# 7 PHP práctico

Los capítulos previos hemos abordado los elementos del lenguaje PHP. Este capítulo te enseñará cómo realizar algunas tareas comunes pero importantes tareas. Aprenderás las mejores formas de manejar cadenas de forma que logremos código conciso que muestre en el navegador exactamente cómo quieres que se haga. También verás cómo crear o modificar archivos, incluyendo a aquellos cargados por los usuarios.

## Usando `printf`

Ya hemos visto las funciones `print` y `echo`, las cuales simplemente muestran texto en el navegador. Pero una función más poderosa, `printf`, controla el formato de la salida permitiendo que se pongan caracteres especiales de formato en una cadena. Para cada carácter de formato, `printf` espera que pases un argumento que se mostrará usando este mismo formato. Por ejemplo, el siguiente ejemplo usa el especificador %d para mostrar el valor 3 en decimal:

```
Printf("Hay %d ítems en su cesta",3);
```

Si se remplaza %d con %b, el valor 3 se mostrará en binario.

| Especificador | Acción de conversión en el argumento arg | Ejemplo |
|---|---|---|
| % | Muestra un carácter % (no se requiere arg) | % |
| b | Muestra arg como entero binario | 1111011 |
| c | Muestra el carácter ASCII de arg | { |
| d | Muestra arg como un entero decimal con singo | 123 |
| e | Muestra arg usando notación científica | 1.23000e+2 |
| f | Muestra arg como punto flotante | 123.00000 |
| o | Muestra arg como un entero octal | 173 |
| s | Muestra arg como una cadena | 123 |
| u | Muestra arg como un entero decimal sin signo | 123 |
| x | Muestra arg en hexadecimal y minúsculas | 7b |
| X | Muestra arg en hexadecimal y mayúsculas | 7B |

Puedes tener tantos especificadores como quieras en la función `printf`, siempre y cuando los especificadores estén precedidos por el símbolo %. Si olvidas algún argumento, recibirás un error informado que se ha encontrado un paréntesis inesperado.

Un ejemplo práctico de `printf` establece el color en HTML usando valores decimales. Por ejemplo, suponiendo que queremos un color cuyo valor es 65 rojo, 127 verde, 245 azul, pero no quieres convertir estos números a hexadecimal. Tienes la siguiente solución:

```
printf("<span style='color:#%X%X%X'>Hello</span>", 65, 127, 245);
```

El resultado de la salida de este comando sería:

```
<span style='color:#417FF5'>Hello</span>
```

# Ajuste de precisión

No solo puede especificar una conversión de tipo, también puede configurar la precisión del resultado mostrado. Por ejemplo, las cantidades de dinero se muestran usualmente con solo dos dígitos de precisión. Sin embargo, después de hacer un cálculo, un valor puede tener una mayor precisión. Para asegurar que estos valores son almacenados correctamente, pero mostrados con solo dos dígitos de precisión, puedes insertar la cadena ".2" entre el símbolo % y el especificador de conversión:

```
printf("El resultado es: $%.2f", 123.42 / 12);
```

La salida de este comando es la siguiente:

```
El resultado es: $10.29
```

Se puede controlar aún más que eso, y especificar un desplazamiento con espacios o ceros poniendo antes del especificador ciertos valores.

*Ejemplo 7-1. Ajuste de precisión.*

```
<?php
    echo "<pre>";

    //desplazar 15 espacios
    printf("El resultado es $%15f\n", 123.42 / 12);

    //desplazar 15 espacios, rellenar con ceros
    printf("El resultado es $%15f\n", 123.42 / 12);

    //desplazar 15 espacios, 2 lugares decimales de precisión
    printf("El resultado es $%15.2f\n", 123.42 / 12);

    //desplazar 15 espacios, 2 decimales de precisión, comp con ceros
    printf("El resultado es $%015.2f\n", 123.42 / 12);

    //desplazar 15 espacios, 2 decimales de precisión, completado con #
    printf("El resultado es $%'#15.2f\n", 123.42 / 12);
?>
```

| Inicio de conversión | Carácter de relleno | # de caract relleno | Mostrar precisión | Especif de convers | Ejemplo |
|---|---|---|---|---|---|
| % | | 15 | | f | 10.285000 |
| % | 0 | 15 | .2 | f | 000000000010.29 |
| % | '# | 15 | .4 | f | ########10.2850 |

# Relleno de cadenas

Puedes también rellenar cadenas a longitudes requeridas, seleccionar diferentes caracteres de relleno y aun elegir justificación derecha e izquierda.

*Ejemplo 7-2. Relleno de cadenas.*

```
<?php
    echo "<pre>";

    $h = 'Rasmus';

    printf("[%s]\n",        $h); //salida standard de string
    printf("[%12s]\n",      $h); //justificación derecha
    printf("[%-12s]\n",     $h); //justificación izquierda con espacios
    printf("[%012s]\n",     $h); //relleno con ceros
    printf("[%'#12s]\n",    $h); //Caracter de relleno #

    $d = 'Rasmus Ledorf';

    printf("[%12.8s]\n",    $d); //Justif derecha, corte de 8 caracteres
    printf("[%-12.12s]\n",  $d); //Justif izqu, corte de 12 caracteres
    printf("[%-'@12.10s]\n",$d); //Justif. izqu, relleno con @, corte 10
car
?>
```

| Inicio de conversión | Justific | Carácter de relleno | # de caract relleno | corte | Espec de conv | Ejemplo |
|---|---|---|---|---|---|---|
| % | | | | | S | [Rasmus] |
| % | - | | 10 | .2 | S | [Rasmus ] |
| % | | '# | 8 | .4 | s | [####Rasmus] |

### Usando `sprintf`

A veces, no necesitas que el resultado de una conversión de cadena sea mostrado, pero necesitas usarlo en tu código. Con `sprintf` puedes enviar la salida a otra variable en lugar del navegador.

```
$hexstring = sprintf("%X%X%X", 65, 127, 245);
```

# Funciones de fecha y hora

Para mantener un seguimiento de la fecha y la hora, PHP usa marcas de tiempo de Unix, los cuales son el número de segundos pasados desde el 01 de enero de 1970. Para determinar la marca de tiempo actual, puedes usar la función:

```
echo time()
```

Debido a que el valor es almacenado como segundos, para obtener la marca de tiempo para la siguiente semana, podrías usar lo siguiente, agregar 7 días x 24 horas x 60 minutos x 60 segundos para el valor devuelto:

```
echo time() + 7*24*60*60;
```

Si quieres crear una marca de tiempo para una fecha dada, puedes usar la función mktime. Esta salida es la marca de tiempo para el primer segundo del primer minuto de la primera hora del primer día del año 2000:

```
echo mktime(0, 0, 0, 1, 1, 2000);
```

Para mostrar el formato, use la función date, la cual soporta una plétora de opciones de formato que permiten mostrar la fecha de la forma que se desee.

```
date($format, $timestamp);
```

el parámetro `$format` debe ser una cadena conteniendo especificadores, y `$timestamp` debe ser una marca de tiempo Unix. Para la lista completa de especificadores vea la documentación. El siguiente comando mostrará la fecha en el formato "Thursday July 6th, 2017 – 1:38pm"

```
echo date("l F jS, Y – g:ia", time());
```

| Formato | Descripción | Valor devuelto |
|---|---|---|
| **d** | Día del mes con ceros | 01 a 31 |
| **D** | Día de la semana | Mon a Sun |
| **j** | Dia del mes si ceros | 1 a 31 |
| **l** | Día de la semana con nombre | Sunday a Saturday |
| **N** | Día de la semana con numero | 1 a 7 |
| **S** | Sufijo del día de la semana | St, nd, rd, o th |
| **w** | Día de la semana desde Lunes | 0 a 6 |
| **z** | Día del año | 0 a 365 |
| | | |
| **W** | Número de semana del año | 01 a 52 |

| | | |
|---|---|---|
| **F** | Nombre del mes | January a December |
| **m** | Número del mes con ceros | 01 a 12 |
| **M** | Nombre del mes en tres letras | Jan a Dec |
| **n** | Número del mes sin ceros | 1 a 12 |
| **t** | Número de días de un mes | 28 a 31 |
| | | |
| **L** | Año bisiesto | 1, 0 |
| **y** | Año de dos dígitos | 00 a 99 |
| **Y** | Año de 4 dígitos | 0000 a 9999 |
| | | |
| **a** | Antes o después de medio día | am o pm |
| **A** | Antes o después de medio día Mayusc | AM o PM |
| **g** | Hora del día, formato 12 horas | 1 a 12 |
| **G** | Hora del día, formato 24 horas | 0 a 23 |
| **h** | Hora del día, formato 12 h con ceros | 01 a 12 |
| **H** | Hora del día, formato 24 h con ceros | 00 a 23 |
| **i** | Minutos con ceros | 00 a 59 |
| **s** | Segundos con ceros | 00 a 59 |

## Constantes de tiempo

Hay constantes útiles que puedes usar con el comando date para devolver la fecha de forma específica. Por ejemplo, date(`DATE_RSS`) devuelve la fecha actual y el tiempo en un formato válido para un feed `RSS`. Algunas de las constantes más usadas son las siguientes:

```
DATE_ATOM
DATE_COOKIE
DATE_RSS
DATE_W3C
```

## Usando `checkdate`

Has visto cómo mostrar una fecha válida en variedad de formatos. ¿Pero cómo puedes verificar si un usuario ha enviado una fecha válida al programa? La respuesta es pasar el mes, el día y el año a la función `checkdate`, la cual retorna un valor `TRUE` si la fecha es válida, o `FALSE` si no lo es.

Por ejemplo, si la entrada es el 31 de setiembre de cualquier año, esto sería una fecha inválida.

*Ejemplo 7-3. Verificar la validez de la fecha.*

```
<?php
```

```
    $mes    = 9;
    $dia    = 31;
    $año    = 2022;

    if (checkdate($mes, $dia, $año)) echo "Fecha es inválida";
    else echo "Fecha es inválida";
?>
```

# Manejo de archivos

Poderoso como es, MySQL no es la única forma de almacenar datos en un servidor web. Algunas veces puede ser más rápido y conveniente acceder directamente a los archivos en el disco duro. Hay casos en los cuales necesitamos hacer esto cuando modificamos imágenes tales como imágenes de avatar subidas por los usuarios, o archivos de log de lo que deseas procesar.

Primero, debes tener en cuenta que, si escribes código que puede ser usado en varias instalaciones de PHP, no hay forma de saber si estos sistemas son case sensitivos. Por ejemplo, Windows y macOS no son case sensitivos, pero Linux y Unix sí lo son. Por tanto, deberías asumir siempre que el sistema es case sensitivo y ceñirte a convenciones como mantener los nombres de los archivos e minúsculas.

## Verificando si un archivo existe

Para determinar si un archivo ya existe, puedes usar la función `file_exists`, la cual devuelve `TRUE` o `FALSE`:

```
If(file_exists("testfile.txt")) echo "El archive existe";
```

## Creando un archivo

En este punto, `testfile.txt` no existe, entonces vamos a crearlo y escribir unas pocas líneas en este.

*Ejemplo 7-4. Creando un archivo de texto simple*

```
<?php
    $fh = fopen("testfile.txt", 'w') or die("No se pudo crear el
archivo");
    $text = <<<_END
    Linea 1
    Linea 2
    Linea 3
    _END;

    fwrite($fh, $text) or die("No se pudo escribir el archivo");
```

```
    fclose($fh);
    echo "Archivo 'testfile.txt' escrito satisfactoriamente";
?>
```

Un programa podría llamar a la función die, el archivo abierto será cerrado automáticamente cerrado cuando finalice el programa.

Cuando corres esto en un navegador, y todo va bien, recibirás el mensaje de que fue todo satisfactoriamente. Si recibes un mensaje de error, tu disco podría estar lleno o no tienes permisos para crear o escribir en el archivo, en este caso deberías modificar los atributos de la carpeta de destino.

| Modo | Acción | Descripción |
|---|---|---|
| **'r'** | Leer desde el comienzo de un archivo | Abre para lectura solamente, pone el puntero al principio del archivo. Devuelve falso si el archivo no existe |
| **'r+'** | Leer desde el comienzo y permitir escritura | Abre para lectura y escritura, devuelve falso si el archivo no existe |
| **'w'** | Escribir desde el comienzo archivo y truncar | Abre para escritura solamente, pone el puntero al inicio. Si el archivo no existe intenta crearlo |
| **'w+'** | Escribir desde el comienzo de un archivo y permitir lectura | Abre el archivo para lectura escritura. Pone el puntero al principio Si el archivo no existe intenta crearlos |
| **'a'** | agregar al final del archivo | Abre para escritura solamente, pone el puntero al final del archivo, si no existe, intenta crearlo |
| **'a+'** | Agregar al final de un archivo y permitir lectura | Abre para lectura y escritura, pone el puntero al final del archivo, si el archivo no existe intenta crearlo |

## Leyendo archivos

La forma más fácil de leer desde un archivo de texto es grabar una línea entera mediante `fgets`

*Ejemplo 7-5. Leer un archivo con fgets*

```
<?php
    $fh = fopen("testfile.txt",'r') or
    die("El archivo no existe o no tienes permisos");

    $linea = fgets($fh);
    fclose($fh);
    echo $linea;
?>
```

Si creaste el archivo en el ejemplo 7-4, obtendrás la primera línea. Puedes obtener muchas líneas o porciones de líneas mediante la función `fread`.

*Ejemplo 7-6. Leyendo un archivo con fread*

```
<?php
    $fh = fopen("testfile.txt", 'r') or
    die("El archivo no existe o no tienes permisos");

    $texto = fread($fh, 3);
    fclose($fh);

    echo $texto;
?>
```

## Copiando archivos

Vamos a usar la función `copy` para crear un clon de testfile.txt. Guardarlo como `copyfile.php`, y entonces llamar al programa en el navegador.

*Ejemplo 7-7. Copiando un archivo.*

```
<?php
    copy('testfile.txt', 'testfile2.txt') or die ("No se puede copiar");
    echo "El archivo fue copiado satisfactoriamente";
?>
```

Si tu revisas tu folder otra vez, veras que tú no tenías el nuevo archivo testfile2.txt en éste. Por esta forma, si no quieres que tu programa termine con un intento fallido de copia, puedes tratar de cambiar la sintaxis. Utilizando el operador `!` (NOT) colocado en una expresión, si se aplica el operador NOT es como decir "si no es posible copiar…"

*Ejemplo 7-8. Sintaxis alterna para la copia de archivo.*

```
<?php
    if (!copy('testfile.txt', 'testfile2.txt')) echo "No se puede copiar";
    else echo "El archivo se ha copiado";
?>
```

## Moviendo un archivo

Para mover un archivo, renombre el archivo con la función `rename`.

*Ejemplo 7-9. Moviendo un archivo*

```
<?php
    if (!rename('testfile2.txt', 'testfile2.new'))
        echo "No se puedo renombrar";
    else echo "Archivo renombrado exitosamente";
?>
```

## Borrando un archivo

Borrando un archivo es justo una forma de usar la función `unlink` para removerlo desde el sistema de archivo.

*Ejemplo 7-10. Borrando un archivo*

```
<?php
    if (!unlink('testfile2.new')) echo "No se pudo borrar el archivo";
    else echo "Archivo borrado satisfactoriamente";
?>
```

## Actualizando archivos

A veces queremos agregar más datos a un archivo guardado, lo que se puede hacer de muchas formas. Puedes usar uno de los modos de adicionar y escribir, o puedes simplemente abrir un archivo para leerlo y escribirlo con uno de los otros modos que soporta escritura, y mover el puntero al lugar correcto en el archivo que deseas leer o escribir.

El puntero de archivo es la posición del archivo en la cual el siguiente acceso al archivo tendrá lugar que puede ser una lectura o escritura. No es lo mismo que el manejador de archivo, el cual contiene detalles sobre el archivo que está siendo accedido.

*Ejemplo 7-11. Actualizando un archivo*

```
<?php
    $fh = fopen("testfile.txt", 'r+') or die("Falló la apertura del
archivo");
    $texto = fgets($fh);
    fseek($fh, 0, SEEK_END);
    fwrite($fh, "$texto") or die("No se puede escribir el archivo");
    fclose($fh);

    echo "Archivo actualizado exitosamente";
?>
```

Este programa abre el archivo testfile.txt para lectura y escritura, lo cual pone el apuntador de archivo al inicio. Si entonces utilizamos la función `fgest` para leer una línea simple desde el archivo. Después de esto, la función `fseek` es llamada para mover el puntero del archivo al final del archivo, y escribe esta línea al final Luego cierra el archivo.

Adicionalmente al manejador de archivo `$fh`, la función `fseek` recibe otros dos parámetros, `SEEK_END` le dice a la función que mueva el apuntador de archivo al final del archivo, y 0 le dice cuántas posiciones hacia atrás debe moverse a partir de este punto. En el caso del ejemplo, el valor 0 se usa por que se requiere que el apuntador apunte al final del archivo.

Hay otras dos opciones de búsqueda disponibles para la función `fseek`: `SEEK_SET` y `SEEK_CUR`. La opción `SEEK_SET` le dice al apuntador de la posición exacta dada por el parámetro precedente.

```
fseek($fh, 18, SEEK_SET);
```

`SEEK_CUR` establece el apuntador en la posición actual más un desplazamiento dado. Así, si el apuntador estaba en la posición 18, la siguiente llamada lo movería a la posición 23:

```
fseek($fh, 5, SEEK_CUR);
```

## Bloqueando archivos para acceso múltiple

Los programas web son llamados a menudo por muchos usuarios. Si más de uno trata de escribir un archivo simultáneamente, puede corromperse. Para manejar usuarios simultáneos, debes usar la función del bloqueo de archivos `flock`. Esta función pone en cola todas las otras solicitudes para acceder al archivo hasta que tu programa libere el bloqueo. Por lo que, si alguno de tus programas usa acceso de escritura en archivos que pueden ser accedidos concurrentemente por múltiples usuarios, debes agregarles un bloqueo de archivos.

*Ejemplo 7-12. Actualizando un archivo con bloqueo de archivos*

```
<?php
    $fh = fopen("testfile.txt", 'r+') or die("Falló la apertura del
archivo");
    $texto = fgets($fh);

    if (flock($fh, LOCK_EX))
    {
        fseek($fh, 0, SEEK_END);
        fwrite($fh, "$texto") or die("No se puede escribir el archivo");
        flock($fh, LOCK_UN);
    }
```

```
    fclose($fh);
    echo "Archivo actualizado exitosamente";
?>
```

Hay un truco para bloquear un archivo para preservar el mejor tiempo de respuesta para los visitantes de tu site: hacerlo exactamente antes de hacer un cambio en un archivo y desbloquearlo inmediatamente después.

## Leer un archivo entero

Una función útil para leer un archivo completo sin tener que usar manejadores de archivos es `file_get_contents`.

*Ejemplo 7-13. Usando `file_get_contents`*

```
<?php
    echo "<pre>";
    echo file_get_contents("testfile.txt");
    echo "</pre>";
?>
```

Esta función se puede usar también para obtener un archivo desde internet.

*Ejemplo 7-14. Obteniendo un archivo de otra URL*

```
<?php
    echo "<pre>";
    echo file_get_contents("http://unajma.edu.pe");
    echo "</pre>";
?>
```

## Subiendo archivos

Subir archivos a un servidor es un asunto que parece desalentador para mucha gente, pero actualmente podría ser mucho más fácil. Todo lo que necesitas es subir un archivo desde un formulario es elegir un tipo de codificación especial llamada `multipart/form-data`, y tu navegador manejará el resto. Para ver cómo trabaja esto, escribe el programa del Ejemplo 7-15 y guardarlo como *upload.php*. cuando corre este, verás un formulario en tu navegador que te permite subir un archivo de tu elección.

*Ejemplo 7-15. Cargador de imágenes*

```
<?php
```

```
    echo <<<_END
    <html><head><title>PHP Form Upload</title></head><body>
    <form method='post' action='upload.php' enctype='multipart/form-data'>
    Select File: <input type='file' name='filename' size='10'>
    <input type='submit' value='Upload'>
    </form>
    _END;

    if ($_FILES)
    {
        $name = $_FILES['filename']['name'];
        move_uploaded_file($_FILES['filename']['tmp_name'], $name);
        echo "Uploaded image '$name'<br><img src='$name'>";
    }

    echo "</body></html>";
?>
```

Vamos a examinar un poco el programa. La primera línea del echo multilínea inicia un documento HTML, muestra el título, e inicia el cuerpo del documento.

Seguidamente viene el formulario, el cual usa el método `POST` para enviar los datos al script upload.php (el mismo programa), y le dice al navegador que los datos enviados deben ser codificados como tipo de contenido `'multipart/form-data'`.

Con el formulario configurado, las siguientes líneas muestran el díalogo Select File: y entonces requiere dos entradas. La primera es para el archivo; esta usa un tipo de entrada file, un nombre o nombre de archivo y un campo de entrada con un ancho de 10 caracteres. La seguda entrada es un botón de `submit` a quien se le da la etiqueta `Upload`.

Este corto programa muestra una técnica común en programación web en la cual un solo programa es llamado dos veces: una cuando el usuario visita la página y la otra cuando presiona el botón `submit`.

El código PHP que recibe la data cargada es muy simple, porque todos los archivos cargados son puestos en un array asociativo del sistema `$_FILES`. Por lo que un chequeo rápido para saber si `$_FILES` contiene algo es suficiente para determinar si el usuario ha cargado un archivo. Esto se hace en la sentencia `if($_FILES).`

La primera vez que el usuario visita la página, `$_FILES` está vacío, por tanto, el programa salta este bloque de código. Cuando el usuario carga un archivo, el programa corre otra vez y descubre un elemento en el array `$_FILES`.

Una vez que el programa ve que un archivo ha sido cargado, se lee el nombre actual desde la computadora, se recibe y se coloca en la variable `$name`. Ahora

se debe mover el archivo dese la ubicación temporal en la cual PHP ha almacenado este a uno más permanente. Hacemos esto usando la función `move_upladed_file`, pasándole el nombre original del archivo, el cual se guarda en el directorio actual.

Finalmente, la imagen cargada se muestra dentro de una etiqueta IMG,

## Usando $_FILES

Cinco cosas se almacenan en el array $_FILES cuando un archivo es cargado

| Elemento del array | Contenido |
|---|---|
| `$_FILES['file']['name']` | El nombre del archivo cargado |
| `$_FILES['file']['type]` | El tipo de contenido |
| `$_FILES['file']['size]` | El tamaño de archivo en bytes |
| `$_FILES['file']['tmp_name']` | El nombre del archivo temporal almacenado en el servidor |
| `$_FILES['file']['error']` | El código de error resultante de la carga del archivo |

El tipo de contenido usado se conoce como MIME (Multipurpose Internet Mail Extension), se conocen como internet media tiypes. Los tipos más frecuentes de tipos que aparece en `$_FILES['file']['type']`.

```
application/pdf
image/gif
multipart/form-data
text/xml
application/zip
image/jpeg
text/css
video/mpeg
audio/mpeg
image/png
text/html
video/mp4
audio/x-wav
image/tiff
text/plain
video/quicktime
```

## Validación

Una cosa que se tiene que verificar es si el archivo recibido es del tipo correcto.

*Ejemplo 7-16. Una versión más segura de upload.php*

```
<?php
    echo <<<_END
    <html><head><title>PHP Form Upload</title></head><body>
    <form method='post' action='upload2.php' enctype='multipart/form-
data'>
    Select File: <input type='file' name='filename' size='10'>
    <input type='submit' value='Upload'>
    </form>
    _END;

    if ($_FILES)
    {
        $name = $_FILES['filename']['name'];

        switch ($_FILES['filename']['type'])
        {
            case 'image/jpeg':  $ext = 'jpg'; break;
            case 'image/gif':   $ext = 'jpg'; break;
            case 'image/png':   $ext = 'png'; break;
            case 'image/tiff':  $ext = 'tif'; break;
            default:            $ext = ''; break;
        }

        if ($ext)
        {
            $n = "image.$ext";
            move_uploaded_file($_FILES['filename']['tmp_name'], $n);
            echo "Uploaded image '$name' as '$n':<br>";
            echo "<img src=$n>";
        }
        else echo "'$name' no es un archivo de imagen";
    }
    else echo "No se ha cargando una imagen";

    echo "</body></html>";
?>
```

# Llamadas al sistema

A veces PHP no tiene una función que necesitas para realizar una acción determinado, pero el sistema operativo sí. En tales casos puedes llamar a una tarea del sistema con `exec`.

Por ejemplo, para ver rápidamente el contenido de un directorio, aunque depende si está corriendo en Linux o Windows.

*Ejemplo 7-17. Ejecutando un comando del sistema*

```
<?php
    $cmd = "dir";

    exec(escapeshellcmd($cmd), $output, $status);

    if ($status) echo "Fallo el comando Exec";
    else
    {
        echo "<pre>";
        foreach($output as $line) echo htmlspecialchars("$line\n");
        echo "</pre>";
    }
?>
```