CHAPTER 5

WORKING WITH FILES AND DIRECTORIES

PHP PROGRAMMING WITH MYSQL 2ND EDITION

MODIFIED BY ANITA PHILIPP -SPRING 2012

2

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

3

FILE TYPES

- **Binary**
  - Series of characters or bytes
  - No special meaning
  - Application determines structure
- **Text**
  - Only printable characters
  - Small set of control or formatting characters
  - End-of-line character sequences: `\n` or `\r\n`

4

File Types

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

5

File Types

Escape Sequence Markers

- UNIX/Linux: \n
- Windows: \n\r
- Macintosh: \r
  - OS X uses Linux core so most command line and OS programs use UNIX/Linux \n carriage return

\n: carriage return \r: line feed

6

File Types

- Written in UNIX/Linux
- Displayed in Windows

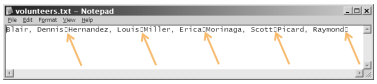


Figure 5-1 Volunteer registration form

7

File Permissions

- **Levels of Access**
  - User
  - Group
  - Other
- **Permissions**
  - Read (r)
  - Write (w)
  - Execute (x)

8

File Permissions

- **Four-digit Octal (base 8) Value**
  - The first digit is always 0
  - Sum of 3 bits per digit
  - Matches 3 permission bits per level of access

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

0640 ***User:*** Read/Write ***Group:*** Read ***Other:*** No Permissions

9

File Permissions

- Change the permissions: Files or Directories

```
chmod($filename, $mode)
chmod("index.html", 0640);
```

- Directories
  - **Read:** List files
    - Read a file but not directory
    - Open only if filename is known
    - Cannot see file in directory listing
  - **Write:** Add new files
  - **Execute:** Access files in directory (assuming file permissions allow)

10

Checking Permissions

- `fileperms()` returns an integer bitmap of permissions
- Permissions extracted using modulus operator with 01000_8
- `decoct()` converts decimal to octal

```
<?php
  $permissions = fileperms("proverbs.txt");
  $permissions = decoct($permissions % 01000);
  echo '<p>Permissions for "proverbs.txt": 0'. $permissions . '</p>';
?>
```

Output:

Permissions for "proverbs.txt": 0646

11

Reading 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

12

Reading Directories (continued)

- **Special Variables**
  - **Handle**
    - represents a resource such as a file or a directory
  - **Directory Pointer** special type of variable
    - refers to the currently selected record in a directory listing
- **Functions**
  - `opendir(`*directory*`)`
    - Opens specified directory
  - `readdir(`*handle*`)`
    - returns the file and directory names
  - `closedir(`*handle*`)`

13

Reading Directories

opendir(), readdir(), closedir()

```
$Dir = "/var/html/uploads";
$DirOpen = opendir($Dir);
while ($CurFile=readdir($DirOpen)) {

   echo $CurFile . "<br />\n";
   }
closedir($DirOpen);
```

scandir()

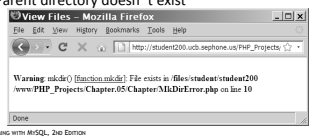
```
$Dir = "/var/html/uploads";
$DirEntries = scandir($Dir);
 foreach ($DirEntries as $Entry) {
   echo $Entry . "<br />\n";
   }
```

Note: *Code must be added to accommodate . and .. In the file structure (pp. 239, 241)*

14

Creating Directories

- Create within the current directory
  - Pass just the new directory name
  - `mkdir("volunteers");`
- Create in another location
  - Use a relative or an absolute path
  - `mkdir("../event");` *// Same level as current*
  - `mkdir("/bin/PHP/utilities");` *//absolute*
- Errors
  - Directory exists
  - Parent directory doesn't exist



15

File & 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

Checks Permissions | Types of Entries

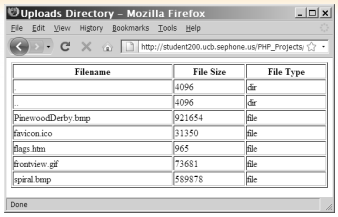
16

File & Directory Information

```
$Dir = "/var/html/uploads";
if (is_dir($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>";
    }
```


17

File &Directory Information



18

File & Directory Information

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

19

Managing Files and Directories

- File and Directory Management
  - Copying
  - Moving
  - Renaming
  - Deleting

20

Copying and Moving Files

- Returns TRUE-success, FALSE-fail
- Syntax: `copy(source, destination)`
- Same Directory: FileName
- Different Directory: Path and FileName

```
if (file_exists("sfweather.txt")) {
        if(is_dir("history")) {
           if (copy("sfweather.txt","history\
    \sfweather_rev.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>";
}
```

Note: Copies but file exists in original location. Delete source file if needed.

21

Renaming Files and Directories

- Returns TRUE-success, FALSE-fail
- Syntax: `rename(OldName, NewName)`

```
if (file_exists($OldName)) \\File exists {
        if(is_dir($NewDirectory")) \\ Directory exists {
           if (!file_exists($NewDirectory . "\\" . $newName))
              // File by same name does not exist{
           if (rename($oldName, $NewDirectory . "\\". $newName))
             // Successfully renamed message
           else
             // rename file error message
        }else{
              // file exists in new location error message
        }
        }else {
        // directory doesn't exists error message
        }
}else{
        // file does not exist error message
}
```

22

Removing Files and Directories

- Pass name of file or directory
- Returns TRUE-success, FALSE-failed
- Test for existence with **`file_exists()`**
- **`unlink()`**
  - Delete files
- **`rmdir()`**
  - Delete directories

```
$FileName="MyFile.txt";
If (file_exists($FileName)){
  if (unlink($FileName)
      // deleted
  else
      // error message
 } else {
      // file doesn't exist error message
}
```

23

UPLOADING AND DOWNLOADING FILES

- File Types
  - Text
  - Images
  - Documents
  - Spreadsheets
- Form Tag
  - Method
    - **"post"** for uploads
  - **enctype** attribute
    - **"multipart/form-data"**
    - browser posts multiple sections
      - regular form data
      - file contents

```
<form method="post" enctype="multipart/form-data". . .>
```

24

SELECTING THE FILE

```
<form method="post" enctype="multipart/form-data"… >
```

- Input tags
  - `Hidden: MAX_FILE_SIZE` - maximum bytes
    - Must appear before the `file` input field

```
<input type="hidden" name="MAX_FILE_SIZE" value="25000" … />
```

  - `file:` browse button for navigating to file

```
<input type="file" name="UploadedFile" … />
```

- php.ini
  - file_uploads must be set to 1 or "on"

25

RETRIEVING FILE INFORMATION

- **$_FILES[]**
  - associative autoglobal array storing uploaded files
  - Nested key elements

```
// Error Codes
$_FILES['UploadedFile']['error']

// Temporary location $_FILES['UploadedFile']
['tmp_name']

// Original file name
$_FILES['UploadedFile']['name']

// Size in bytes
$_FILES['UploadedFile']['size']

// File MIME type
$_FILES['UploadedFile']['type']
```


26

STORING THE UPLOADED FILE

- Uploads into temporary location
- Must be moved
- Considerations
  - Immediately Available or Verified First
    - Example: Virus free, appropriate type
    - Sandbox outside of regular web folders
  - Public or Private
    - **Public**: available to any web site visitors
    - **Private**: available to authorized visitors

27

STORING THE UPLOADED FILE

- **move_uploaded_file()**
  - moves the uploaded file to a new location
  - Returns TRUE (succeeds) or FALSE (fails)
  - If file exists, will overwrite it

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

- *$filename* is the contents of `$_FILES['UploadedFile']['tmp_name']`
- *$destination* is the path and filename of new location

28 STORING THE UPLOADED FILE (CONTINUED)

```
$TempName = $_FILES['picture_file']['tmp_name'];
$OrigName = $_FILES['picture_file']['name']

if (move_uploaded_file($TempName, "uploads/" .
    $OrigName)=== FALSE)
  {
    echo "Could not move uploaded file to \"uploads/" .
    htmlentities($OrigName). "\"<br />\n";
  }
else
  {
   chmod("uploads/" . $OrigName, 0644); //0644 ensures everyone can read
   echo "Successfully uploaded \"uploads/" .
    htmlentities($TempName) . "\"<br />\n";
  }
```

Note: htmlentities () converts characters such as single and double quotes to HTML entities

29 DOWNLOADING FILES

- Inside public XHTML directory structure
  - downloaded with an XHTML hyperlink
- Outside public XHTML directory or to display dialog
  1. Tell the script which file to download
  2. Provide the appropriate headers
  3. Send the file
- **`header()`** function
  - Return header information to the Web browser
  - All headers must be sent before web content
  - If after, treated as text not header content
  - Make sure this is done by 1st characters of 1st line - **`<?php`**

Note: Book examples uses "get" method requiring the use of URL tokens

30 DOWNLOADING FILES-HEADERS

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

31

Downloading Files

```
<?php
   $Dir = "files";
   if (isset($_GET['filename'])) {
        $FileToGet = $Dir . "/" . stripslashes($_GET['filename']);
        if (is_readable($FileToGet)) {
             header("Content-Description: File Transfer");
             header("Content-Type: application/force-download");
             header("Content-Disposition: attachment; filename=\"" . $_GET['filename'] . "\"");
             header("Content-Transfer-Encoding: base64");
             header("Content-Length: " . filesize($FileToGet));
             readfile($FileToGet); //Reads file and sends to browser
             // Do not include XHTML or it will become part of downloaded file info
             $ShowErrorPage = FALSE; }
        else
             $ShowErrorPage = TRUE;
   }
   else
        $ShowErrorPage = TRUE;
   If ($ShowErrorPage) {
   ?>
          <!-- XHTML code to display error page -- >
<?php } ?>
```

32

Writing Entire Files

- **file_put_contents()**
  - Returns number of bytes written or FALSE if failure
  - Creates file if nonexistent
  - Overwrites

    ```
    file_put_contents(FileName,NewContent)
    ```
  - Appends

    ```
    file_put_contents(FileName,NewContent,FILE_APPEND)
    ```
- **fwrite()**
  - Returns number of bytes written or FALSE if failure
  - Writes to an open file - used with fopen() and fclose()

    ```
    $file = fopen(FileName,"w");
    echo fwrite($file,NewContent);
    fclose($file);
    ```


33

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

If include path has been defined then

Writing: *FILE_USE_INCLUDE_PATH*
Reading: *USE_INCLUDE_PATH*

34

Reading an Entire File

readfile()
- Reads a file and writes it to the output buffer
- Returns number of bytes read or FALSE if failure

```
echo readfile("sample.txt");
Output:  This is just a sample.
         23
```

file_get_contents()
- Reads entire contents of a file into a string

```
$EntireFile=file_get_contents("sample.txt");
echo $EntireFile;
Output:  This is just a sample.
```

file()
- Reads entire contents of a file into an array
- Recognizes `\n`, `\r`, or `\r\n`
- Use the `explode()` function to assign values to variables

```
print_r(file("sample.txt"));
Output:
    Array(
          [0]=>This is just a sample.
         )
```

35

Open / Close File Streams

- **Stream**
  - channel used for accessing a resource
  - can read from and write to
- **Input Stream**
  - reads data from a resource (such as a file)
- **Output Stream**
  - writes data to a resource
- **Process**
  1. `fopen()`
  2. Read / Write Data
  3. `fclose()`

36

Opening a File Stream

- **fopen()**
  - Opens a handle to a file stream

```
$Handle=fopen("FileName", "mode");
```

- **fclose()**
  - Closes an open file to save space in memory
  - File is "flushed"
  - Allow other processes to read/write from the file

```
fclose($Handle);
```

37

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

38

Opening a File Stream

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

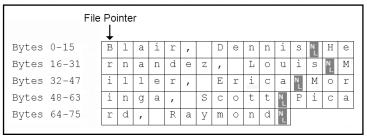


Figure 5-15 Location of the file pointer when the `fopen()` function uses a *mode* argument of "r+"

39

Opening a File Stream

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

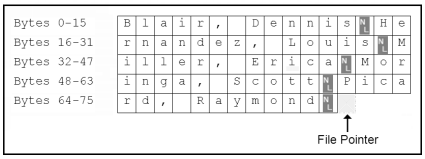


Figure 5-16 Location of the file pointer when the `fopen()` function uses a mode argument of "a+"

40

Locking Files

flock()

- prevents multiple users from modifying a file simultaneously
- Not really:
  - Advisory
  - Only prevents other scripts that use flock()

```
flock($Handle, operation)
    if(flock($Handle,LOCK_EX)){
       //Some Code
       flock($Handle, LOCK_UN)
     }
```

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

41

Reading Data Incrementally

- **fopen()** and **fclose()** – requirement
- **fgets()** function uses the file pointer to iterate through a text file
  - With each function call, file pointer automatically moves to the next line in the text file
  - **fgetc()** - moves to next character

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 XHTML tags the line contains, and then moves the file pointer to the next line
fread($handle, length)	Returns up to *length* characters and moves the file pointer to the next available character
stream_get_line($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

42

Reading Data Incrementally

```
<?php
    $file = fopen("test.txt","r");
    while(! feof($file)){
        echo fgets($file). "<br />";
     }
     fclose($file);
?>
```

The output of the code above will be:
Hello, this is a test file.
There are three lines here.
This is the last line.

THE END

PHP PROGRAMMING WITH MYSQL
2ND EDITION

MODIFIED BY ANITA PHILIPP -SPRING 2012