servidor Linux ya que conlleva una gestión de los permisos más estricta que otros sistemas como Windows o Mac.





### Anotación: Usuario que debe poseer los permisos

El usuario que debe poseer los permisos necesarios, debe ser el usuario que ejecuta el servidor web.

Si estás utilizando un servidor **Apache en Linux**, por defecto, este usuario será **apache** en la mayoría de distribuciones.

Si utilizas sistemas **Windows o Mac**, este usuario será el que ha arrancado el servidor. En el caso de que se inicie mediante un servicio del sistema, el usuario será la cuenta por defecto para servicios del sistema.

Desde PHP podemos modificar los permisos de un archivo o carpeta mediante la siguiente instrucción:

```
chmod(ruta,permisos);
```

**ruta** será la ruta donde se encuentra el archivo o directorio. Podemos utilizar tanto rutas relativas (partiendo del directorio en el que se encuentra el script de PHP) como absolutas.

**permisos** serán los permisos a asignar expresados mediante un **octal**. Para indicar que un número está expresado en octal en PHP debemos indicar un **0** antes del número. Ejemplo: 0655 es un número expresado en octal.



```
chmod('documento.pdf',0777);
```

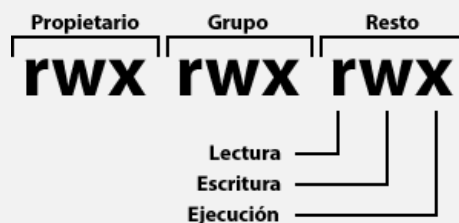
Con esta instrucción, hemos asignado todos los permisos al archivo documento.pdf



### Cómo expresar permisos en octal

Expresar permisos en octal es la forma más habitual de expresar permisos en sistemas Linux. Si has trabajado con Linux anteriormente, estarás familiarizado con el tema.

Si no es así te indicamos cómo puedes calcular los permisos que deseas indicar.



Si te fijas en la figura anterior, verás que hay tres secciones de permisos:

1. Propietario del fichero

2. Grupo del propietario del fichero
3. Resto de usuarios

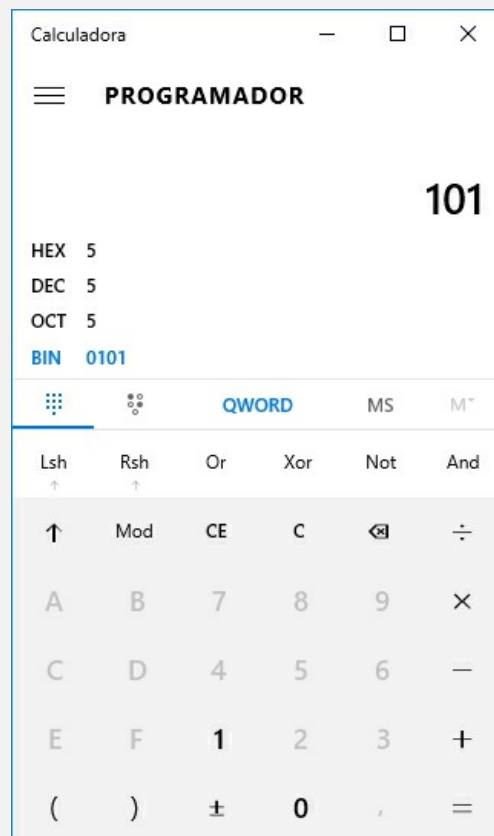
En cada sección indicamos si el usuario tiene los siguientes permisos:

1. Lectura
2. Escritura
3. Ejecución

Indicaremos un 1 en las posiciones en las que deseemos que el archivo posea dicho permiso y un 0 en el resto.

#### EJEMPLO: 111 110 100

Cada una de las tres secciones expresa un número en binario, que deberemos transformar a octal. Podemos utilizar la calculadora de Windows en modo programador para calcular la conversión.



#### EJEMPLO: 764

Con este permiso octal indicamos que el propietario posee todos los permisos, los usuarios del grupo del propietario tienen permisos de lectura y escritura, pero no de ejecución y el resto de usuarios solo tiene usuarios de lectura.

**Hemos aprendido**



En esta unidad hemos aprendido:

- Para subir archivos al servidor utilizaremos un formulario con las siguientes características:
  - Debe contener al menos una etiqueta **<input>** de tipo **file**.
  - El método de envío del formulario debe ser **POST**.
  - Debe tener el atributo **enctype="multipart/form-data"**.
- Para recuperar el archivo enviado utilizaremos el array **\$\_FILES**.
- Debemos tener en cuenta las **limitaciones establecidas en las directivas de PHP** sobre la subida de archivos.
- Es necesario tener muy en cuenta **qué tipos de archivos** permitimos subir y **dónde los guardamos** para evitar **problemas de seguridad**.
- Utilizaremos las funciones **file\_exists**, **file\_get\_contents**, **file\_put\_contents**, **rename** y **unlink** para gestionar archivos en el servidor.
- Utilizaremos las funciones **file\_exists**, **opendir**, **readdir**, **is\_dir**, **is\_file**, **mkdir**, **rename** y **rmdir** para gestionar carpetas en el servidor.
- Podemos gestionar los permisos de los archivos y carpetas del servidor con la función **chmod**.

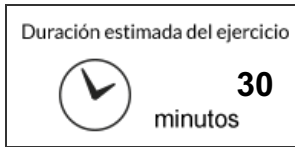
fundacionunirioja.adrformacion.com © ADR Infor SL  
Héctor García González

fundacionunirioja.adrformacion.com © ADR Infor SL  
Héctor García González

© Infor SL

## Ejercicios

### Ejercicio 1: Gestor de imágenes



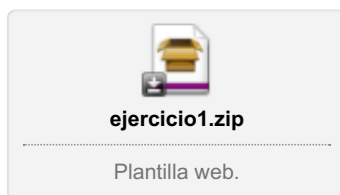
Para completar correctamente este ejercicio deberás desarrollar un pequeño gestor de imágenes.

El gestor deberá permitir almacenar los archivos de las imágenes en una carpeta del servidor, permitiendo eliminar o añadir más imágenes a voluntad.

### Lo necesario para comenzar

Descarga esta página y sítela a través de tu servidor web. Deberás descomprimir los archivos que contiene el fichero ZIP en el directorio público de tu servidor. Te servirá como base para desarrollar el ejercicio correctamente.

En la plantilla encontrarás ya maquetado el gestor de imágenes con todas sus funcionalidades.



### Pasos a seguir

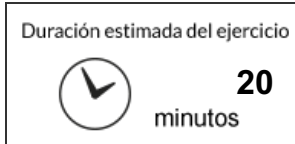
1. Muestra todas las imágenes de la carpeta **archivos** del servidor.
2. Dota de funcionalidad al botón **Añadir** permitiendo agregar imágenes a la carpeta **archivos**.
3. Dota de funcionalidad al botón **Eliminar** permitiendo eliminar imágenes del servidor.

### Solución



Solución al ejercicio

### Ejercicio 2: Árbol de carpetas y archivos

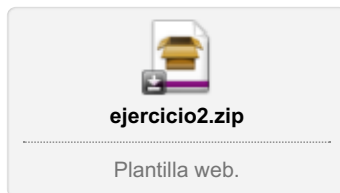


Para completar correctamente el ejercicio, deberás mostrar el árbol de carpetas y subcarpetas completo de un directorio dado, indicando los archivos que contiene cada una.

## Lo necesario para comenzar

Descarga esta página y sítela a través de tu servidor web. Deberás descomprimir los archivos que contiene el fichero ZIP en el directorio público de tu servidor. Te servirá como base para desarrollar el ejercicio correctamente.

En la plantilla encontrarás ya maquettato el árbol de archivos.



## Pasos a seguir

1. Muestra todos los archivos y carpetas del directorio dado.
2. Repite la operación cada vez que te encuentres una carpeta para mostrar el árbol completo.

## Solución



Solución al ejercicio

## Recursos

### Enlaces de Interés



<http://php.net/manual/es/book.filesystem.php>  
<http://php.net/manual/es/book.filesystem.php>

Referencia completa de funciones de manejo de ficheros de PHP en la página oficial.

### Preguntas Frecuentes

1. **Estoy intentando subir un archivo al servidor pero la componentetmp\_name de \$\_FILES aparece vacío.**

Si **tmp\_name** está vacío, significa que el archivo no se ha subido correctamente. Verifica los límites de subida de archivos en las directivas de **php.ini** para comprobar que no estés excediendo ninguno de ellos.

2. **Quiero leer el archivo que el usuario envía al servidor, pero no necesito guardarlo ¿Es obligatorio guardarlo para leerlo?**

No, puedes leer directamente el contenido del archivo desde la ubicación temporal sin necesidad de guardarlo. Puedes usar la siguiente sintaxis:

```
file_get_contents($_FILES['archivo']['tmp_name']);
```

3. **¿Puedo seleccionar varios archivos simultáneamente en un solo input para subirlos al servidor?**

Si trabajamos exclusivamente con HTML y PHP no puedes hacerlo. Cada input de tipo file, sube siempre un único archivo al servidor.

No obstante existen soluciones para poder realizar esta funcionalidad. La más simple es utilizar javascript para generar un formulario en directo cuando se seleccionan los archivos. Este formulario se deberá crear con los input necesarios para subir el número de archivos seleccionados.

### Glosario.

- **Linux:** Sistema operativo de código abierto y gratuito que es ampliamente utilizado para servidores y otros usos profesionales de equipos informáticos.
- **Octal:** Sistema numérico en base 8. Utiliza los dígitos del 0 al 7. Suele ser muy empleado en informática ya que un byte siempre está compuesto por 8 bits.

- **Tipo MIME:** Son convenciones dirigidas al intercambio a través de Internet de todo tipo de archivos. Mediante una serie de cadenas de texto estandarizadas, especifican de qué tipo es un archivo determinado.