Chapter 5

Working with Files and Directories

PHP Programming with MySQL
2nd Edition

Objectives

In this chapter, you will:

- Understand file type and permissions
- Work with directories
- Write data to files
- Read data from files
- Open and close a file stream
- Manage files and directories

Understanding File Types and Permissions

- **File types** affect how information is stored in files and retrieved from them
- **File permissions** determine the actions that a specific user can and cannot perform on a file

Understanding File Types

- A **binary file** is a series of characters or bytes for which PHP attaches no special meaning
  - Structure is determined by the application that reads or writes to the file
- A **text file** has only printable characters and a small set of control or formatting characters
  - Text files translate the end-of-line character sequences such as `\n` or `\r\n` to carriage returns

Understanding File Types (continued)

Escape Sequence	Meaning	Byte Value		
		Decimal	Octal	Hexadecimal
\t	Horizontal tab	9	011	09
\r	Line feed	10	012	0A
\v	Vertical tab	11	013	0B
\f	Form feed	12	014	0C
\n	Carriage return	13	015	0D

Table 5-1 Control characters in a text file

Understanding File Types (continued)

- Different operating systems use different escape sequences to identify the end of a line:
  - Use the `\n` sequence to end a line on a UNIX/Linux operating system
  - Use the `\n\r` sequence to end a line on a Windows operating system
  - Use the `\r` sequence to end a line on a Macintosh operating system.

Understanding File Types (continued)

- Scripts written in a UNIX/Linux text editor display differently when opened in a Windows-based text editor

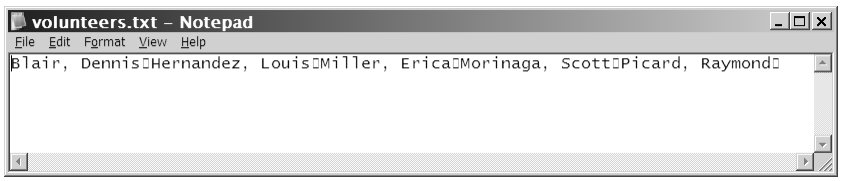


Figure 5-1 Volunteer registration form

Working with File Permissions

- Files and directories have three levels of access:
  - User
  - Group
  - Other
- The three typical permissions for files and directories are:
  - Read (r)
  - Write (w)
  - Execute (x)

Working with File Permissions (continued)

- File permissions are calculated using a four-digit octal (base 8) value
  - Octal values encode three bits per digit, which matches the three permission bits per level of access
  - The first digit is always 0
  - To assign more than one value to an access level, add the values of the permissions together

Working with File Permissions (continued)

Permissions	First Digit (Leftmost) Always 0	Second Digit User (u)	Third Digit Group (g)	Fourth Digit (Rightmost) Other (o)
Read (r)	0	4	4	4
Write (w)	0	2	2	2
Execute (x)	0	1	1	1

Table 5-2 Octal values for the *mode* parameter of the `chmod()` function

Working with File Permissions (continued)

- The `chmod()` function is used to change the permissions or modes of a file or directory
- The syntax for the `chmod()` function is

```
chmod($filename, $mode)
```

- Where `$filename` is the name of the file to change and `$mode` is an integer specifying the permissions for the file

Checking Permissions

- The `fileperms()` function is used to read permissions associated with a file
  - The `fileperms()` function takes one argument and returns an integer bitmap of the permissions associated with the file
  - Permissions can be extracted using the arithmetic modulus operator with an octal value of 01000
- The `dococt()` function converts a decimal value to an octal value

Reading Directories

- The following table lists the PHP functions that read the names of files and directories

Function	Description
`chdir(directory)`	Changes to the specified directory
`chroot(directory)`	Changes the root directory of the current process to the specified directory
`closedir(handle)`	Closes a directory handle
`getcwd()`	Gets the current working directory
`opendir(directory)`	Opens a handle to the specified directory
`readdir(handle)`	Reads a file or directory name from the specified directory handle
`rewinddir(handle)`	Resets the directory pointer to the beginning of the directory
`scandir(directory[, sort])`	Returns an indexed array containing the names of files and directories in the specified directory

Table 5-3 PHP directory functions

Reading Directories (continued)

- The `opendir()` function is used to iterate through entries in a directory
- A **handle** is a special type of variable that PHP used to represent a resource such as a file or a directory
- The `readdir()` function returns the file and directory names of an open directory
- The **directory pointer** is a special type of variable that refers to the currently selected record in a directory listing

Reading Directories (continued)

- The `closedir()` function is used to close the directory handle
- The following code lists the files in the open directory and closes the directory.

```
$Dir = "/var/html/uploads";
$DirOpen = opendir($Dir);
while ($CurFile = readdir($DirOpen)) {
      echo $CurFile . "<br />\n";
}
closedir($DirOpen);
```


Reading Directories (continued)

- The following Figure shows the directory listing for three files: kitten.jpg, polarbear.jpg, and gorilla.gif

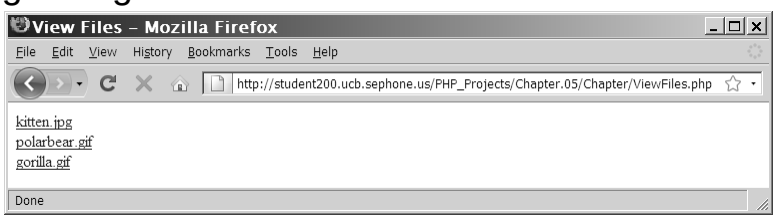


Figure 5-2 Listing of the "files" subdirectory using the `opendir()`, `readdir()`, and `closedir()` functions

Reading Directories (continued)

- The `scandir()` function returns the names of the entries in a directory to an array sorted in ascending alphabetical order

```
$Dir = "/var/html/uploads";
$DirEntries = scandir($Dir);
foreach ($DirEntries as $Entry) {
     echo $Entry . "<br />\n";
}
```


Reading Directories (continued)

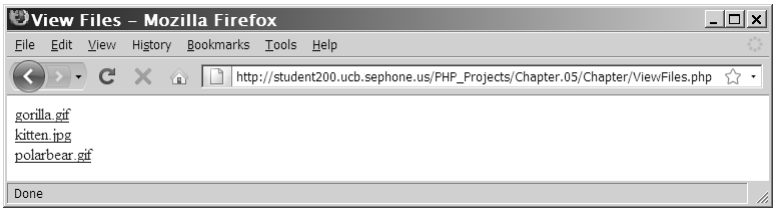


Figure 5-3 Listing of the "files" subdirectory using the `scandir()` function

Creating Directories

- The `mkdir()` function creates a new directory
- To create a new directory within the current directory:
  - Pass just the name of the directory you want to create to the `mkdir()` function

```
mkdir("volunteers");
```


Creating Directories (continued)

- To create a new directory in a location other than the current directory:
  - Use a relative or an absolute path

```
mkdir("../event");

mkdir("/bin/PHP/utilities");
```

Creating Directories (continued)

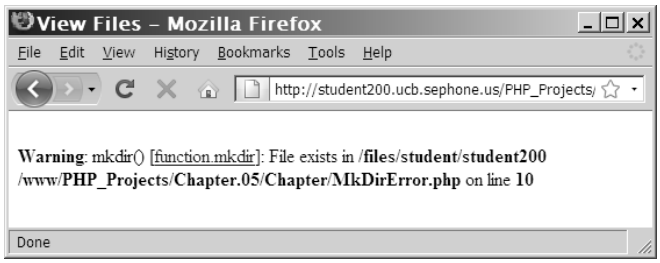


Figure 5-4 Warning that appears if a directory already exists

Obtaining File and Directory Information

Function	Description
`file_exists(filename)`	Determines whether a file or directory exists
`is_dir(filename)`	Determines whether a filename specifies a directory
`is_executable(filename)`	Determines whether a file is executable
`is_file(filename)`	Determines whether a filename specifies a regular file
`is_link(filename)`	Determines whether a filename specifies a symbolic link
`is_readable(filename)`	Determines whether a file is readable
`is_writable(filename)` or `is_writeable(filename)`	Determines whether a file is writable

Table 5-4 PHP file and directory status functions

Obtaining File and Directory Information (continued)

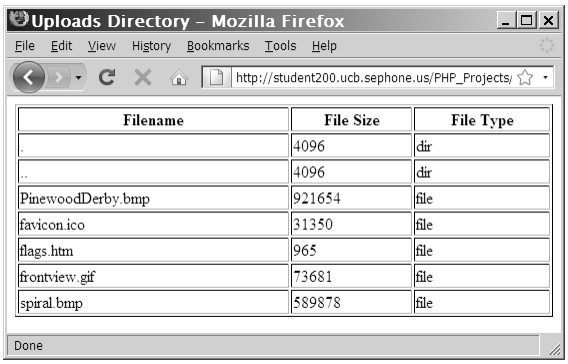
Function	Description
`fileatime(filename)`	Returns the last time the file was accessed
`filectime(filename)`	Returns the last time the file information was modified
`filemtime(filename)`	Returns the last time the data in a file was modified
`fileowner(filename)`	Returns the name of the file's owner
`filesize(filename)`	Returns the size of the file in bytes
`filetype(filename)`	Returns the file type

Table 5-5 Common file and directory information functions

Obtaining File and Directory Information (continued)

```
$Dir = "/var/html/uploads";
if (is_dir($Dir)) {
   echo "<table border='1' width='100%'>\n";
   echo "<tr><th>Filename</th><th>File Size</th>
        <th>File Type</th></tr>\n";
   $DirEntries = scandir($Dir);
   foreach ($DirEntries as $Entry) {
     $EntryFullName = $Dir . "/" . $Entry;
      echo "<tr><td>" . htmlentities($Entry) . "</td><td>" .
      filesize($EntryFullName) . "</td><td>" .
      filetype($EntryFullName) . "</td></tr>\n";
   }
    echo "</table>\n";
}
else
   echo "<p>The directory " . htmlentities($Dir) . " does not
   exist.</p>";
```

Obtaining File and Directory Information (continued)



Filename	File Size	File Type
.	4096	dir
..	4096	dir
PinewoodDerby.bmp	921654	file
favicon.ico	31350	file
flags.htm	965	file
frontview.gif	73681	file
spiral.bmp	589878	file

Figure 5-5 Output of script with file and directory information functions

Writing an Entire File

- PHP supports two basic functions for writing data to text files:
  - `file_put_contents()` function writes or appends a text string to a file and returns the number of bytes written to the file
  - `fwrite()` function incrementally writes data to a text file

Writing an Entire File (continued)

- The `file_put_contents()` function writes or appends a text string to a file
- The syntax for the `file_put_contents()` function is:

```
file_put_contents (filename, string[, options])
```


Writing an Entire File (continued)

```
$EventVolunteers = "Blair, Dennis\n";
$EventVolunteers .= "Hernandez, Louis\n";
$EventVolunteers .= "Miller, Erica\n";
$EventVolunteers .= "Morinaga, Scott\n";
$EventVolunteers .= "Picard, Raymond\n";
$VolunteersFile = "volunteers.txt";
file_put_contents($VolunteersFile,
  $EventVolunteers);
```

Writing an Entire File (continued)

- If no data was written to the file, the function returns a value of `0`
- Use the return value to determine whether data was successfully written to the file

```
if (file_put_contents($VolunteersFile, $EventVolunteers) > 0)
        echo "<p>Data was successfully written to the
              $VolunteersFile file.</p>";
else
        echo "<p>No data was written to the $VolunteersFile file.</p>";
```


Writing an Entire File (continued))

- The `FILE_USE_INCLUDE_PATH` constant searches for the specified filename in the path that is assigned to the `include_path` directive in your php.ini configuration file
- The `FILE_APPEND` constant appends data to any existing contents in the specified filename instead of overwriting it

Reading an Entire File

Function	Description
`file(filename[, use_include_path])`	Reads the contents of a file into an indexed array
`file_get_contents(filename[,options])`	Reads the contents of a file into a string
`readfile(filename[,use_include_path])`	Displays the contents of a file

Table 5-8 PHP functions that read the entire contents of a text file

Reading an Entire File (continued)

- The `file_get_contents()` function reads the entire contents of a file into a string

```
$DailyForecast = "<p><strong>San Francisco daily weather
forecast</strong>: Today: Partly cloudy. Highs from the 60s to
mid 70s. West winds 5 to 15 mph. Tonight: Increasing clouds. Lows
in the mid 40s to lower 50s. West winds 5 to 10 mph.</p>";
file_put_contents("sfweather.txt", $DailyForecast);
```

```
$SFWeather = file_get_contents("sfweather.txt");
echo $SFWeather;
```

Reading an Entire File (continued)

- The `readfile()` function displays the contents of a text file along with the file size to a Web browser

```
readfile("sfweather.txt");
```


Reading an Entire File (continued)

- The `file()` function reads the entire contents of a file into an indexed array
- Automatically recognizes whether the lines in a text file end in `\n`, `\r`, or `\r\n`

```
$January = " 61, 42, 48\n ";
$January .= "62, 41, 49\n ";
$January .= " 62, 41, 49\n ";
$January .= " 64, 40, 51\n ";
$January .= " 69, 44, 55\n ";
$January .= " 69, 45, 52\n ";
$January .= " 67, 46, 54\n ";
file_put_contents("sfjanaverages.txt", $January);
```

Reading an Entire File (continued)

```
$JanuaryTemps = file("sfjanaverages.txt");
for ($i=0; $i<count($JanuaryTemps); ++$i) {
      $CurDay = explode(", ", $JanuaryTemps[$i]);
      echo "<p><strong>Day " . ($i + 1) . "</strong><br />";
      echo "High: {$CurDay[0]}<br />";
      echo "Low: {$CurDay[1]}<br />";
      echo "Mean: {$CurDay[2]}</p>";
}
```


Reading an Entire File (continued)

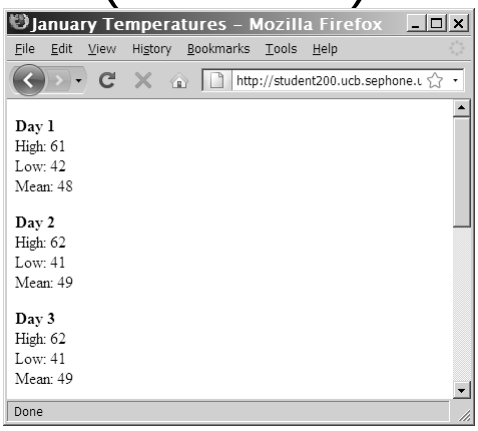


Figure 5-13 Output of individual lines in a text file

Opening and Closing File Streams

- A **stream** is a channel used for accessing a resource that you can read from and write to
- The **input stream** reads data from a resource (such as a file)
- The **output stream** writes data to a resource
  1. Open the file stream with the `fopen()` function
  2. Write data to or read data from the file stream
  3. Close the file stream with the `fclose()` function

Opening a File Stream

- A **handle** is a special type of variable that PHP uses to represent a resource such as a file
- The `fopen()` function opens a handle to a file stream
- The syntax for the `fopen()` function is:

```
open_file =  fopen("text file", "mode");
```

- A **file pointer** is a special type of variable that refers to the currently selected line or character in a file

Opening a File Stream (continued)

Argument	Description
a	Opens the specified file for writing only and places the file pointer at the end of the file; attempts to create the file if it doesn't exist
a+	Opens the specified file for reading and writing and places the file pointer at the end of the file; attempts to create the file if it doesn't exist
r	Opens the specified file for reading only and places the file pointer at the beginning of the file
r+	Opens the specified file for reading and writing and places the file pointer at the beginning of the file
w	Opens the specified file for writing only and deletes any existing content in the file; attempts to create the file if it doesn't exist
w+	Opens the specified file for reading and writing and deletes any existing content in the file; attempts to create the file if it doesn't exist
x	Creates and opens the specified file for writing only; returns FALSE if the file already exists
x+	Creates and opens the specified file for reading and writing; returns FALSE if the file already exists

Table 5-9 Valid *method* argument values of the `fopen()` function

Closing a File Stream

- Use the `fclose` function when finished working with a file stream to save space in memory
- Use the statement `fclose($handle);` to ensure that the file doesn't keep taking up space in your computer's memory and allow other processes to read to and write from the file

Managing Files and Directories

- PHP can be used to manage files and the directories that store them
- Among the file directory and management tasks for files and directories are
  - Copying
  - Moving
  - Renaming
  - Deleting

Copying and Moving Files

- Use the `copy()` function to copy a file with PHP
- The function returns a value of `TRUE` if it is successful or `FALSE` if it is not
- The syntax for the `copy()` function is:

```
copy(source, destination)
```

- For the *source* and *destination* arguments:
  - Include just the name of a file to make a copy in the current directory, or
  - Specify the entire path for each argument

Copying and Moving Files (continued)

```
if (file_exists("sfweather.txt")) {
         if(is_dir("history")) {
              if (copy("sfweather.txt",
                     "history\\sfweather01-27-2006.txt"))
                     echo "<p>File copied successfully.</p>";
              else
                     echo "<p>Unable to copy the file!</p>";
       }
       else
              echo ("<p>The directory does not exist!</p>");
}
else
       echo ("<p>The file does not exist!</p>");
```


Renaming Files and Directories

- Use the `rename()` function to rename a file or directory with PHP
- The `rename()` function returns a value of true if it is successful or false if it is not
- The syntax for the `rename()` function is:

  `rename(`*`old_name, new_name`*`)`

Removing Files and Directories

- Use the `unlink()` function to delete files and the `rmdir()` function to delete directories
- Pass the name of a file to the `unlink()` function and the name of a directory to the `rmdir()` function
- Both functions return a value of true if successful or false if not
- Use the `file_exists()` function to determine whether a file or directory name exists before you attempt to delete it

Summary

- In PHP, a file can be one of two types: binary or text
- A **binary file** is a series of characters or bytes for which PHP attaches no special meaning
- A **text file** has only printable characters and a small set of control of formatting characters
- A text file translates the end-of-line character sequences in code display
- The UNIX/Linux platforms end a line with the `\n` sequence

Summary (continued)

- The Windows platforms end a line with the `\n\r` sequence
- The Macintosh platforms end a line with the `\r` sequence
- Files and directories have three levels of access: user, group, and other
- Typical file and directory permissions include read, write, and execute
- PHP provides the `chmod()` function for changing the permissions of a file within PHP

Summary (continued)

- The syntax for the `chmod()` function is `chmod($filename, $mode)`
- The `chmod()` function uses a four-digit octal value to assign permissions
- The `fileperms()`, which takes filename as the only parameter, returns a bitmap of the permissions associated with a file
- The `opendir()` function iterates through the entries in a directory

Summary (continued)

- A **handle** is a special type of variable that represents a resource, such as a file or directory
- To iterate through the entries in a directory, you open a handle to the directory with the `opendir()` function
- Use the `readdir()` function to return the file and directory names from the open directory
- Use the `closedir()` function to close a directory handle

Summary (continued)

- The `scandir()` function returns an indexed array of the files and directories (in ascending alphabetical order) in a specified directory
- The `mkdir()`, with a single name argument, creates a new directory
- The `is_readable()`, `is_writeable()`, and `is_executable()` functions check the the file or directory to determine if the PHP scripting engine has read, write, or execute permissions, respectively

Summary (continued)

- A **symbolic link**, which is identified with the `is_link()` is a reference to a file not on the system
- The `is_dir()` determines if a directory exists
- Directory information functions provide file access dates, file owner, and file type
- Uploading a file refers to transferring the file to a Web server

Summary (continued)

- Setting the `enctype` attribute of the opening from tag to multipart/form-data instructs the browser to post one section for regular form data and one section for file contents
- The `file` input type creates a browse button that allows the user to navigate to a file to upload
- To limit the size of the file upload, above the file input field, insert a hidden field with an attribute `MAX_FILE_SIZE` and a value in bytes

Summary (continued)

- An uploaded file's information (error code, temporary file name, filename, size, and type) is stored in the `$_FILES` array
- MIME (Multipurpose Internet Mail Extension) generally classifies the file upload as in "image.gif", "image.jpg", "text/plain," or "text/html"
- The `move_uploaded_file()` function moves the uploaded file to its permanent destination

Summary (continued)

- The `file_put_contents()` function writes or appends a text string to a file and returns the number of bytes written to the file
- The `FILE_APPEND` constant appends data to any existing contents in the specified filename instead of overwriting it
- The `file_get_contents()` and `readfile()` functions read the entire contents of a file into a string

Summary (continued)

- A **stream** is a channel that is used for accessing a resource to which you may read, and write.
- The **input stream** reads data from a resource, such as a file
- The **output stream** writes data to a resource, such as a file
- The `fopen()` opens a handle to a file stream using the syntax `$open_file = fopen("text file", "mode");`

Summary (continued)

- A **file pointer** is a variable that refers to the currently selected line or character in a file
- Mode arguments used with the `fopen()` function specifies if the file is opened for reading, writing, or executing, and the indicates the location of the file pointer
- The `fclose() function` with a syntax of `fclose($handle);` is used to close a file stream

Summary (continued)

- The `fwrite()` incrementally writes data to a text file
- To prevent multiple users from modifying a file simultaneously use the `flock()` function
- A number of PHP functions are available to iterate through a text file by line or character
- Use the `copy()` function to copy a file with PHP
- Use the `rename()` function to rename a file or directory with PHP

Summary (continued)

- The `unlink()` function is used to delete files and the `rmdir()` function is used to delete directories
- In lieu of a move function, the `rename()` function renames a file and specifies a new directory to store the renamed file