

Files and Directories (part two)

**MIS 4530
Dr. Garrett**

Reading an Entire File

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

Function	Description
<code>file(filename[, use_include_path])</code>	Reads the contents of a file into an indexed array
<code>file_get_contents(filename[, use_include_path])</code>	Reads the contents of a file into a string
<code>fread(\$handle, length)</code>	Reads the contents of a file into a string up to a maximum number of bytes
<code>readfile(filename[, use_include_path])</code>	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 (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>";  
}
```

file () Function (continued)

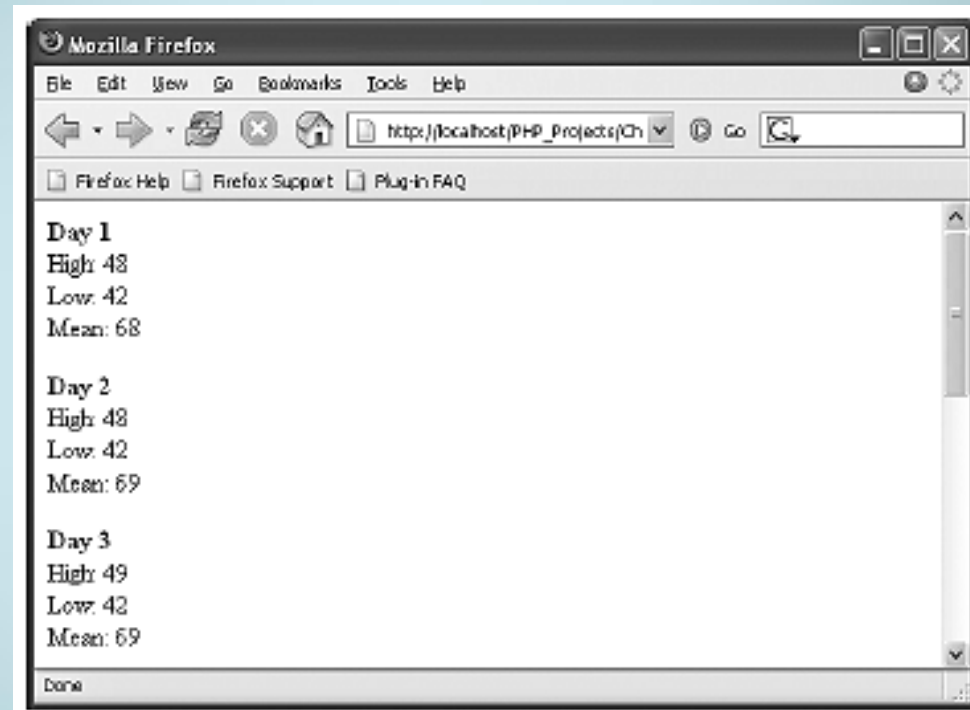


Figure 6-8 Output of individual lines in a text file

Reading Data Incrementally

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

Function	Description
<code>fgetc(\$handle)</code>	Returns a single character and moves the file pointer to the next character
<code>fgetcsv(\$handle, length[, delimiter, string_enclosure])</code>	Returns a line, parses the line for CSV fields, and then moves the file pointer to the next line
<code>fgets(\$handle[, length])</code>	Returns a line and moves the file pointer to the next line
<code>fgetss(\$handle, length[, allowed_tags])</code>	Returns a line, strips any HTML tags the line contains, and then moves the file pointer to the next line
<code>stream_get_line(\$handle, length, delimiter)</code>	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 (continued)

- 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
<code>chdir(<i>directory</i>)</code>	Changes to the specified directory
<code>chroot(<i>directory</i>)</code>	Changes to the root directory
<code>closedir(<i>\$handle</i>)</code>	Closes a directory handle
<code>getcwd()</code>	Gets the current working directory
<code>opendir(<i>directory</i>)</code>	Opens a handle to the specified directory
<code>readdir(<i>\$handle</i>)</code>	Reads a file or directory name from the specified directory handle
<code>rewinddir(<i>\$handle</i>)</code>	Resets the directory pointer to the beginning of the directory
<code>scandir(<i>directory</i>[, <i>sort</i>])</code>	Returns an indexed array containing the names of files and directories in the specified directory

Reading Directories (continued)

- 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 (continued)

```
$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 (continued)

- 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 (continued)

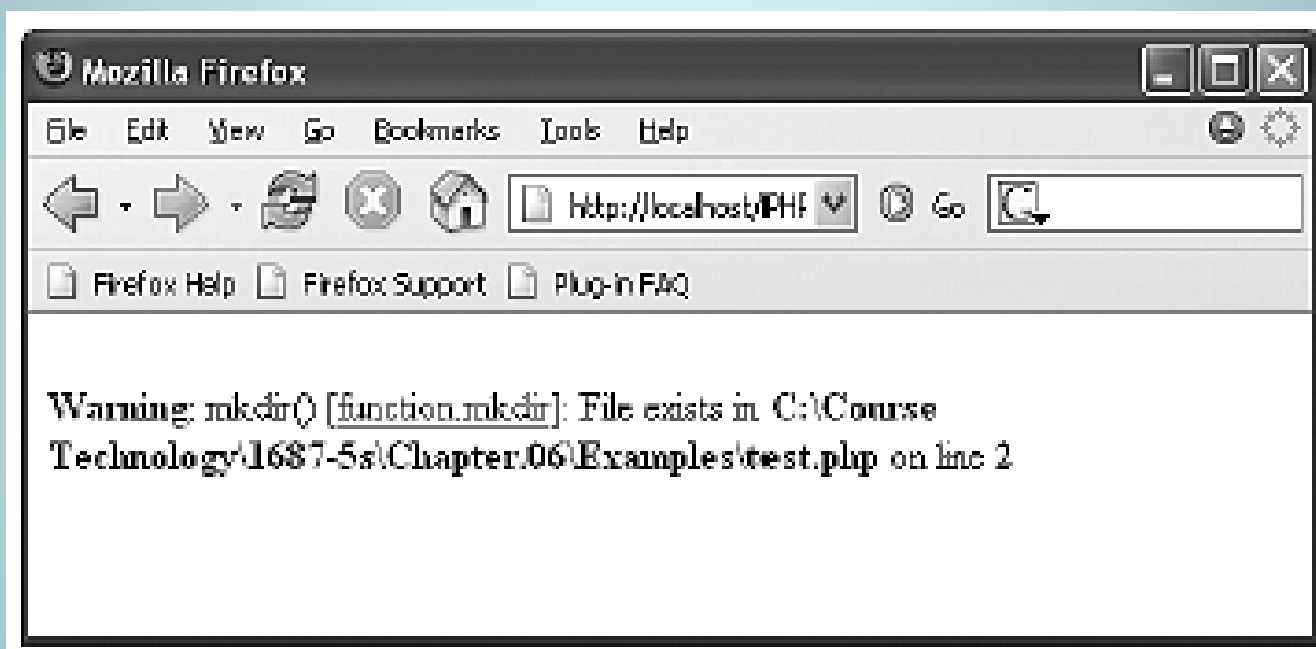


Figure 6-9 Warning that appears if a directory already exists

# Obtaining File and Directory Information

Table 6-7 PHP file and directory status functions

| Function                             | Description                                   |
|--------------------------------------|-----------------------------------------------|
| <code>file_exists(filename)</code>   | Determines whether a file or directory exists |
| <code>is_dir(filename)</code>        | Determines whether a filename is a directory  |
| <code>is_executable(filename)</code> | Determines whether a file is executable       |
| <code>is_file(filename)</code>       | Determines whether a file is a regular file   |
| <code>is_readable(filename)</code>   | Determines whether a file is readable         |
| <code>is_writable(filename)</code>   | Determines whether a file is writable         |

# Obtaining File and Directory Information (continued)

```
$DailyForecast = "<p>San Francisco daily weather
forecast: 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 (continued)

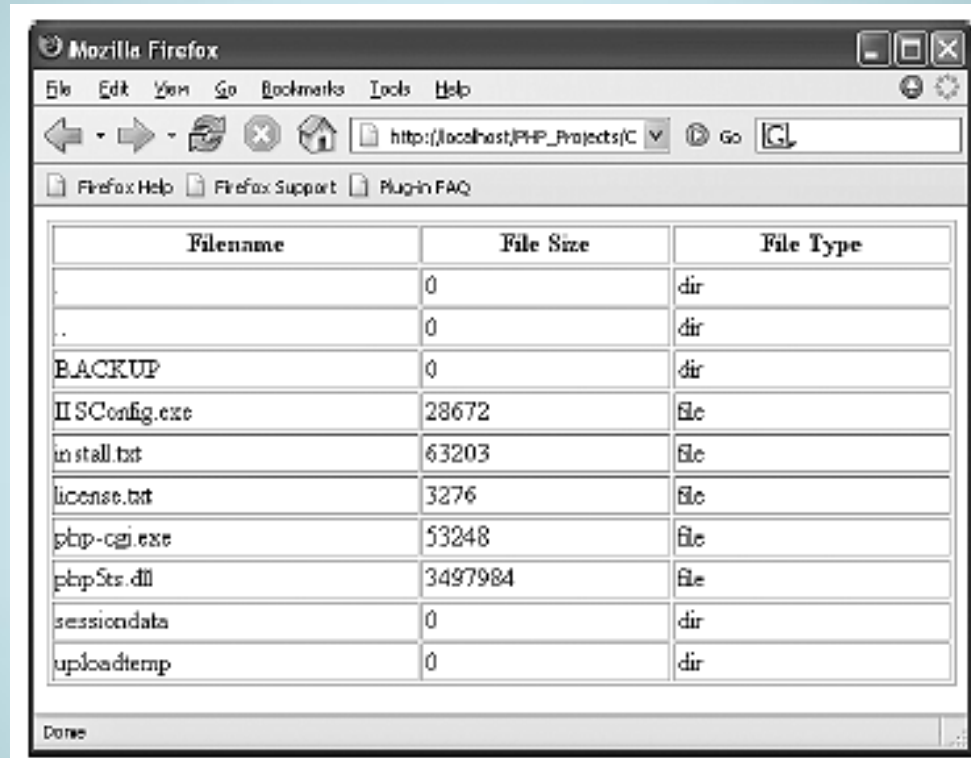
Table 6-8 Common file and directory information functions

| Function                         | Description                                 |
|----------------------------------|---------------------------------------------|
| <code>fileatime(filename)</code> | Returns the last time the file was accessed |
| <code>filectime(filename)</code> | Returns the last time the file was modified |
| <code>fileowner(filename)</code> | Returns the name of the file's owner        |
| <code>filetype(filename)</code>  | Returns the file type                       |

# Obtaining File and Directory Information (continued)

```
$Dir = "C:\\PHP";
if(is_dir($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 (continued)



The screenshot shows a Mozilla Firefox browser window with the address bar set to `http://localhost/PHP_Projects/C`. The browser displays a table with three columns: **Filename**, **File Size**, and **File Type**. The table lists various files and directories, including `BACKUP`, `II SConfig.exe`, `install.txt`, `license.txt`, `php-cgi.exe`, `php5ts.dll`, `sessiondata`, and `uploadtemp`.

| Filename       | File Size | File Type |
|----------------|-----------|-----------|
| .              | 0         | dir       |
| ..             | 0         | dir       |
| BACKUP         | 0         | dir       |
| II SConfig.exe | 28672     | file      |
| install.txt    | 63203     | file      |
| license.txt    | 3276      | file      |
| php-cgi.exe    | 53248     | file      |
| php5ts.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 (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)
```