CHAPTER 6

Working with Files and Directories

Objectives

- Open and close files
- Write data to files
- Read data from files
- Manage files and directories

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 =  open("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

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 6-1 Mode arguments of the fopen() function

Opening a File Stream

```
$BowlersFile = fopen("bowlers.txt", "r+");
```

Figure 6-1 Location of the file pointer when the `fopen()` function uses a *`mode`* argument of "r+"

Opening a File Stream

```
$BowlersFile = fopen("bowlers.txt", "a+");
```

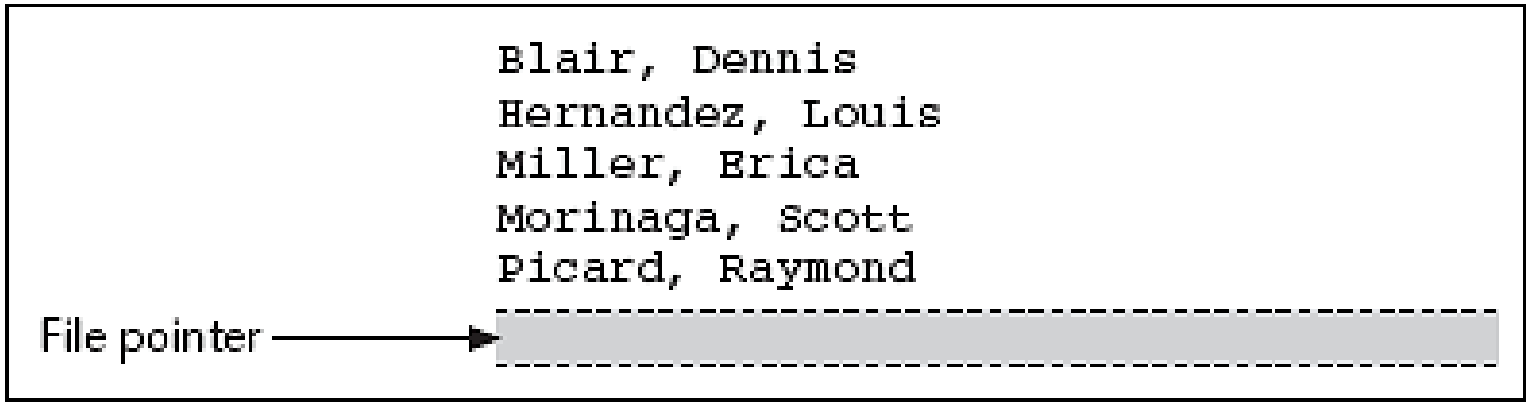


Figure 6-2 Location of the file pointer when the `fopen()` function uses a *`mode`* argument of "a+"

Closing a File Stream

- Use the `fclose` function when finished working with a file stream to save space in memory

```
$BowlersFile = fopen("bowlers.txt", "a");
$NewBowler = "Gosselin, Don\n";
fwrite($BowlersFile, $NewBowler);
fclose($BowlersFile);
```

Writing Data to Files

- PHP supports two basic functions for writing data to text files:
  - `file_put_contents()` function writes or appends a text string to a file
  - `fwrite()` function incrementally writes data to a text file

Writing Data to Files

- Escape sequences used to identify the end of a line:
  - UNIX/Linux platforms use the `\n` carriage return
  - Macintosh platforms use `\r` carriage return
  - Windows uses both the `\r` carriage return escape sequence and the `\n` newline escape sequence

Writing an Entire File

- 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])
```

file_put_contents() Function

```
$TournamentBowlers = "Blair, Dennis\n";
$TournamentBowlers .= "Hernandez, Louis\n";
$TournamentBowlers .= "Miller, Erica\n";
$TournamentBowlers .= "Morinaga, Scott\n";
$TournamentBowlers .= "Picard, Raymond\n";
$BowlersFile = "bowlers.txt";
file_put_contents($BowlersFile, $TournamentBowlers);
```

file_put_contents() Function

- 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($BowlersFile, $TournamentBowlers) > 0)
   echo "<p>Data was successfully written to the
            $BowlersFile file.</p>";
else
   echo "<p>No data was written to the $BowlersFile file.</p>";
```

Writing an Entire File

- 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

Writing an Entire File

```
<h1>Coast City Bowling Tournament</h1>
<?php
if (isset($_GET['first_name']) && isset($_GET['last_name'])) {
      $BowlerFirst = $_GET['first_name'];
      $BowlerLast = $_GET['last_name'];
      $NewBowler = $BowlerLast . ", " . "$BowlerFirst" . "\n";
      $BowlersFile = "bowlers.txt";
      if (file_put_contents($BowlersFile, $NewBowler, FILE_APPEND) > 0)
          echo "<p>{$_GET['first_name']} {$_GET['last_name']} has
              been registered for the bowling tournament!</p>";
      else
          echo "<p>Registration error!</p>";
}
else
      echo "<p>To sign up for the bowling tournament, enter your first
          and last name and click the Register button.</p>";
?>
<form action="BowlingTournament.php" method="get"
enctype="application/x-www-form-urlencoded">
<p>First Name: <input type="text" name="first_name" size="30" /></p>
<p>Last Name: <input type="text" name="last_name" size="30" /></p>
<p><input type="submit" value="Register" /></p>
</form>
```

Writing an Entire File

Figure 6-6 Bowling registration form

Handling Magic Quotes

- **Magic quotes** automatically adds a backslash (\) to any:
  - Single quote (')
  - Double quote (”)
  - NULL character contained in data that a user submits to a PHP script

```
My best friend's nickname is “Bubba”
My best friend\'s nickname is \”Bubba\”
```

Handling Magic Quotes

Table 6-2 Magic quote directives

Directive	Description
`magic_quotes_gpc`	Applies magic quotes to any user-submitted data
`magic_quotes_runtime`	Applies magic quotes to runtime-generated data, such as data received from a database
`magic_quotes_sybase`	Applies Sybase-style magic quotes, which escape special characters with a single quote (') instead of a backslash (\)

- Disable magic quotes in your php.ini configuration file and instead manually escape the strings with the `addslashes()` function

addslashes() Function

- Accepts a single argument representing the text string you want to escape and returns a string containing the escaped string

```
$Nickname = addslashes($_GET['nickname']);
echo $Nickname; // My best friend\'s nickname is \”Bubba\”.
```

- With magic quotes enabled:

```
My best friend\\\'s nickname is \\\”Bubba\\\”
```

stripslashes() Function

- Removes slashes that were added with the `addslashes()` function
- To prevent the display of escaped characters, use the `stripslashes()` function with the text you want to print

```
if (file_put_contents($BowlersFile, $NewBowler, FILE_APPEND) > 0)
      echo "<p>" . stripslashes($_GET['first_name']) . " "
             . stripslashes($_GET['last_name'])
             . " has been registered for the bowling tournament!</p>";
else
      echo "<p>Registration error!</p>";
```

stripslashes() Function

```
if (isset($_GET['first_name']) && isset($_GET['last_name'])) {
      $BowlerFirst = addslashes($_GET['first_name']);
      $BowlerLast = addslashes($_GET['last_name']);
      $NewBowler = $BowlerLast . ", " . "$BowlerFirst" . "\n";
      $BowlersFile = "bowlers.txt";
      if (file_put_contents($BowlersFile, $NewBowler, FILE_APPEND) >
   0)
            echo "<p>{$_GET['first_name']}{$_GET['last_name']}
                  has been registered for the bowling tournament!</p>";
      else
            echo "<p>Registration error!</p>";
}
else
      echo "<p>To sign up for the bowling tournament, enter your first
            and last name and click the Register button.</p>";
```

stripslashes() Function

Figure 6-7 Output of text with escaped characters

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

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)
```

Table 6-3 Operational constants of the `flock()` function

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

Reading an Entire File

Table 6-4 PHP functions that read the entire contents of a text file

Function	Description
`file(filename[, use_include_path])`	Reads the contents of a file into an indexed array
`file_get_contents(filename[, use_include_path])`	Reads the contents of a file into a string
`fread($handle, length)`	Reads the contents of a file into a string up to a maximum number of bytes
`readfile(filename[, use_include_path])`	Prints the contents of a file

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;
```

`readfile()` Function

- Prints the contents of a text file along with the file size to a Web browser

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

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 = "48, 42, 68\n";
$January .= "48, 42, 69\n";
$January .= "49, 42, 69\n";
$January .= "49, 42, 61\n";
$January .= "49, 42, 65\n";
$January .= "49, 42, 62\n";
$January .= "49, 42, 62\n";
file_put_contents("sfjanaverages.txt", $January);
```

file() Function

```
$JanuaryTemps = file("sfjanaverages.txt");
for ($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>";
}
```

file() Function

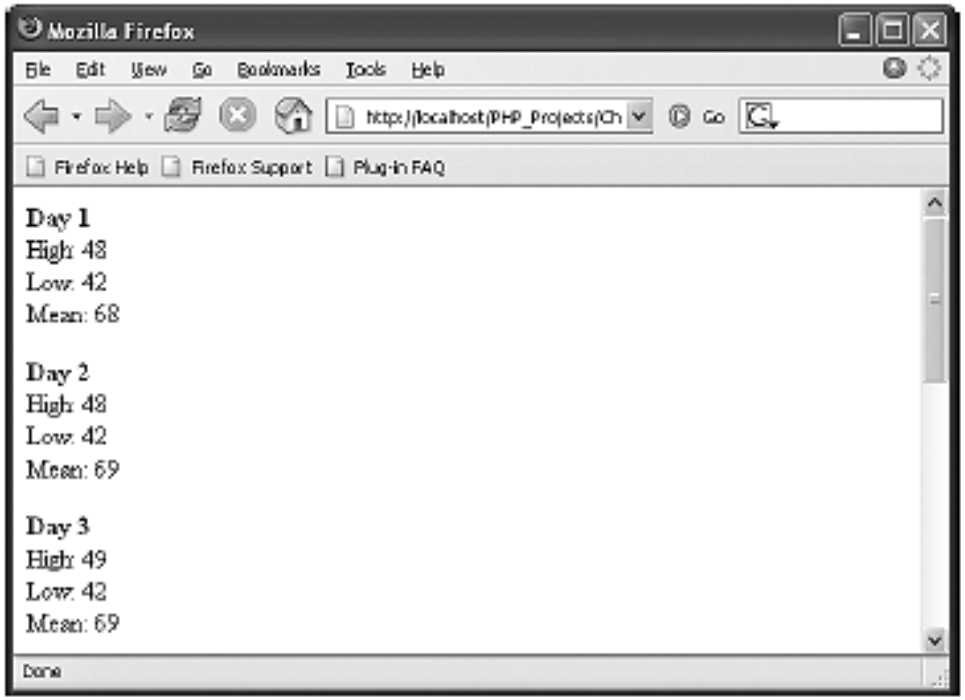


Figure 6-8 Output of individual lines in a text file

Reading Data Incrementally

Table 6-5 PHP functions that iterate through a text file

Function	Description
`fgetc($handle)`	Returns a single character and moves the file pointer to the next character
`fgetcsv($handle, length[, delimiter, string_enclosure])`	Returns a line, parses the line for CSV fields, and then moves the file pointer to the next line
`fgets($handle[, length])`	Returns a line and moves the file pointer to the next line
`fgetss($handle, length[, allowed_tags])`	Returns a line, strips any HTML tags the line contains, and then moves the file pointer to the next line
`stream_get_line($handle, length, delimiter)`	Returns a line that ends with a specified delimiter and moves the file pointer to the next line

- The `fgets()` function uses the file pointer to iterate through a text file

Reading Data Incrementally

- You must use `fopen()` and `fclose()` with the functions listed in Table 6-5
- Each time you call any of the functions in Table 6-5, the file pointer automatically moves to the next ***line*** in the text file (except for `fgetc()`)
- Each time you call the `fgetc()` function, the file pointer moves to the next ***character*** in the file

Reading Directories

Table 6-6 PHP directory functions

Function	Description
`chdir(directory)`	Changes to the specified directory
`chroot(directory)`	Changes to the root 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

- To iterate through the entries in a directory, 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

Reading Directories

```
$Dir = "C:\\PHP";
$DirOpen = opendir($Dir);
while ($CurFile = readdir($DirOpen)) {
        echo $CurFile . "<br />";
}
closedir($DirOpen);
```

scandir() Function

- Returns an indexed array containing the names of files and directories in the specified directory

```
$Dir = "C:\\PHP";
$DirEntries = scandir($Dir);
foreach ($DirEntries as $Entry)
{
        echo $Entry . "<br />";
}
```

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("bowlers");
```

Creating Directories

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

```
mkdir("..\\tournament");
mkdir("C:\\PHP\\utilities");
```

Creating Directories

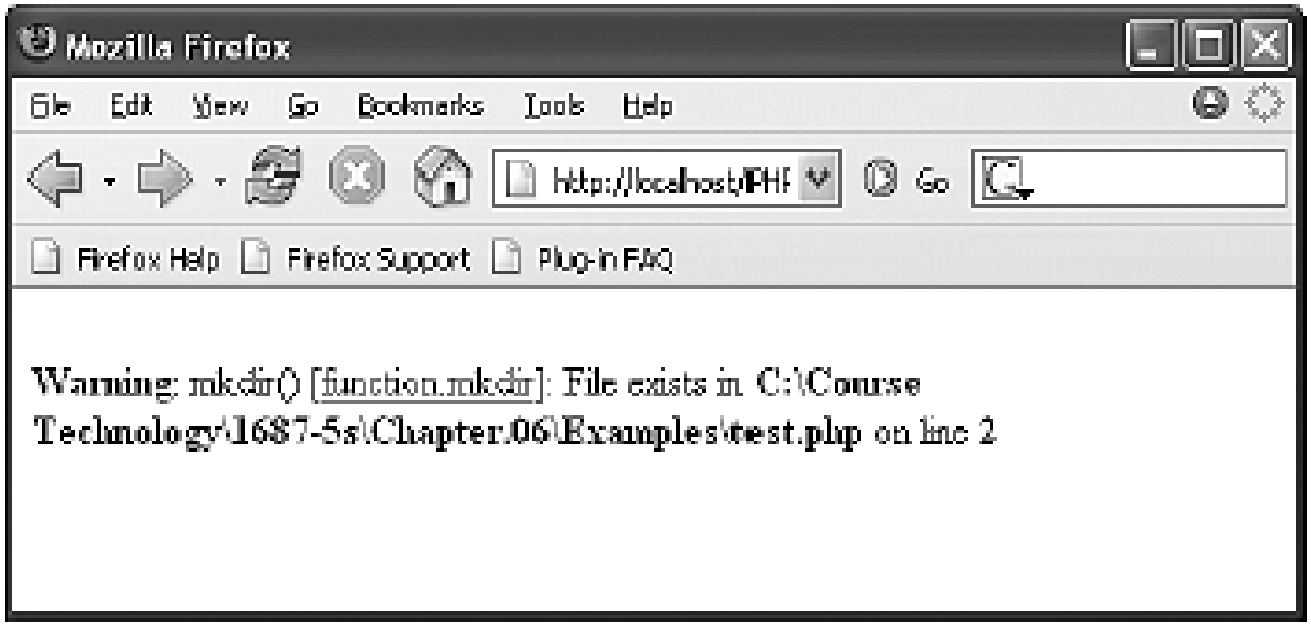


Figure 6-9 Warning that appears if a directory already exists

Obtaining File and Directory Information

Table 6-7 PHP file and directory status functions

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

Obtaining File and Directory Information

```
$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>";
$WeatherFile = "sfweather.txt";
if (is_writable($WeatherFile)) {
     file_put_contents($WeatherFile, $DailyForecast);
     echo "<p>The forecast information has been saved to
           the $WeatherFile file.</p>";
}
else
     echo "<p>The forecast information cannot be saved to
           the $WeatherFile file.</p>";
```

Obtaining File and Directory Information

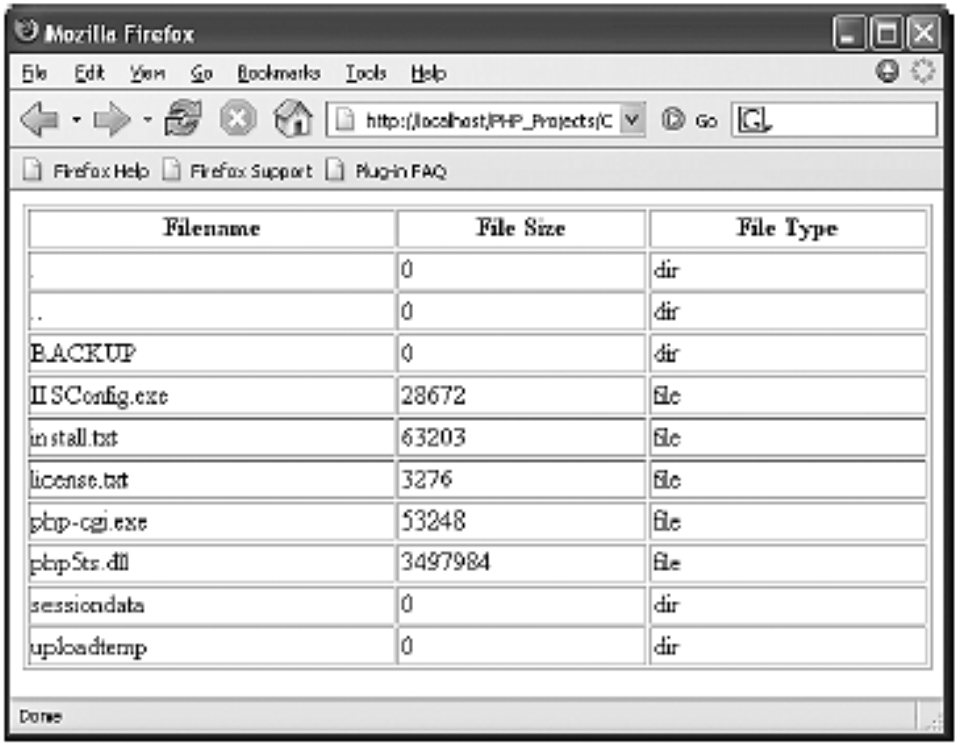
Table 6-8 Common file and directory information functions

Function	Description
`fileatime(filename)`	Returns the last time the file was accessed
`filectime(filename)`	Returns the last time the file was modified
`fileowner(filename)`	Returns the name of the file's owner
`filetype(filename)`	Returns the file type

Obtaining File and Directory Information

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

Obtaining File and Directory Information



Filename	File Size	File Type
.	0	dir
..	0	dir
BACKUP	0	dir
IISConfig.exe	28672	file
install.txt	63203	file
license.txt	3276	file
php-cgi.exe	53248	file
phpSts.dll	3497984	file
sessiondata	0	dir
uploadtemp	0	dir

Figure 6-10 Output of script with file and directory information functions

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

```
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

- The stream is used for accessing a resource, such as a file, that you can read from and write to
- A handle is a special type of variable that PHP uses to represent a resource such as a file
- The `fopen()` function opens a stream to a text file
- A file pointer is a special type of variable that refers to the currently selected line or character in a file

Summary

- Use the `fclose()` function to ensure that the file doesn't keep taking up space in your computer's memory
- PHP supports two basic methods for writing data to text files: `file_put_contents()` and the `fwrite()` function
- Magic quotes automatically add backslashes to any single quote, double quote, or NULL character contained in data that a user submits to a PHP script

Summary

- PHP includes various functions, such as the `fgets()` function, that allow you to use the file pointer to iterate through a text file
- To iterate through the entries in a directory, you open a handle to the directory with the `opendir()` function
- PHP includes various file and directory status functions, such as the `file_exists()` function, which determines whether a file or directory exists