

## Gestión de archivos por PHP

El tratamiento de archivos resulta ser una práctica muy común en cualquier sitio web. Muy a menudo nos vemos en la necesidad de procesar un texto para cambiarle el formato, buscar una cadena en su interior o cualquier otro tipo de operación.

PHP propone un sinfín de funciones para la gestión de archivos que van desde las más elementales de apertura, lectura y cierre a otras más rebuscadas como el cálculo de espacio en el disco duro, tamaño del archivo, gestión de derechos de acceso...

Pretendemos mostraros cuáles son las funciones más esenciales para el tratamiento de archivos para posteriormente ejemplificarlas en un par de scripts que pueden resultar útiles:

Funciones de gestión de archivos		
Función	Descripción	Sintaxis
<b>copy</b>	Copia un archivo	copy(\$origen,\$destino)
<b>rename</b>	Cambia el nombre del archivo de <i>\$antes</i> a <i>\$despues</i>	rename(\$antes,\$despues)
<b>unlink</b>	Borra el archivo	unlink(\$archivo)

Funciones para la lectura de archivos		
Función	Descripción	Sintaxis
<b>fopen</b>	Abre un archivo y le asigna un identificador id. Veremos el modo más adelante	\$id = fopen(\$archivo, \$modo)
<b>fgets</b>	Lee una línea de un archivo hasta un número máximo de caracteres	fgets(\$id,\$max)
<b>fwrite</b>	Escribe una cadena dentro del archivo	fwrite(\$id, \$cadena)
<b>fseek</b>	Avanza o retrocede el puntero del archivo un cierto número de posiciones	fseek(\$id,\$posiciones)
<b>feof</b>	Comprueba si el puntero que lee el archivo ha llegado al final	feof(\$id)
<b>fpassthru</b>	Lee completamente el archivo y lo muestra	fpassthru(\$id)
<b>fclose</b>	Cierra el archivo abierto previamente	fclose(\$id)

Las operaciones más elementales, copia, borrado y cambiar el nombre, requieren únicamente el nombre (y path) del archivo sobre el cual se ejerce la operación. Para operaciones más complejas, como la lectura de líneas o la escritura de texto dentro del archivo, se requiere de una previa apertura del archivo al cual le asignaremos un identificador *\$id*.

Una vez abierto el archivo, podremos desplazarnos a lo largo de él por medio de un puntero imaginario que avanza o retrocede por las líneas de texto y mediante el cual nos situaremos en el lugar escogido para insertar, modificar o simplemente copiar una cadena de texto.

Existen distintos modos de apertura que nos permiten definir las acciones que podemos realizar sobre el archivo. Aquí os mostramos los diferentes modos que, como veréis, son de lo más variado:

Modos de apertura de archivos	
Sintaxis	Descripción
'r'	Sólo lectura
'r+'	Lectura y escritura
'w'	Sólo escritura
'w+'	Lectura y escritura. Suprime el contenido anterior si se escribe. El archivo es creado si no existe.

'a'	Sólo escritura. El archivo es creado si no existe y el puntero se coloca al final.
'a+'	Lectura y escritura. El archivo es creado si no existe y el puntero se coloca al final.

## **Ejemplo: Lectura secuencial de archivos con PHP**

Aprenderemos a leer el contenido del archivo, tarea que llevaremos a cabo por medio de la función *fgets*.

Esta función se encarga de leer línea a línea el contenido de un archivo texto por lo que su utilización ha de ser incluida dentro de una estructura de tipo bucle o loop.

En el ejemplo que mostramos a continuación nos hemos servido de esta lectura secuencial para localizar dentro del texto una cadena cualquiera a la que, a continuación, le cambiamos el formato para ponerla en negrita por medio de la etiqueta `<b>` de HTML. Esto nos puede resultar útil si llevamos a cabo búsquedas internas en nuestro sitio y queremos resaltar la cadena de búsqueda en el texto de la página encontrada.

Evidentemente, la utilidad de *fgets* resulta ser mucho más amplia. Podemos emplearla, por ejemplo, con archivos remotos para extraer las etiquetas meta de HTML o para muchas otras cosas que se nos puedan ocurrir.

```

<?php
function negrita($path,$cadena)
{
    //Iniciamos la variable
    $texto = "";
    //Abrimos el archivo en modo lectura
    $fp = fopen($path,"r");
    //Leemos linea por linea el contenido del archivo
    while ($linea= fgets($fp,1024))
    {
        //Sustituimos las ocurrencias de la cadena que buscamos
        $linea = str_replace($cadena,"<b>$cadena</b>",$linea);
        //Anadimos la linea modificada al texto
        $texto .= $linea;
    }
    return $texto;
}
//Definimos el path y la cadena
$path="escribe el camino de acceso a tu archivo";
$cadena = "escribe tu cadena";
//Llamamos la funcion
$texto = negrita ($path,$cadena);
//Mostramos el texto
echo $texto;
?>

```

Se podría ver el resultado de esta función en una variante del script anterior donde se incluya un formulario para recibir el parámetro *cadena* y buscar las ocurrencias de dicha cadena dentro del texto del archivo deseado.

Nuestro ejemplo se vería así

Introduce la cadena de búsqueda:

 

El script es utilizado en forma de función para facilitaros su empleo, reutilización y almacenamiento. Su modo de actuar es el siguiente:

- Inicializa la variable *texto* en la cual iremos almacenando las líneas leídas en el bucle.
- Abre el archivo (local o remoto) en modo lectura por medio de la función *fopen*.
- Lee una por una las líneas del archivo hasta una longitud de 1024 caracteres y reemplaza las posibles ocurrencias de la cadena de búsqueda por la misma cadena colocada entre las etiquetas `<b>` y `</b>` por medio de la función *str\_replace*. El texto, modificado o no, es almacenado en la variable *texto*.
- Devolvemos la variable *texto* como resultado de la función.

El resto de script es simplemente un ejemplo de llamada a la función donde los parámetros *path* y *cadena* han de ser especificados.