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
- Upload and download files
- Write data to files
- Read data from files
- Open and close a file stream
- Manage files and directories

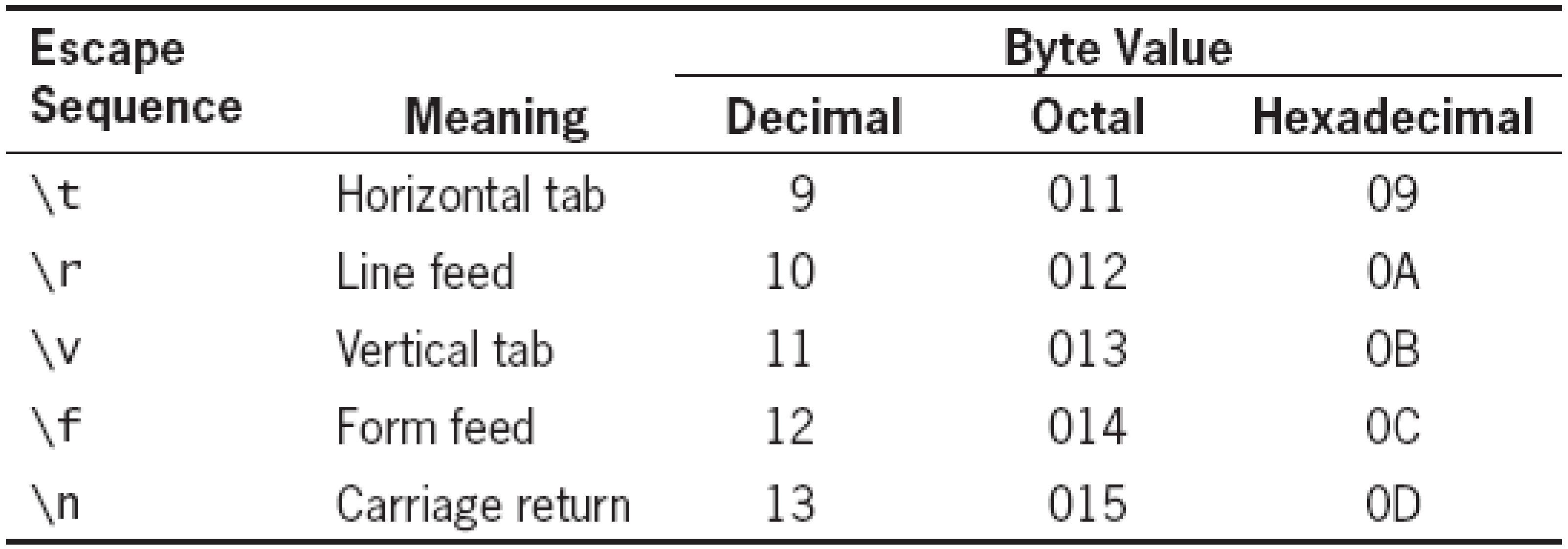
Understanding File Types and Permissions

- **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`, `\r` 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

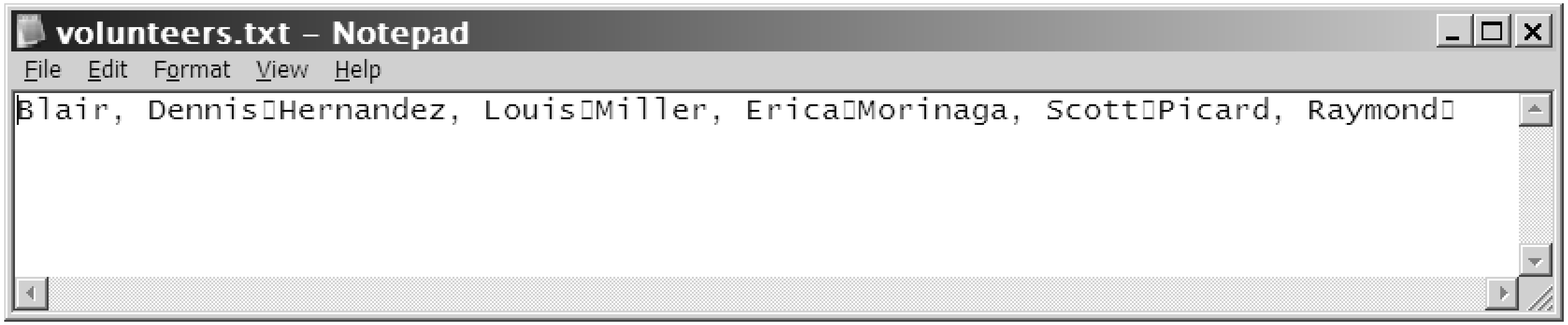
Horizontal tab, Line feed, Vertical tab, Form feed, Carriage Return, Control characters, Decimal, Octal, Hexadecimal,

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



```
Blair, Dennis□Hernandez, Louis□Miller, Erica□Morinaga, Scott□Picard, Raymond□
```

Figure 5-1 Volunteer registration form

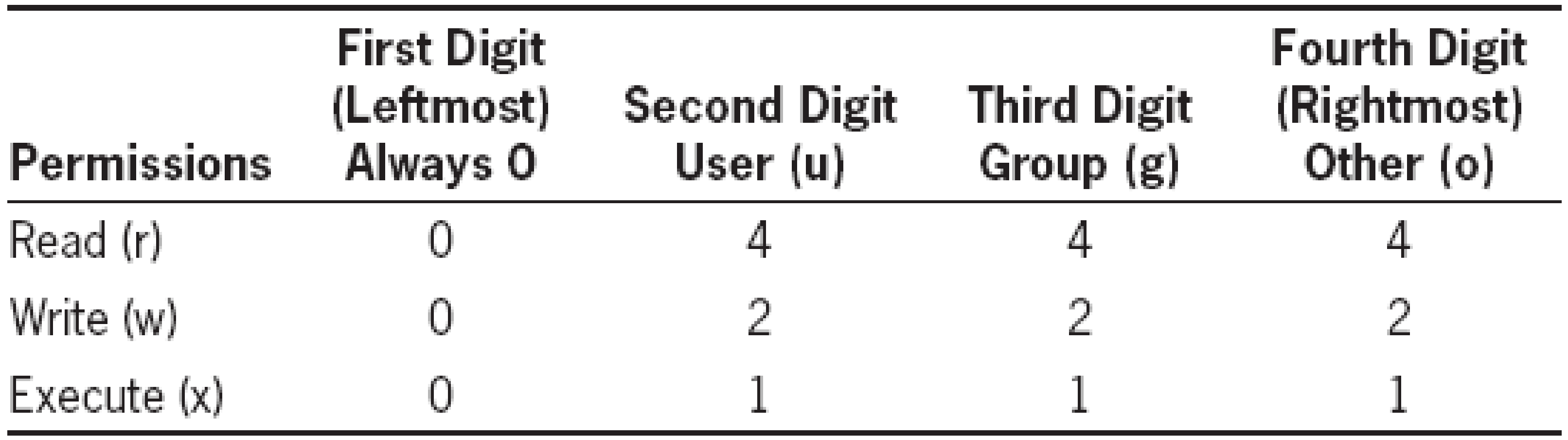
Working with File Permissions

- 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

Read (r), Write(w), Execute(x), User (u), Group (g), Other (o),

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

```
chmod(“example.exe”, 0644)
```

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 `decoct()` function converts a decimal value to an octal value

Short Quiz, p. 237-238

1. Explain the difference between a binary file and a text file.
   a. Binary file has no special meaning, just bits and bites, no operating system
2. What are the different end-of-line markers for Windows, Macintosh, and UNIX/Linux?
   a. Windows: \n, \r
   b. Mac: \r
   c. Linux: \n
3. What functions are used to change and retrieve the permissions of a file?
   a. chmod and file permissions
4. What are the 3 typical permissions for files and directories?
   a. Read, write, and execute
5. What are the 3 levels of access for files and directories?
   a. user, group, and other

Reading Directories

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

Function	Description
chdir(directory)	• Changes to 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

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

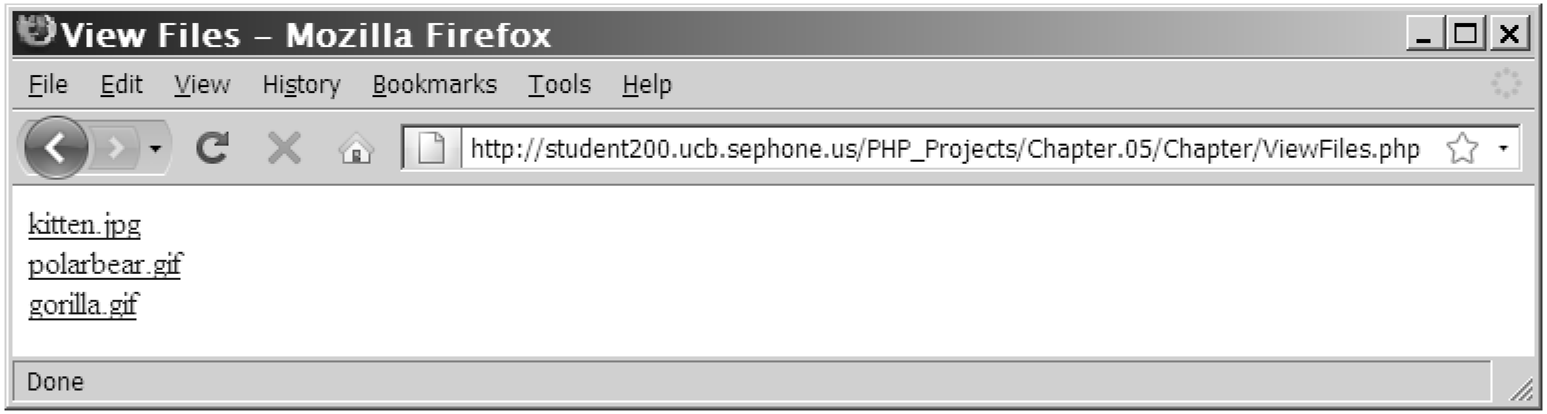
Reading Directories (continued)

- 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


View Files – Mozilla Firefox
File Edit View History Bookmarks Tools Help
Done

http://student200.ucb.sephone.us/PHP_Projects/Chapter.05/Chapter/ViewFiles.php

kitten.jpg
polarbear.gif
gorilla.gif

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

Reading Directories (continued)

- The PHP scripting engine returns the navigation shortcuts (“.” and “..”) when it reads a directory
- The `strcmp()` function can be used to exclude those entries

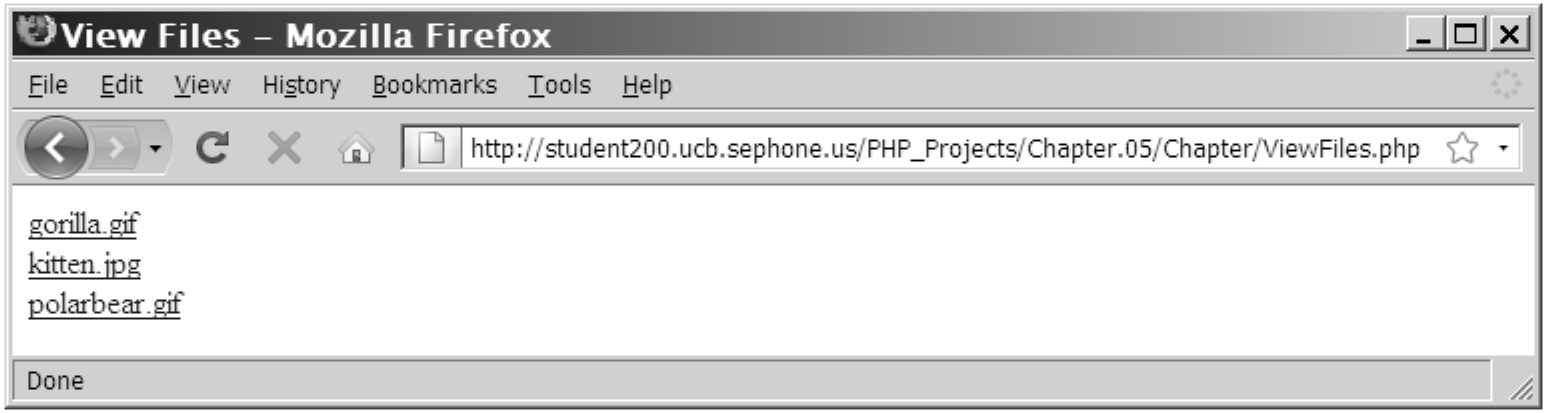
```
 ...
 while ($CurFile = readdir($DirOpen))
   if ((strcmp($CurFile, '.') != 0) &&
      (strcmp($CurFile, '..') != 0))
      echo "<a href=\"files/" . $CurFile . "\">" .
$CurFile . "</a><br />";
}
 ...
```

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)


View Files – Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://student200.ucb.sephone.us/PHP_Projects/Chapter.05/Chapter/ViewFiles.php
Done

```
gorilla.gif
kitten.jpg
polarbear.gif
```

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

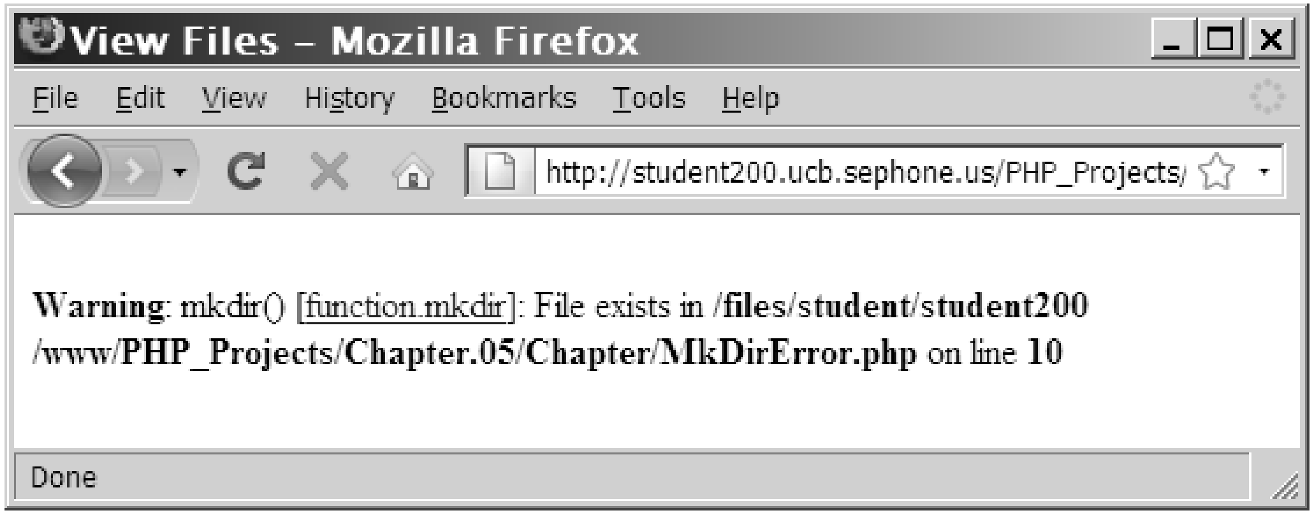
```
mkdir("volunteers");
```

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)


View Files – Mozilla Firefox
File
Edit
View
History
Bookmarks
Tools
Help
http://student200.ucb.sephone.us/PHP_Projects/
Done

```
Warning: mkdir() [function.mkdir]: File exists in /files/student/student200
/www/PHP_Projects/Chapter.05/Chapter/MkDirError.php on line 10
```

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

Obtaining File and Directory Information (continued)

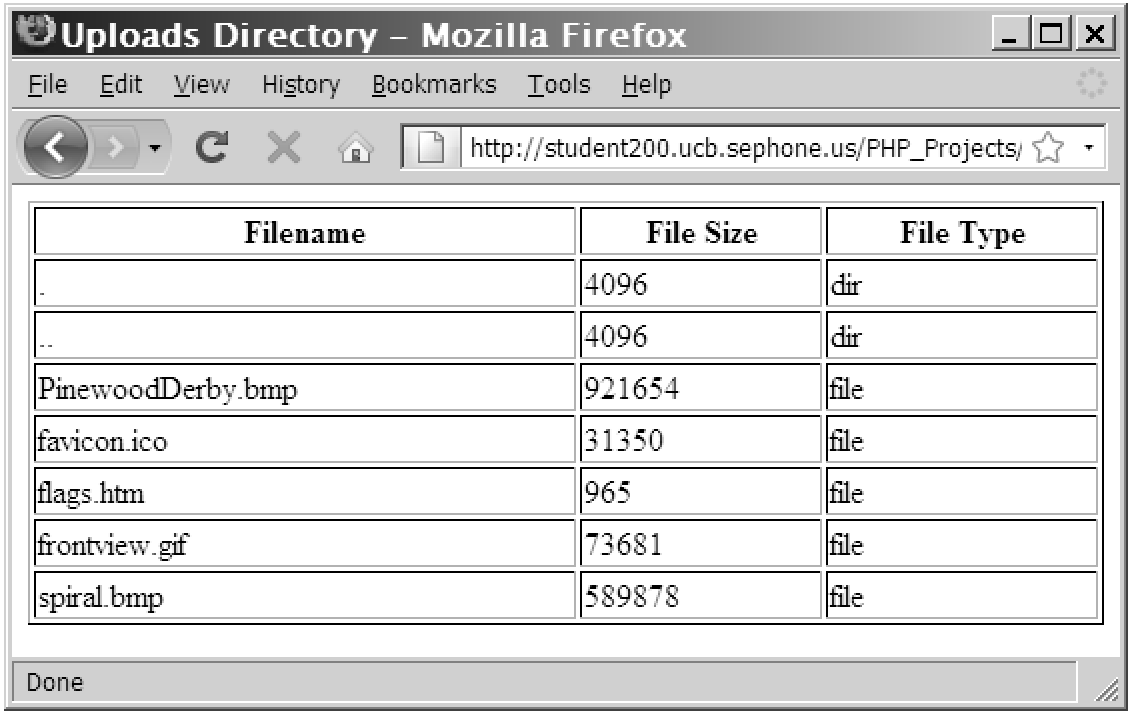
```
$Dir = "/var/html/uploads";
if (is_dir($Dir)) {  // check whether a specified filename is a
   directory before attempting to access it.
   echo "<table border='1' width='100%'>\n";
   echo "<tr><th>Filename</th><th>File Size</th>
      <th>File Type</th></tr>\n";
   $DirEntries = scandir($Dir);
   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)

- The following table returns additional information about files and directories:

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

Obtaining File and Directory Information (continued)


Uploads Directory – Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://student200.ucb.sephone.us/PHP_Projects/
Done

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 – code p. 244.

Short Quiz, p. 246-247

1. What functions are used to iterate through files and directories in a specific directory?
   a. readdir() and scandir()
2. What function returns an indexed array containing the names of files and directories in the specified directory?
   a. scandir()
3. What is one benefit of using the `scandir()` function versus the `readdir()` function?
   a. Dont have to manually open and close the directory
4. What function is used to create a directory?
   a. mkdir()
5. What functions are used to determine if a directory entry is a file or a directory?
   a. is_file and is_dir

Uploading and Downloading Files

- Web applications allow visitors to upload files to and from from their local computer (often referred to as the **client**)
- The files that are uploaded and downloaded may be simple text files or more complex file types, such as images, documents, or spreadsheets

Selecting the File

- Files are uploaded through an XHTML form using the "post" method
- An `enctype` attribute in the opening form tag must have a value of "multipart/form-data," which instructs the browser to post multiple sections – one for regular form data and one for the file contents

Selecting the File (continued)

- The `file` input field creates a Browse button for the user to navigate to the appropriate file to upload

```
<input type="file" name="picture_file" />
```

- The `MAX_FILE_SIZE` (uppercase) attribute of a hidden form field specifies the maximum number of bytes allowed in the uploaded file
  - The `MAX_FILE_SIZE` hidden field must appear before the file input field

Retrieving the File Information

- When the form is posted, information for the uploaded file is stored in the `$_FILES` autoglobal array
  `$_FILES['`*`filefield`*`']['`*`key`*`']`
- The `$_FILES[]` array contains five elements:
  - `$_FILES['picture_file']['`**`error`**`'] //   Contains the error code associated with  the file`
  - `$_FILES['picture_file']['`**`tmp_name`**`'] // Contains the temporary location of the file contents`

Retrieving the File Information (continued)

- `// Contains the name of the original file`
  `$_FILES['picture_file']['`**`name`**`']`

- `// Contains the size of the uploaded file in bytes`

  `$_FILES['picture_file']['`**`size`**`']`

- `// Contains the type of the file`

  `$_FILES['picture_file']['`**`type`**`']`

Storing the Uploaded File

- Uploaded files are either public or private depending on whether they should be immediately available or verified first
  - **Public** files are freely available to anyone visiting the Web site
  - **Private** files are only available to authorized visitors

Storing the Uploaded File (continued)

- The `move_uploaded_file()` function moves the uploaded file from its temporary location to a permanent destination with the following syntax:

  ```
  move_uploaded_file(string
  $filename, string $destination)
  ```

- *`$filename`* is the contents of `$_FILES['`*`filefield`*`']['tmp_name']` and *`$destination`* is the path and filename of the location where the file will be stored.

Storing the Uploaded File (continued)

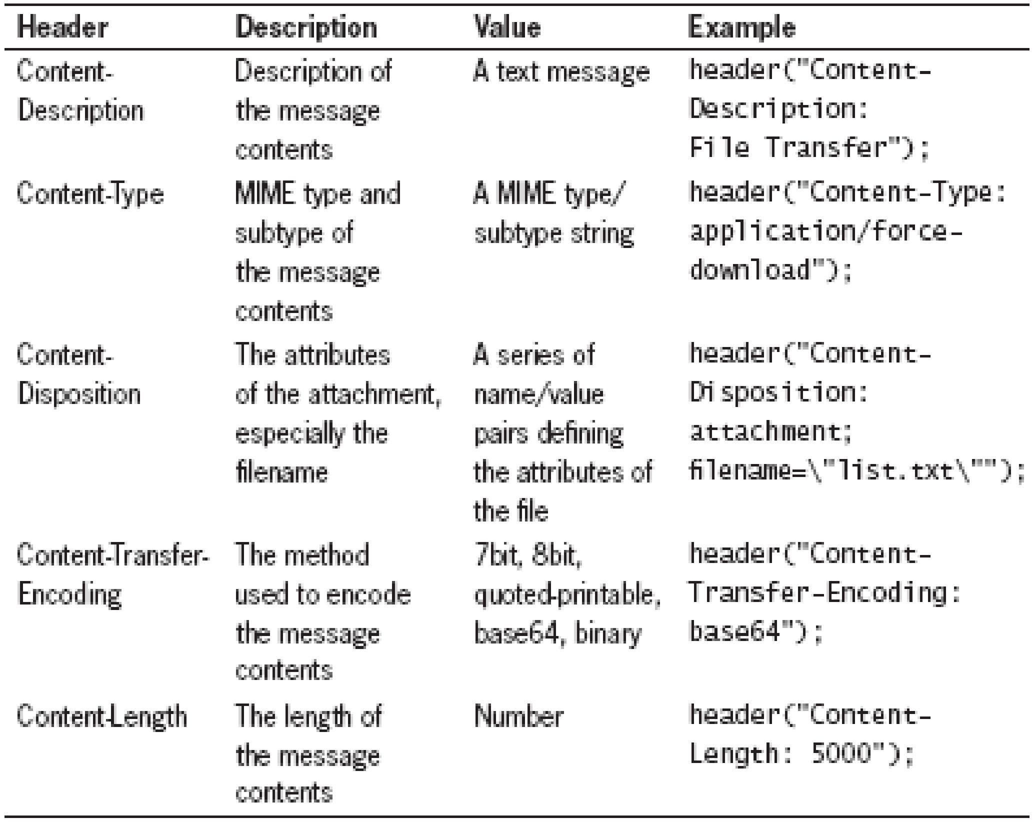
- The function returns TRUE if the move succeeds, and FALSE if the move fails

```
if
(move_uploaded_file($_FILES['picture_file']['tmp_name'],
"uploads/" . $_FILES['picture_file']['name']) === FALSE)
    echo "Could not move uploaded file to \"uploads/" .
htmlentities($_FILES['picture_file']['name']) . "\"<br
/>\n";

 else

echo "Successfully uploaded \"uploads/" .
htmlentities($_FILES['picture_file']['name']) . "\"<br
/>\n";
```

Downloading Files

- Files in the public XHTML directory structure can be downloaded with an XHTML hyperlink
- Files **outside the public XHTML directory** require a three-step process:
  - Tell the script which file to download
  - Provide the appropriate headers
  - Send the file – readfile() function
- The `header()` function is used to return header information to the Web browser

Downloading Files (continued)



Header	Description	Value	Example
Content-Description	Description of the message contents	A text message	`header("Content-Description: File Transfer");`
Content-Type	MIME type and subtype of the message contents	A MIME type/ subtype string	`header("Content-Type: application/force-download");`
Content-Disposition	The attributes of the attachment, especially the filename	A series of name/value pairs defining the attributes of the file	`header("Content-Disposition: attachment; filename=\"list.txt\"");`
Content-Transfer-Encoding	The method used to encode the message contents	7bit, 8bit, quoted-printable, base64, binary	`header("Content-Transfer-Encoding: base64");`
Content-Length	The length of the message contents	Number	`header("Content-Length: 5000");`

Table 5-7 Content headers for downloading a file

Content-Description, Description of, A text message Description, The message contents
Content-type, MIME type/subtype string
Content-Disposition, The attributes of the attachment, especially the file name, A series of name/value pairs defining the attributes of the file
Content-Transfer-Encoding, The method used to encode the message contents, 7bit, 8bit, quated-printable, base64, binary
Content-length, The length of the message contents, Number

Short Quiz, p. 256-257

1. What type of form input element is used to choose the file to upload?
   a. file
2. What hidden form input element restricts the size of the uploaded file?
   a. name="MAX_FILE_SIZE"
3. What is the name of the autoglobal array that contains the uploaded file information?
   a. files
4. What function is used to pass headers to the client Web browser?
   a. header()
5. What function is used to send the contents of a file to the client Web browser?
   a. readfile()

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)

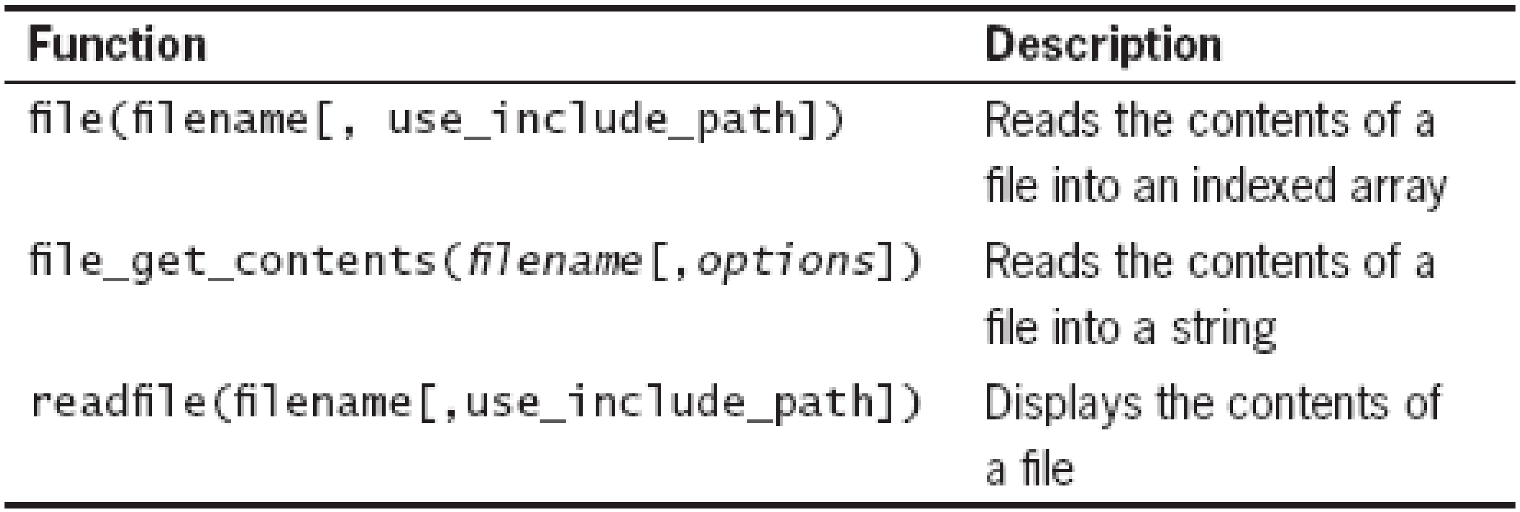
- You can use an absolute or relative path with the filename you pass to the function
- HOWEVER, even thought the function will create a filename that does not exist, it will not create directories that do not exist

```
$Dir = “comments”;
if (is_dir($Dir)) {
…
```

Writing an Entire File (continued)

- In addition to the filename and text string arguments, you can pass a third argument to the `file_put_contents()`function that contains either:
- 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

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

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)

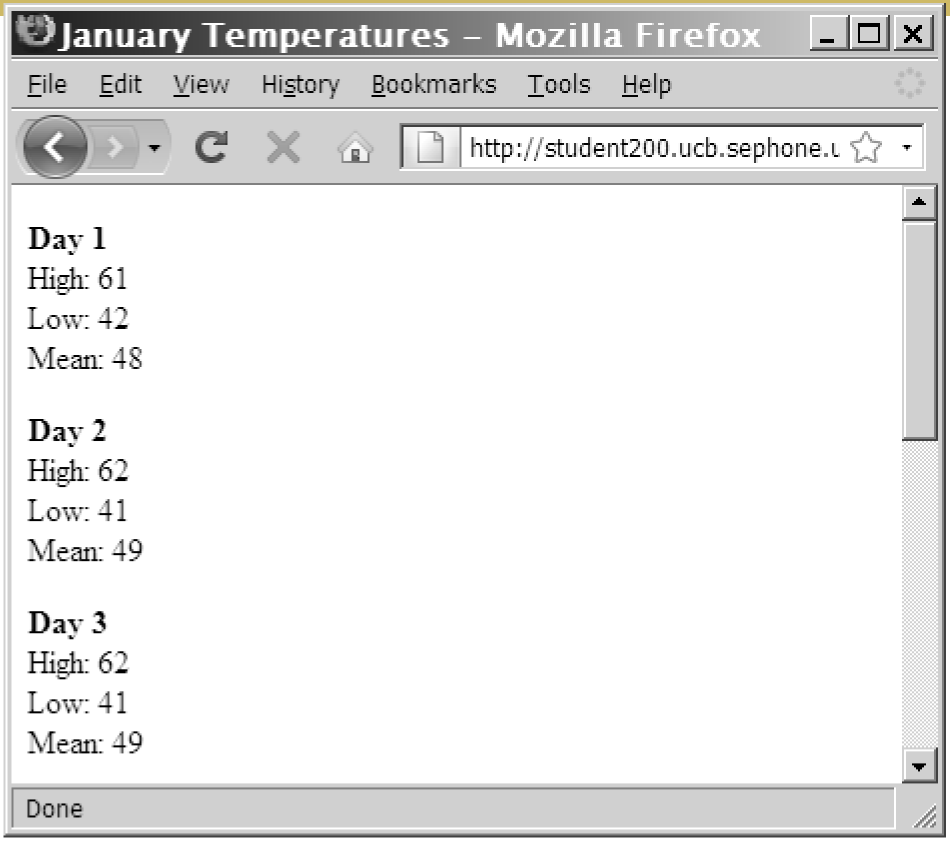
- The `file()` function reads the entire contents of a file into an indexed array
- Automatically recognizes whether the lines in a text file end in `\n`, `\r`, or `\r\n`

```
$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)


January Temperatures – Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://student200.ucb.sephone.u
Done

```
Day 1
High: 61
Low: 42
Mean: 48

Day 2
High: 62
Low: 41
Mean: 49

Day 3
High: 62
Low: 41
Mean: 49
```

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

Short Quiz, p. 267

1. Explain how to determine if the file_put_contents() function successfully wrote data to a file?
    a. It will return something greater than zero
2. Explain why one should use the is_dir() function before using the file_put_contents() function to write data.
    a. Just to make sure the directory is there
3. What function is useful for reading an entire file into a variable as a single block of text?
    a. file_get_contents()
4. What is the difference between the `file()` and `file_get_contents()` functions?
    a. file() returns an array
    b. file_get_contents() returns a string

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

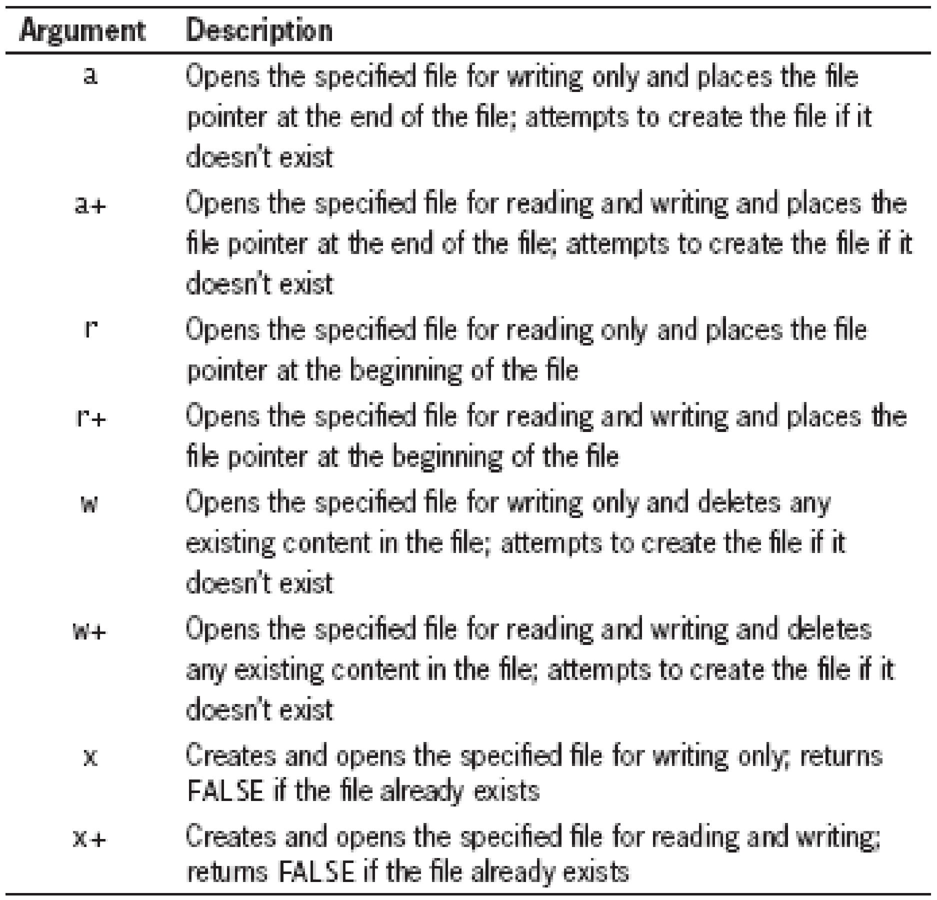
Opening a File Stream

- 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 – a way of keeping track of where you are in a file.

Opening a File Stream (continued)



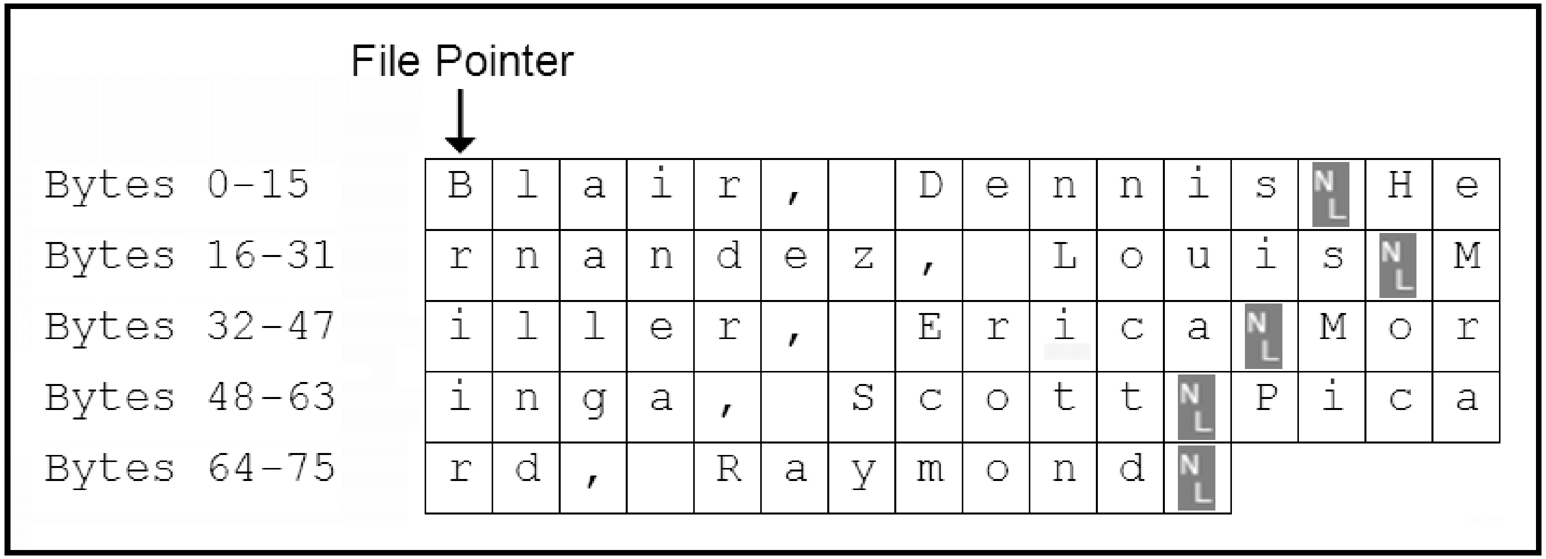
Argument	Description
a	Opens the specified file for writing only and places the file pointer at the end of the file; attempts to create the file if it doesn't exist
a+	Opens the specified file for reading and writing and places the file pointer at the end of the file; attempts to create the file if it 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

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 crate 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

Opening a File Stream (continued)

```
$VolunteersFile = fopen(“volunteers.txt", “r+");
```

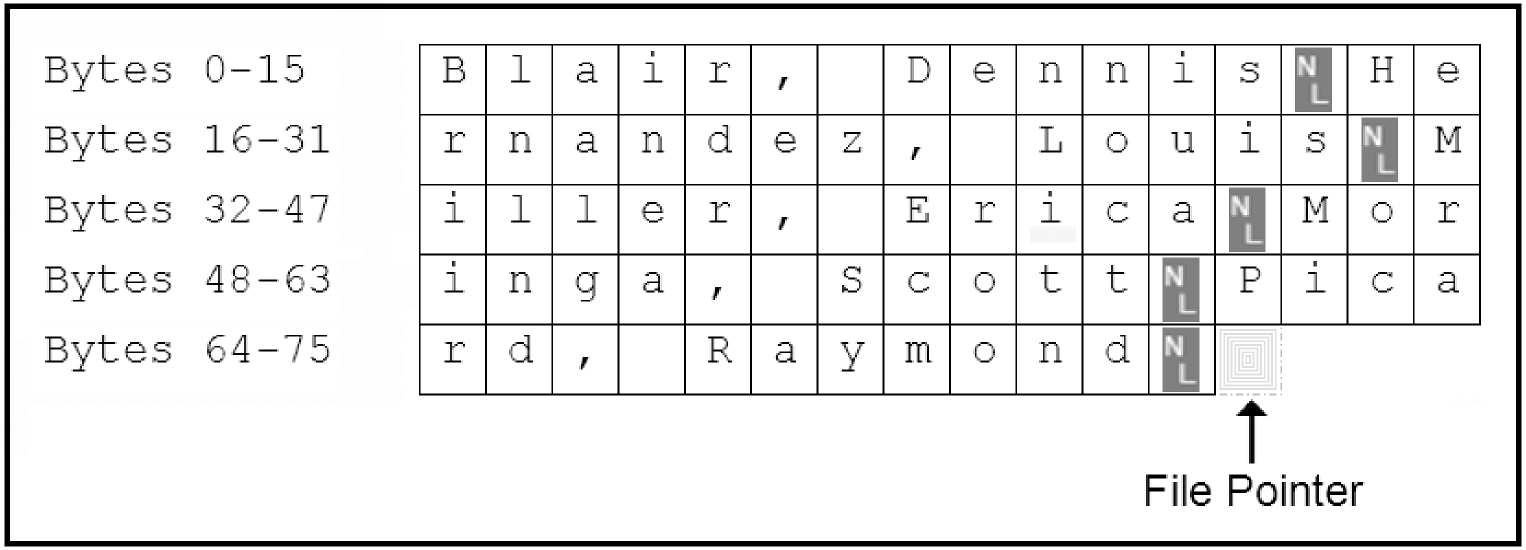


Bytes 0–15	B	l	a	i	r	,		D	e	n	n	i	s	NL	H	e
Bytes 16–31	r	n	a	n	d	e	z	,		L	o	u	i	s	NL	M
Bytes 32–47	i	l	l	e	r	,		E	r	i	c	a	NL	M	o	r
Bytes 48–63	i	n	g	a	,		S	c	o	t	t	NL	P	i	c	a
Bytes 64–75	r	d	,		R	a	y	m	o	n	d	NL				

Figure 5-15 Location of the file pointer when the `fopen()` function uses a *mode* argument of “`r+`”.

Opening a File Stream (continued)

```
$VolunteersFile = fopen(“volunteers.txt", “a+");
```



Bytes 0-15	B	l	a	i	r	,		D	e	n	n	i	s	NL	H	e
Bytes 16-31	r	n	a	n	d	e	z	,		L	o	u	i	s	NL	M
Bytes 32-47	i	l	l	e	r	,		E	r	i	c	a	NL	M	o	r
Bytes 48-63	i	n	g	a	,		S	c	o	t	t	NL	P	i	c	a
Bytes 64-75	r	d	,		R	a	y	m	o	n	d	NL				

Figure 5-16 Location of the file pointer when the `fopen()` **function uses a *mode* argument of “**`a+`**”.**

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

Writing Data Incrementally

- Use the `fwrite()` function to incrementally write data to a text file
- The syntax for the `fwrite()` function is:

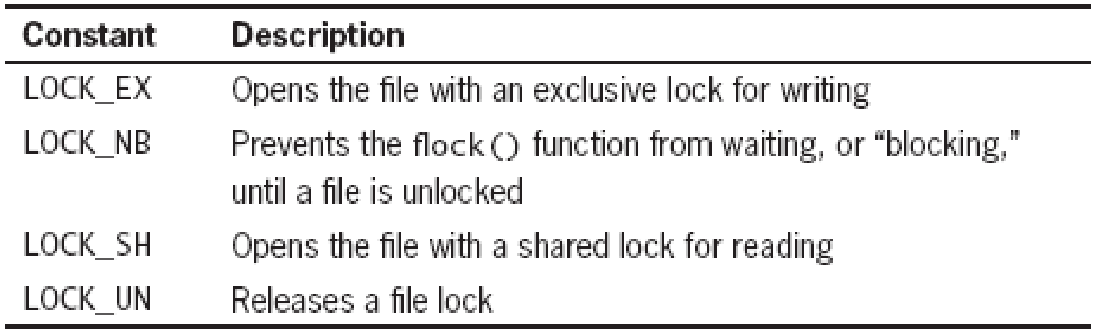
  `fwrite($`*`handle`*`, `*`data`*`[, `*`length`*`]);`

- The `fwrite()` function returns the number of bytes that were written to the file
- If no data was written to the file, the function returns a value of `0`
- `file_put_contents()` vs. `fwrite()`- p. 270

Locking Files

- To prevent multiple users from modifying a file simultaneously use the `flock()` function
- The syntax for the `flock()` function is:

```
flock($handle, operation)
```

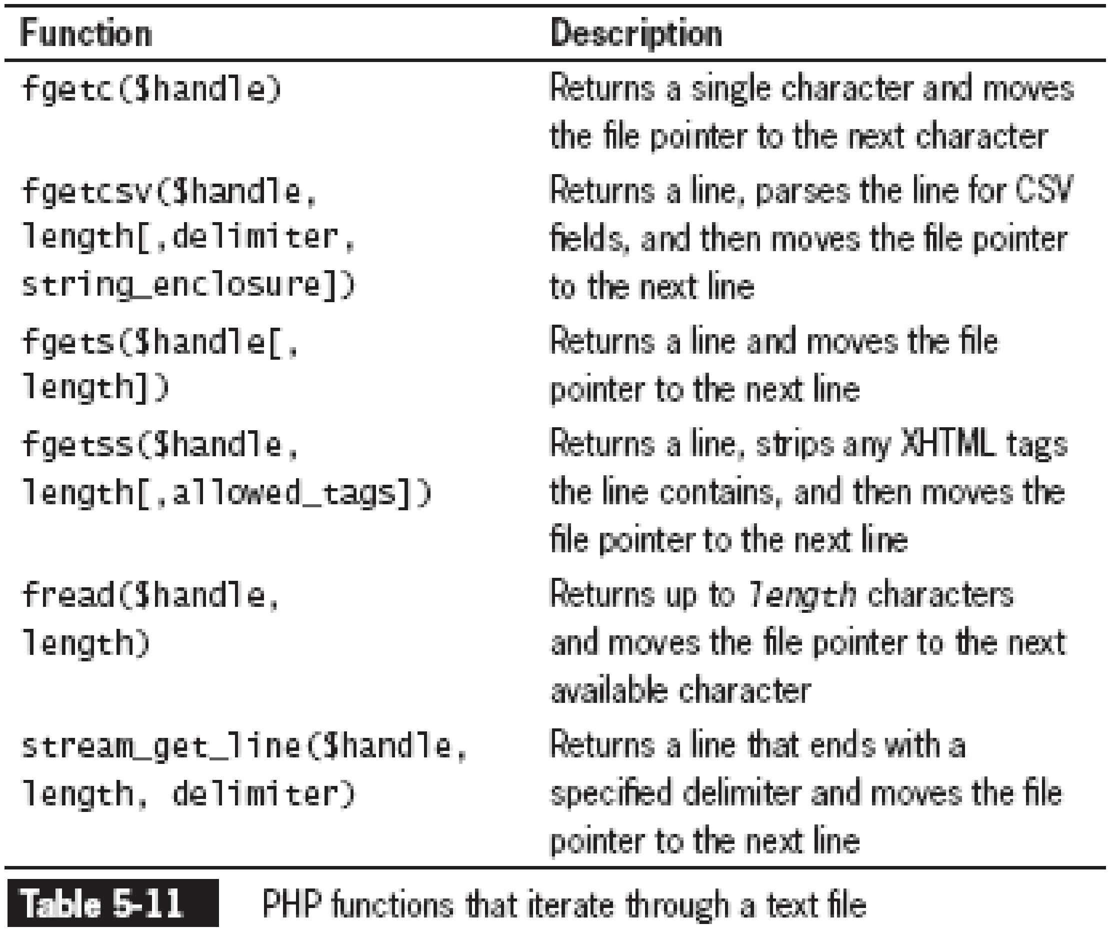


Constant	Description
LOCK_EX	Opens the file with an exclusive lock for writing
LOCK_NB	Prevents the flock() function from waiting, or "blocking," until a file is unlocked
LOCK_SH	Opens the file with a shared lock for reading
LOCK_UN	Releases a file lock

Table 5-10 Operational constants of the flock() function

LOCK_EX Opens the file with an exclusive lock for writing
LOCK_NB Prevents the flock() function from waiting, or "blocking," until a file is unlocked
LOCK_SH Opens the file with a shared lock for reading
LOCK_UN Releases a file lock

Reading Data Incrementally



	Function	Description
1	`fgetc($handle)`	Returns a single character and moves the file pointer to the next character
2	`fgetcsv($handle, length[,delimiter, string_enclosure])`	Returns a line, parses the line for CSV fields, and then moves the file pointer to the next line
3	`fgets($handle[, length])`	Returns a line and moves the file pointer to the next line
4	`fgetss($handle, length[,allowed_tags])`	Returns a line, strips any XHTML tags the line contains, and then moves the file pointer to the next line
5	`fread($handle, length)`	Returns up to *length* characters and moves the file pointer to the next available character
6	`stream_get_line($handle, length, delimiter)`	Returns a line that ends with a specified delimiter and moves the file pointer to the next line

Table 5-11 PHP functions that iterate through a text file

1. Returns a single character and moves the file pointer to the next character
2. Returns a line, parses the line for CSV fields, and then moves the file pointer to the next line
3. Returns a line and moves the file pointer to the next line
4. Returns a line, strips any XHTML tags the line contains, and then moves the file pointer to the next line
5. Returns up to length characters and moves the file pointer to the next available character
6. Returns a line that ends with a specified delimiter and moves the file pointer to the next line

Reading Data Incrementally (continued)

- You must use `fopen()` and `fclose()` with the functions listed in Table 5-11
- Each time you call any of the functions in Table 5-11, the file pointer automatically moves to the next ***line*** in the text file (except for `fgetc()` and `fread()`)
- Each time you call the `fgetc()` function, the file pointer moves to the next ***character*** in the file
- The `fread()` function advances the file pointer to the next available ***character*** in the file

Short Quiz, p. 277

1. What is a file stream?
    a. a channel that is used for accessing a research for which you may need to reed or write
2. Explain the function of the file pointer as it relates to writing data to files.
    a. refers to the currently selected line or character in the file
3. Explain the term “reading data incrementally.”
    a. use a file pointer to iterate through a text file instead of reading the entire file into PHP
4. What function is used to prevent multiple users from modifying a file simultaneously?
    a. flock()
5. What function must be called if the `fopen()` function successfully opened a file?
    a. fclose()

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

Short Quiz, p. 283

1. During the file copy process, what function is used to delete the original file?
   a.
2. Why is it important to use the `scandir()` function before using the `rmdir()` function?
   a.
3. What two entries will exist in a directory in most operating systems, even if the directory is empty?
   a.
4. How do you move a file in PHP?
   a.
5. What is the difference between the `unlink()` and `rmdir()` functions?
   a.

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