Operaciones con ficheros

PHP ofrece muchas funciones para trabajar con ficheros en el servidor. Vamos a empezar con las más básicas.

fopen: permite abrir un archivo. Se utiliza para crear un recurso de archivo que luego se puede utilizar con otras funciones de manejo de archivos.

Sintaxis:
fopen(string $filename, string $mode): resource|false

- **$filename**: La ruta al archivo o URL que deseas abrir.
- **$mode**: El modo en el que se abrirá el archivo. Determina si el archivo es para lectura, escritura, o ambos, y si se debe crear un nuevo archivo si no existe.

Los modos de apertura más comunes son:

- **'r'**: Abre el archivo solo para lectura. El puntero del archivo comienza al inicio del archivo.
- **'r+'**: Abre el archivo para lectura y escritura. El puntero del archivo comienza al inicio del archivo.
- **'w'**: Abre el archivo solo para escritura. Si el archivo no existe, intenta crearlo. Si el archivo existe, borra el contenido actual.
- **'w+'**: Abre el archivo para lectura y escritura. Si el archivo no existe, intenta crearlo. Si el archivo existe, borra el contenido actual.
- **'a'**: Abre el archivo para solo escritura. El puntero del archivo se sitúa al final del archivo. Si el archivo no existe, intenta crearlo.
- **'a+'**: Abre el archivo para lectura y escritura. El puntero del archivo se sitúa al final del archivo para escritura. Si el archivo no existe, intenta crearlo.
- **'x'**: Crea y abre un nuevo archivo solo para escritura. Retorna false y genera un error si el archivo ya existe.
- **'x+'**: Crea y abre un nuevo archivo para lectura y escritura. Retorna false y genera un error si el archivo ya existe.

fclose: se utiliza para cerrar un recurso de archivo abierto. Esta función es esencial para la gestión de recursos y para asegurar que los datos se escriben correctamente en el archivo y que no quedan bloqueos de archivo innecesarios.

Sintaxis:
fclose(resource $handle): bool

- **$handle**: Es el recurso de archivo que se quiere cerrar. Este recurso es típicamente obtenido a través de una llamada a fopen() o a funciones similares que abren archivos.

```
<?php
   $archivo = "ejemplo.txt";
   $modo = "a+";

   // Abre el archivo para lectura y escritura;
   // crea el archivo si no existe
   $manejador = fopen($archivo, $modo);

   if ($manejador === false) {
         echo "No se pudo abrir el archivo.";
   } else {
         echo "Archivo abierto con éxito.";
         // Aquí se pueden realizar operaciones adicionales con el archivo.

         // Finalmente, cierra el archivo
         fclose($manejador);
   }
?>
```

Ejercicio: Vamos a crear dos ficheros (archivo1.txt y archivo2.txt) que contengan un párrafo de un texto cualquiera. Luego vamos a hacer un script para probar a abrirlos en distintos modos cada uno (r+ y w), a ver qué sucede.
Después vamos a añadir otro fopen en modo w, con un archivo3.txt (este no lo hemos creado previamente).

```php
<?php
   $miarchivo=fopen("archivo1.txt", "r+");
   if ($miarchivo == false) {
         echo "error al abrir el archivo1.txt";
   }


   $miarchivo2=fopen("archivo2.txt", "w");
   $miarchivo3=fopen("archivo3.txt", "w");

   fclose($miarchivo);
   fclose($miarchivo2);
   fclose($miarchivo3);

?>
```

fread es una función en PHP utilizada para leer una cantidad específica de bytes desde un archivo abierto. Esta función es comúnmente usada para leer el contenido de archivos en PHP, especialmente cuando se trabaja con archivos binarios o cuando se necesita un control preciso sobre la cantidad de datos leídos.

Sintaxis:
fread(resource $handle, int $length): string|false

- **$handle**: Un recurso de archivo que se obtiene al abrir un archivo con fopen() u otra función similar.
- **$length**: La cantidad máxima de bytes a leer. Es importante notar que fread() leerá hasta length bytes o hasta el final del archivo, lo que ocurra primero.

```php
<?php
   $archivo = fopen("archivo1.txt", "r");
   $contenido = fread($archivo, filesize("archivo1.txt"));
   fclose($archivo);

   echo $contenido;
?>
```

La función filesize() en PHP se utiliza para obtener el tamaño de un archivo en bytes. Es una función útil cuando necesitas saber cuántos datos contiene un archivo, por ejemplo, antes de leerlo o al realizar operaciones de manejo de archivos.
filesize(string $filename): int|false

fgetc() es una función en PHP utilizada para leer un único carácter desde un archivo abierto.
Sintaxis:
fgetc(resource $handle): string|false

fgets() es una función en PHP que se utiliza para leer una línea desde un archivo abierto.
Sintaxis:
fgets(resource $handle, int $length = ?): string|false

- **$handle**: Un recurso de archivo abierto.
- **$length**: Opcional. Define la longitud máxima de la línea a leer.

```
<?php
$archivo = fopen("ejemplo.txt", "r");
while (($linea = fgets($archivo)) !== false) {
        echo $linea;
}
fclose($archivo);
?>
```

feof() es una función en PHP que verifica si el puntero del archivo ha llegado al final del archivo (EOF).
Sintaxis:
feof(resource $handle): bool

```
<?php
$archivo = fopen("ejemplo.txt", "r");
while (!feof($archivo)) {
        $linea = fgets($archivo);
        if ($linea !== false) {
        echo $linea;
        }
}
fclose($archivo);
?>
```

Ejercicio: Usando fgetc y feof, leer un fichero y mostrar su contenido respetando los saltos de línea.

```php
<?php
$nombreArchivo = "ejemplo.txt";

// Abrir el archivo en modo de lectura
$archivo = fopen($nombreArchivo, "r");

if (!$archivo) {
	die("Error al abrir el archivo");
}

// Leer y mostrar cada carácter hasta el final del archivo
while (!feof($archivo)) {
	$caracter = fgetc($archivo);

	if ($caracter === false) {
	break;
	}

	// Reemplazar el salto de línea con <br> para el navegador
	if ($caracter === "\n") {
	echo "<br>";
	} else {
	echo $caracter;
	}
}

// Cerrar el archivo
fclose($archivo);
?>
```

Ejercicio: Usando fgetc y feof, contar cada vocal cada vez que aparezca en el fichero y mostrarlo por pantalla.

```php
<?php
$nombreArchivo = "ejemplo.txt";

// Contadores para cada vocal
$contadorVocales = [
	'a' => 0,
	'e' => 0,
	'i' => 0,
	'o' => 0,
	'u' => 0
];

// Abrir el archivo en modo de lectura
$archivo = fopen($nombreArchivo, "r");

if (!$archivo) {
	die("Error al abrir el archivo");
}

// Leer y contar las vocales hasta el final del archivo
while (!feof($archivo)) {
	$caracter = fgetc($archivo);

	// Convertir a minúscula para la comparación
	$caracter = strtolower($caracter);

	// Comprobar si el carácter es una vocal
	if ($caracter == 'a' || $caracter == 'e' || $caracter == 'i' || $caracter == 'o' ||
$caracter == 'u') {
	$contadorVocales[$caracter]++;
	}
}

// Cerrar el archivo
fclose($archivo);

// Mostrar los resultados
foreach ($contadorVocales as $vocal => $conteo) {
	echo "La vocal '$vocal' aparece $conteo veces.\n";
}
?>
```

file_get_contents() es una función que se utiliza para leer todo el contenido de un archivo en una sola operación. Esta función es muy útil y eficiente para la lectura de archivos, ya que simplifica la obtención del contenido completo sin necesidad de abrir y cerrar explícitamente el archivo, ni de manejar bucles de lectura.

Sintaxis:
file_get_contents(string $filename, bool $use_include_path = false, resource $context = ?, int $offset = 0, int $maxlen = ?): string|false

- **$filename**: La ruta del archivo del que se desea leer el contenido. Puede ser una ruta local o una URL.

```
<?php
$contenido = file_get_contents("ejemplo.txt");

if ($contenido === false) {
        echo "No se pudo leer el archivo.";
} else {
        echo $contenido;
}
?>
```

file() se utiliza para leer todo el contenido de un archivo y devolverlo en forma de un array. Cada elemento del array corresponde a una línea del archivo, incluyendo los saltos de línea.
Sintaxis:
file(string $filename, int $flags = 0, resource $context = ?): array|false

- **$filename**: La ruta al archivo que deseas leer.

```
<?php
$nombreArchivo = "ejemplo.txt";

$lineas = file($nombreArchivo);

if ($lineas == false) {
        echo "No se pudo leer el archivo.";
} else {
        foreach ($lineas as $linea) {
        echo $linea;
        }
//var_dump($lineas);
}
?>
```

fwrite() es una función en PHP utilizada para escribir datos en un archivo. Esta función es fundamental para operaciones de archivos en las que necesitas guardar datos en un archivo.

Sintaxis:
fwrite(resource $handle, string $string, int $length = ?): int|false

- **`$handle`**: Un recurso de archivo que se obtiene al abrir un archivo con `fopen()` u otra función similar de apertura de archivos.
- **`$string`**: Los datos que deseas escribir en el archivo.

```
<?php
$archivo = fopen("ejemplo.txt", "w");

if ($archivo === false) {
	die("Error al abrir el archivo");
}

$texto = "Ejemplo de texto para escribir en el archivo.";
$numBytesEscritos = fwrite($archivo, $texto);

if ($numBytesEscritos === false) {
	echo "Error al escribir en el archivo.";
} else {
	echo "Se escribieron $numBytesEscritos bytes.";
}

fclose($archivo);
?>
```

fflush() es una función que se utiliza para vaciar el buffer de salida de un archivo, es decir, forzar a que se escriban en el archivo todos los datos que están actualmente almacenados en el buffer de salida.

Sintaxis:
fflush(resource $handle): bool

- **`$handle`**: Un recurso de archivo que se obtiene al abrir un archivo con `fopen()` u otra función similar.

Cuando escribes en un archivo usando funciones como `fwrite()`, los datos pueden ser almacenados primero en un buffer de salida antes de ser realmente escritos en el medio de almacenamiento (disco duro, etc.). Esto puede mejorar la eficiencia al reducir el número de operaciones de escritura en disco. Sin embargo, en algunos casos, puede ser necesario asegurarse de que todos los datos almacenados en el buffer se escriban inmediatamente en el archivo. Aquí es donde `fflush()` resulta útil.
Situaciones específicas donde `fflush()` puede ser útil:

1. **Escritura de Datos Críticos**: Si estás escribiendo datos críticos y necesitas asegurarte de que se escriban en el archivo inmediatamente (por ejemplo, en aplicaciones de registro de datos donde cada bit de información es crucial).
2. **Archivos Compartidos**: En situaciones donde múltiples procesos o hilos están accediendo y escribiendo en el mismo archivo, fflush() puede ser útil para minimizar el tiempo en el que los datos escritos se encuentran solo en el buffer y no en el archivo.
3. **Feedback en Tiempo Real**: Si tu script PHP es largo y deseas que el usuario vea los resultados de la escritura en el archivo en tiempo real (por ejemplo, en una interfaz de línea de comandos).

```
<?php
$archivo = fopen("ejemplo.txt", "w");

if ($archivo === false) {
	die("Error al abrir el archivo");
}

fwrite($archivo, "Algunos datos");
fflush($archivo); // Vacía el buffer, asegurando que los datos se escriban en el archivo

fclose($archivo);
?>
```

rewind() es una función en PHP que se utiliza para reiniciar el puntero de un archivo al principio del mismo. Es útil cuando necesitas leer o escribir desde el inicio de un archivo después de haber realizado alguna operación en él.

Sintaxis:
rewind(resource $handle): bool

- **`$handle`**: Un recurso de archivo que se obtiene al abrir un archivo con `fopen()` u otra función similar.

```
<?php
$archivo = fopen("ejemplo.txt", "r+");

// Realiza alguna operación de lectura o escritura aquí...

// Reinicia el puntero al inicio del archivo
if (rewind($archivo)) {
	echo "Puntero del archivo reiniciado con éxito.";
} else {
	echo "Error al reiniciar el puntero del archivo.";
}

fclose($archivo);
?>
```

Ejercicio: Abre un fichero existente, añade un texto al final, devuelve el puntero al inicio y luego muestra todo el contenido.

```php
<?php
$nombreArchivo = "ejemplo.txt";
$textoParaAgregar = "Texto de ejemplo para añadir al fichero existente.\n";

// Abre el archivo en modo de añadir (a+). Esto coloca el puntero al final del archivo.
$archivo = fopen($nombreArchivo, "a+");

if ($archivo === false) {
	die("Error al abrir el archivo.");
}

// Añade texto al final del archivo.
fwrite($archivo, $textoParaAgregar);

// Regresa el puntero al inicio del archivo.
rewind($archivo);

// Lee y muestra el contenido del archivo.
while (!feof($archivo)) {
	$linea = fgets($archivo);
	echo $linea;
}

// Cierra el archivo.
fclose($archivo);
?>
```

stat() en PHP se utiliza para obtener información detallada sobre un archivo especificado. Esta función proporciona varios detalles acerca del archivo, como su tamaño, permisos, última fecha de acceso, última fecha de modificación, entre otros.

Sintaxis:
stat(string $filename): array|false

- **`$filename`**: La ruta al archivo del cual se quiere obtener la información.

La función `stat()` devuelve un array que contiene los siguientes elementos:

- `0` o `dev`: Identificador del dispositivo.
- `1` o `ino`: Número de i-nodo.
- `2` o `mode`: Modo de protección del archivo.
- `3` o `nlink`: Número de enlaces.
- `4` o `uid`: Identificador del usuario propietario.
- `5` o `gid`: Identificador del grupo propietario.
- `6` o `rdev`: Tipo de dispositivo, si es un dispositivo inodo.
- `7` o `size`: Tamaño del archivo en bytes.
- `8` o `atime`: Fecha de último acceso (timestamp).
- `9` o `mtime`: Fecha de última modificación (timestamp).
- `10` o `ctime`: Fecha de último cambio en el i-nodo (timestamp).
- `11` o `blksize`: Tamaño de bloque para el sistema de archivos.
- `12` o `blocks`: Número de bloques asignados.

```
<?php
$nombreArchivo = "ejemplo.txt";

$informacion = stat($nombreArchivo);

if ($informacion !== false) {
	echo "Tamaño del archivo: " . $informacion['size'] . " bytes\n";
	echo "Último acceso: " . date("F d Y H:i:s.", $informacion['atime']);
} else {
	echo "No se pudo obtener la información del archivo.";
}
?>
```

Ejercicio: Crea un script que reciba datos de un formulario (como nombre, apellido y email) y los escriba en un archivo de texto, donde cada línea representa un usuario registrado.

```php
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $nombre = $_POST["nombre"];
    $apellido = $_POST["apellido"];
    $email = $_POST["email"];

    // Validación simple
    /* !filter_var($email, FILTER_VALIDATE_EMAIL esto sirve para validar el email, podéis hacerlo directamente desde el formulario*/
    if (empty($nombre) || empty($apellido) || !filter_var($email, FILTER_VALIDATE_EMAIL)) {
        echo "Por favor, complete el formulario correctamente.";
    } else {
        $archivo = fopen("usuarios.txt", "a+");
        if ($archivo === false) {
            die("Error al abrir el archivo.");
        }

        $linea = "$nombre, $apellido, $email\n";
        if (fwrite($archivo, $linea) === false) {
            echo "Error al escribir en el archivo.";
        } else {
            echo "Usuario registrado con éxito.";
        }

        fclose($archivo);
    }
}
?>
```

Ejercicio: Copiar solamente las líneas pares de un fichero a otro (podéis usar la función file()).

```
<?php
$nombreArchivoOrigen = "archivo_origen.txt";
$nombreArchivoDestino = "archivo_destino.txt";

// Leer todas las líneas del archivo origen
$lineas = file($nombreArchivoOrigen, FILE_IGNORE_NEW_LINES);

// Abrir el archivo destino para escritura
$archivoDestino = fopen($nombreArchivoDestino, "w");

if ($archivoDestino === false) {
	die("Error al abrir el archivo destino.");
}

// Recorrer las líneas y escribir solo las pares
foreach ($lineas as $numeroLinea => $linea) {
	if ($numeroLinea % 2 == 1) {
	fwrite($archivoDestino, $linea . "\n");
	}
}

// Cerrar el archivo destino
fclose($archivoDestino);
?>
```

```
/*
$numeroLinea=0;
while ($numeroLinea < count($lineas)) {
    // Escribir solo las líneas pares (índices impares en el array)
    if ($numeroLinea % 2 == 1) {
        fwrite($archivoDestino, $lineas[$numeroLinea] . "\n");
    }
    $numeroLinea++;
}*/
```

Gestión de errores con try-catch

`try-catch` es una construcción del lenguaje utilizada para manejar excepciones (errores) en tiempo de ejecución de manera controlada. Esta estructura permite que un bloque de código sea probado en busca de errores mientras se está ejecutando, y si se produce un error, se "atrapa" la excepción con un bloque `catch` para manejarlo de manera adecuada.

Sintaxis Básica

```
try {
        // Código que puede lanzar una excepción
} catch (Exception $e) {
        // Código que maneja la excepción
        // $e es el objeto de excepción que contiene detalles del error
}
```

Casos típicos de uso

1. **Manejo de errores de conexión a Bases de Datos**: Cuando te conectas a una base de datos, pueden ocurrir errores como fallos en la conexión o errores en las consultas SQL. Utilizar `try-catch` te permite manejar estos errores de forma elegante.
2. **Manejo de errores en operaciones de archivos**: Al abrir, leer o escribir en archivos, pueden ocurrir errores como permisos insuficientes o archivos no encontrados. `try-catch` puede ser útil para controlar estos errores.
3. **Validaciones Personalizadas**: Puedes lanzar excepciones personalizadas en tus funciones o métodos si una condición específica no se cumple, y luego manejar estas excepciones con `try-catch`.

Ejemplo:

```
try {
        $file = fopen("archivo.txt", "r");
        // Leer o escribir en el archivo
} catch (Exception $e) {
        echo "Error al abrir el archivo: " . $e->getMessage();
}
```