ISIT307 - WEB SERVER PROGRAMMING

LECTURE 3 – WORKING WITH FILES AND DIRECTORIES

LECTURE PLAN

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

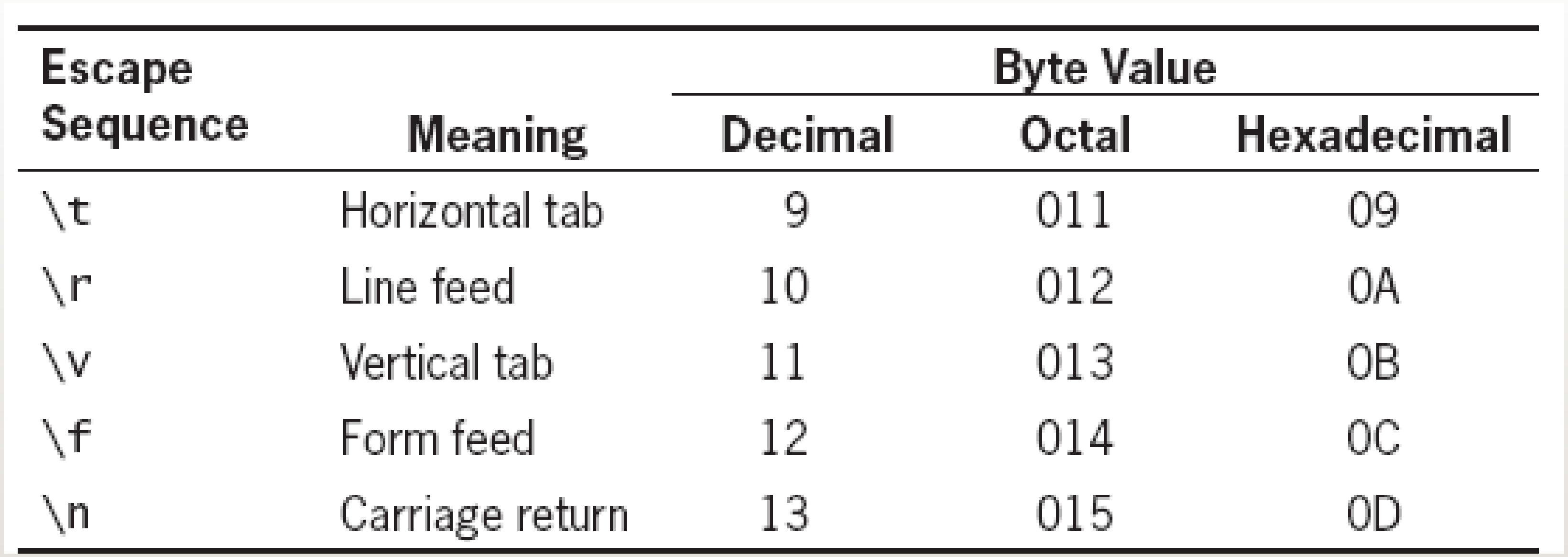
UNDERSTANDING FILE TYPES AND PERMISSIONS

- **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

CONTROL CHARACTERS IN A TEXT FILE



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

PHP Programming with MySQL, 2011, Cengage Learning.

UNDERSTANDING FILE TYPES

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

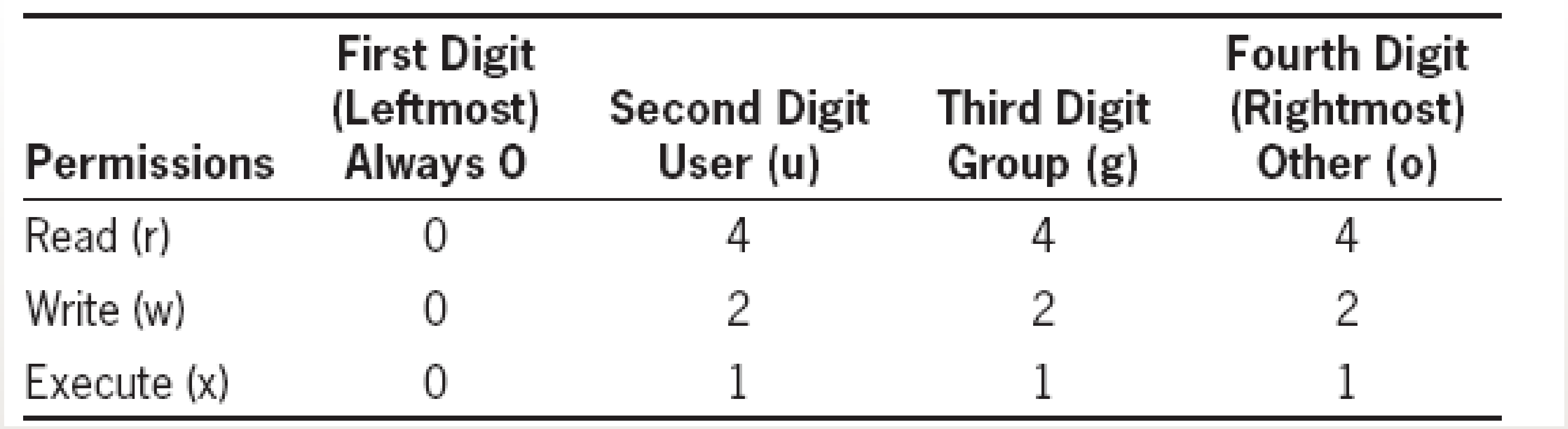
WORKING WITH FILE PERMISSIONS

- 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

- 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



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

PHP Programming with MySQL, 2011, Cengage Learning.

WORKING WITH FILE PERMISSIONS

- 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
- For example

```
chmod("myfile.txt", 0754)
```

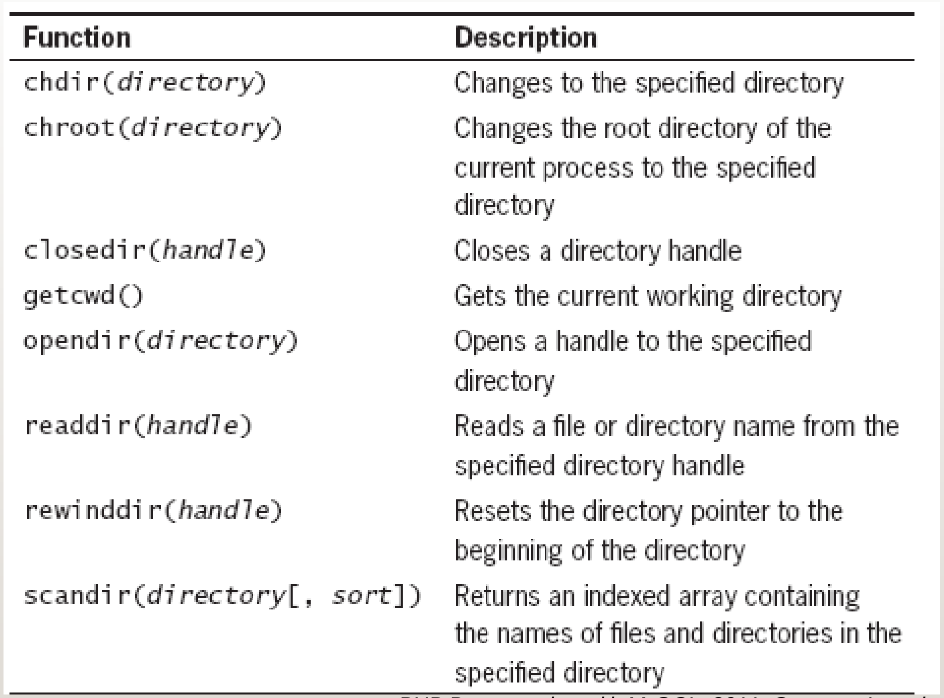
CHECKING PERMISSIONS

- 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 modulo operator with an octal value of 01000
- The `decoct()` function converts a decimal value to an octal value

```
$perms = fileperms($testfile);
$perms = decoct($perms % 01000);
echo "file permissions for $testfile: 0" .
                          $perms . "<br />\n";
```

READING DIRECTORIES

- The following table lists the PHP functions for working with 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

PHP Programming with MySQL, 2011, Cengage Learning

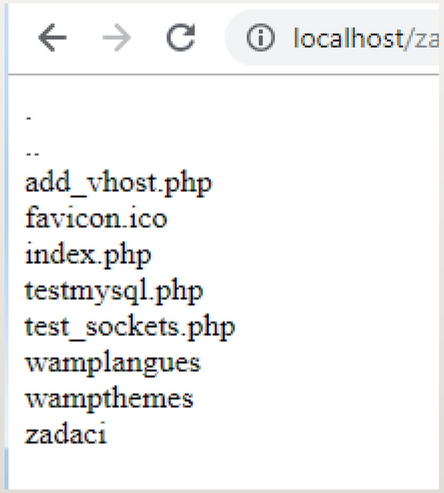
READING DIRECTORIES

- The `opendir()` function is used to open a directory and 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
- Each time the `readdir()` is called, it returns the current file or directory name and move the directory pointer to the next

READING DIRECTORIES – EXAMPLE (1)

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



```
.
..
add_vhost.php
favicon.ico
index.php
testmysql.php
test_sockets.php
wamplangues
wampthemes
zadaci
```

READING DIRECTORIES

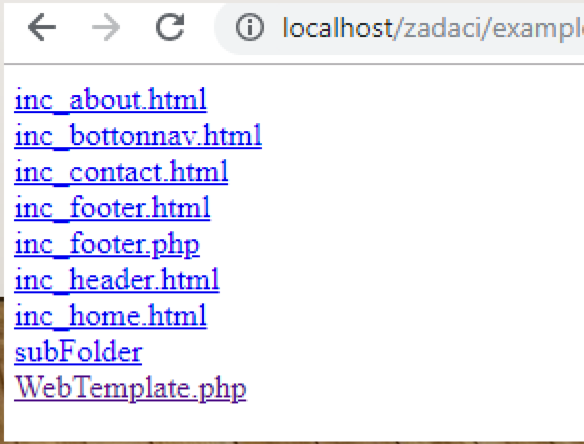
- The PHP scripting engine returns the navigation shortcuts ("." for current directory, and ".." for parent directory) when it reads a directory
- The `strcmp()` function can be used to exclude those entries

READING DIRECTORIES – EXAMPLE (2)

```
<?php
$Dir = ".";     //$Dir = "name_of_dir";
$DirOpen = opendir($Dir);
while ($CurFile = readdir($DirOpen)) {
   if ((strcmp($CurFile, '.') != 0) && (strcmp($CurFile, '..') != 0))

      echo "<a href=\"./" . $CurFile ."\">" . $CurFile . "</a><br/>\n";

      // echo "<a href=\"name_of_dir/" . $CurFile . "\">" . $CurFile .
            "</a><br />\n";
}
closedir($DirOpen);
?>
```

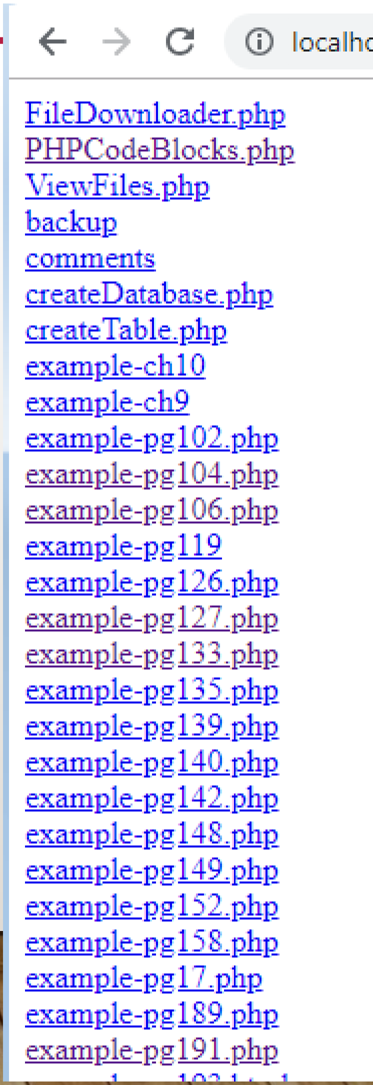


```
inc_about.html
inc_bottonnav.html
inc_contact.html
inc_footer.html
inc_footer.php
inc_header.html
inc_home.html
subFolder
WebTemplate.php
```

READING DIRECTORIES – EXAMPLE(3)

- The `scandir()` function returns the names of the entries in a directory to an array sorted in ascending alphabetical order (if we pass 1 as second argument the entries will be sorted in descending alphabetical order)

```
…
$Dir = ".";
$DirEntries = scandir($Dir);    // scandir($Dir, 1);
foreach ($DirEntries as $Entry) {
    if ((strcmp($Entry, '.') != 0) && (strcmp($Entry, '..') != 0))
        echo "<a href=\"./" . $Entry . "\">" . $Entry .
                                                      "</a><br />\n";
}
```



localhc

FileDownloader.php
PHPCodeBlocks.php
ViewFiles.php
backup
comments
createDatabase.php
createTable.php
example-ch10
example-ch9
example-pg102.php
example-pg104.php
example-pg106.php
example-pg119
example-pg126.php
example-pg127.php
example-pg133.php
example-pg135.php
example-pg139.php
example-pg140.php
example-pg142.php
example-pg148.php
example-pg149.php
example-pg152.php
example-pg158.php
example-pg17.php
example-pg189.php
example-pg191.php

CREATING DIRECTORIES

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

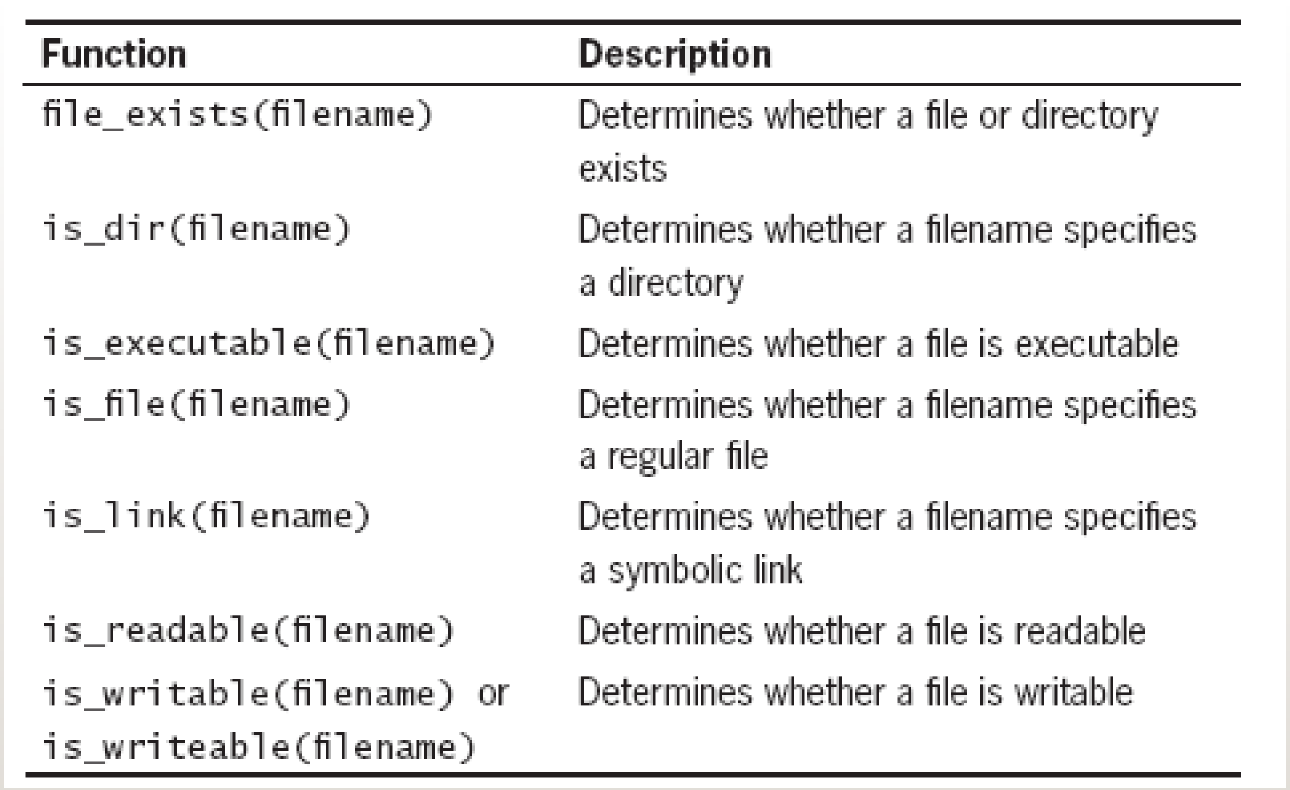
```
mkdir("volunteers");
```

CREATING DIRECTORIES

- 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");
```

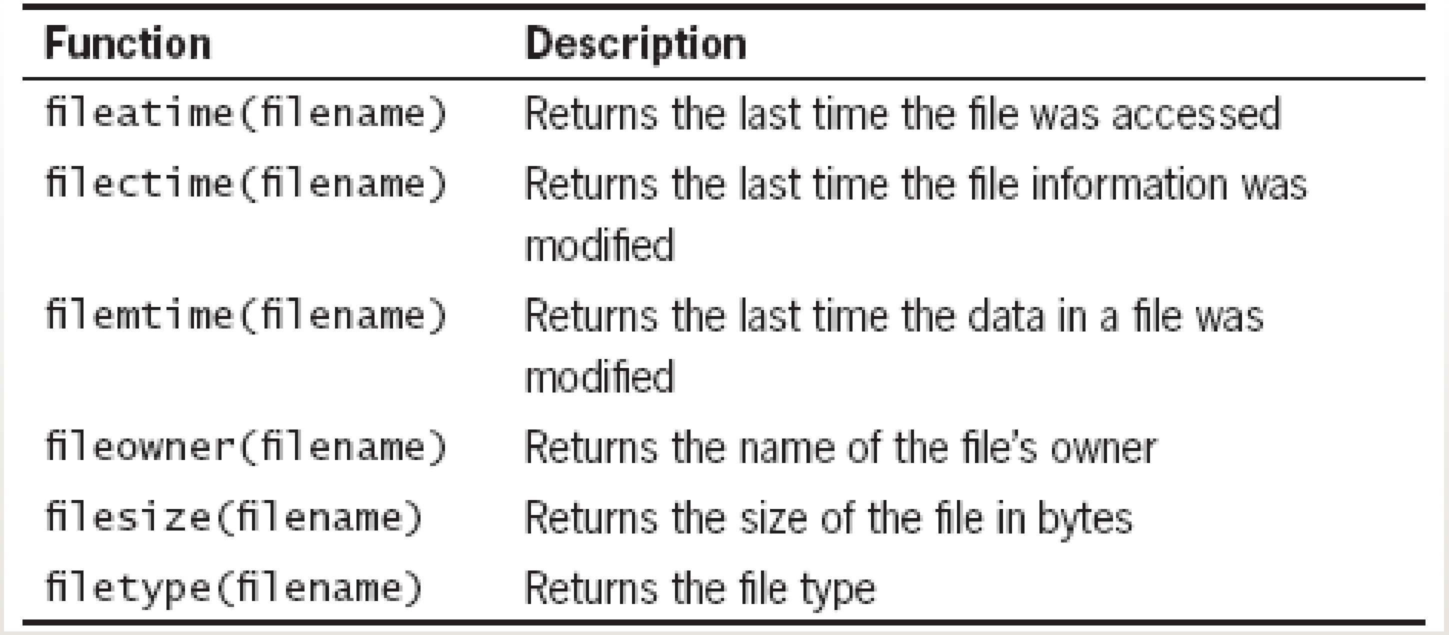
PHP FILE AND DIRECTORY FUNCTIONS



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

PHP Programming with MySQL, 2011, Cengage Learning.

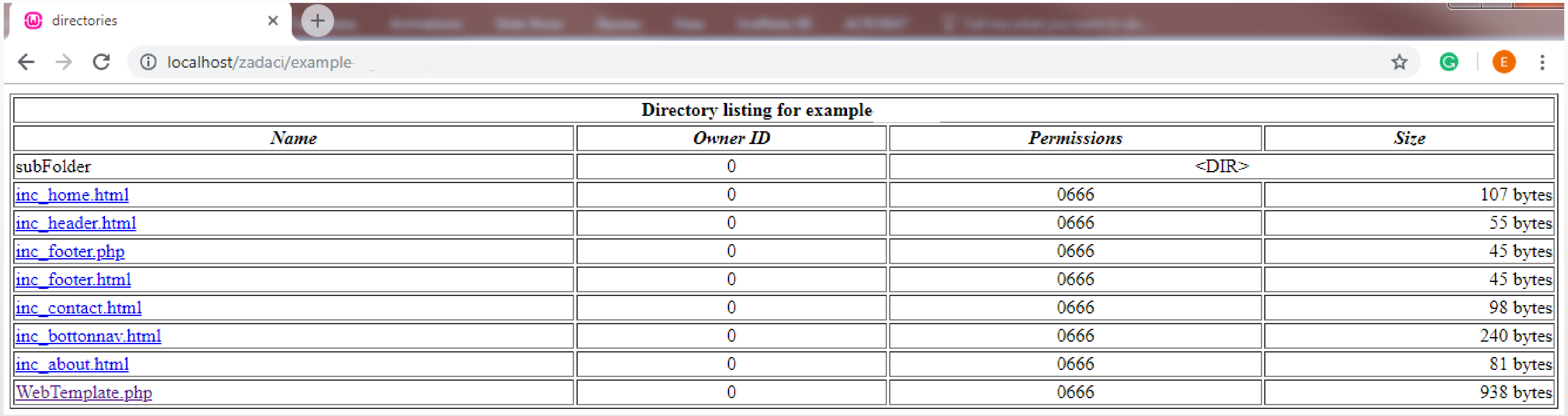
FILE AND DIRECTORY INFORMATION FUNCTIONS



Function	Description
`fileatime(filename)`	Returns the last time the file was accessed
`filectime(filename)`	Returns the last time the file 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

PHP Programming with MySQL, 2011, Cengage Learning.

OBTAINING FILE AND DIRECTORY INFORMATION - EXAMPLE



Directory listing for example			
Name	***Owner ID***	***Permissions***	***Size***
subFolder	0	<DIR>	
inc_home.html	0	0666	107 bytes
inc_header.html	0	0666	55 bytes
inc_footer.php	0	0666	45 bytes
inc_footer.html	0	0666	45 bytes
inc_contact.html	0	0666	98 bytes
inc_bottonnav.html	0	0666	240 bytes
inc_about.html	0	0666	81 bytes
WebTemplate.php	0	0666	938 bytes

UPLOADING AND DOWNLOADING FILES

- Web applications allow visitors to upload/download files to and 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 HTML 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

- 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 input 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 `$_FILES` autoglobal array
- The `$_FILES[]` array element 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`
  - `$_FILES['picture_file']['name']`
    `// Contains the name of the original file`
  - `$_FILES['picture_file']['size']`
    `// Contains the size of the uploaded file in bytes`
  - `$_FILES['picture_file']['type']`
    `// Contains the type of the file`

STORING THE UPLOADED FILE

- Uploaded files have two considerations, before are moved to permanent position:
  - whether the file should be immediately available or verified first
  - is the file public (freely available to anyone visiting the Web site) or private (only available to authorized visitors)

STORING THE UPLOADED FILE

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

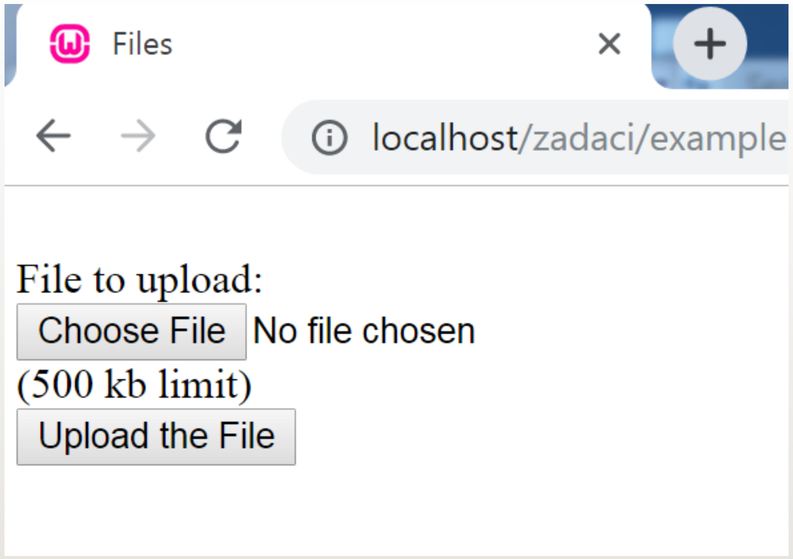
```
move_uploaded_file($filename, $destination)
```

- where *`$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 - EXAMPLE



File to upload:

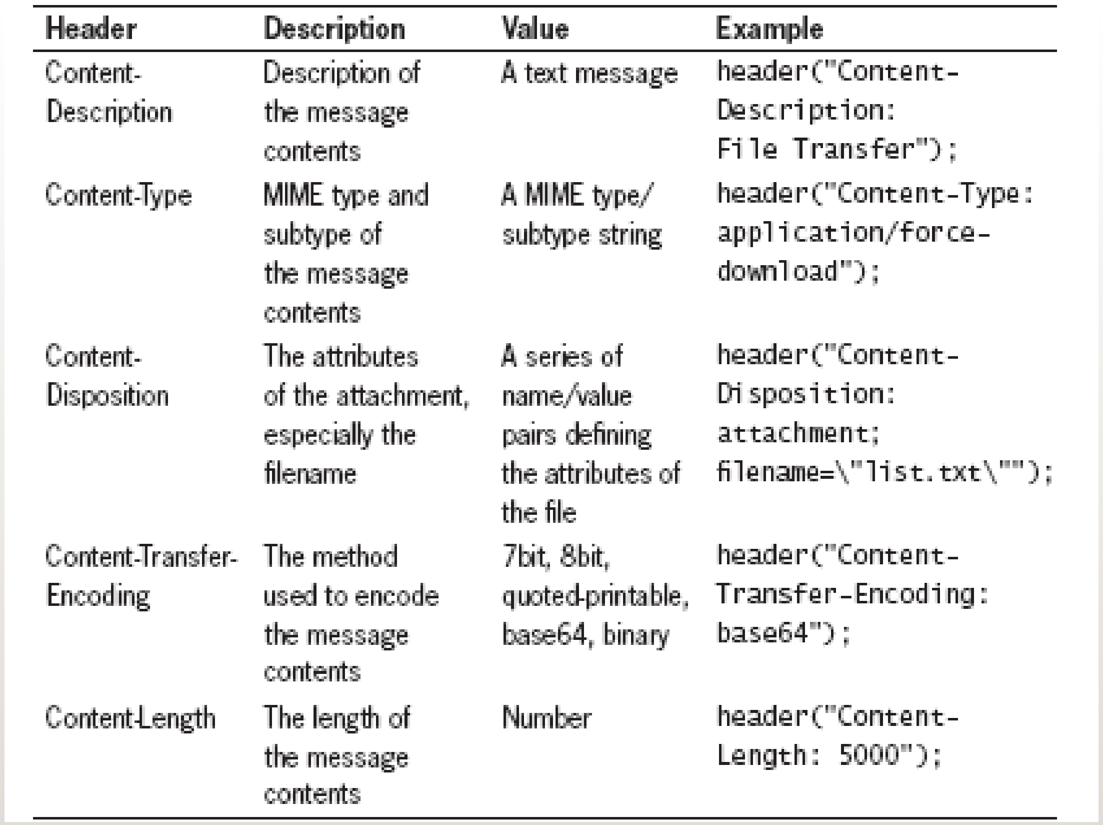
No file chosen

(500 kb limit)

DOWNLOADING FILES

- Files in the public HTML directory structure can be downloaded with an HTML hyperlink
- Files outside the public HTML directory require a three-step process:
  - Tell the script which file to download (can be used URL tokens)
  - Provide the appropriate headers
  - Send the file
- The headers must be send prior to any web content
- The `header()` function can be used to return header information to the Web browser

CONTENT HEADERS FOR DOWNLOADING A FILE



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");`

PHP Programming with MySQL, 2011, Cengage Learning.

DOWNLOADING FILES - EXAMPLE

WRITING AN ENTIRE FILE

- 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 syntax is:

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

WRITING AN ENTIRE FILE - EXAMPLE

```
$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)

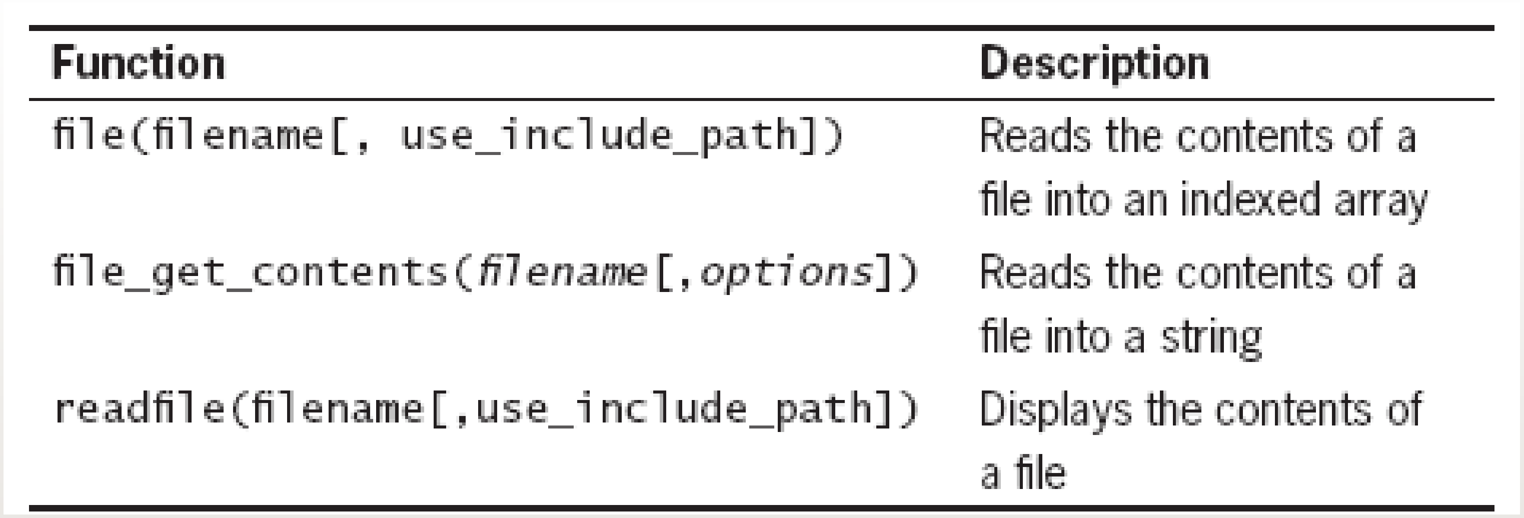
- If no data was written to the file, the function returns a value of `0`
- We can use the return value to determine whether data was successfully written to the file

```
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

- There are other arguments that can be pass to the `file_put_contents()`
  - The `FILE_USE_INCLUDE_PATH` constant searches for the specified filename in the path that is assigned to the `include_path` directive in the 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

PHP Programming with MySQL, 2011, Cengage Learning.

READING AN ENTIRE FILE (CONTINUED)

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

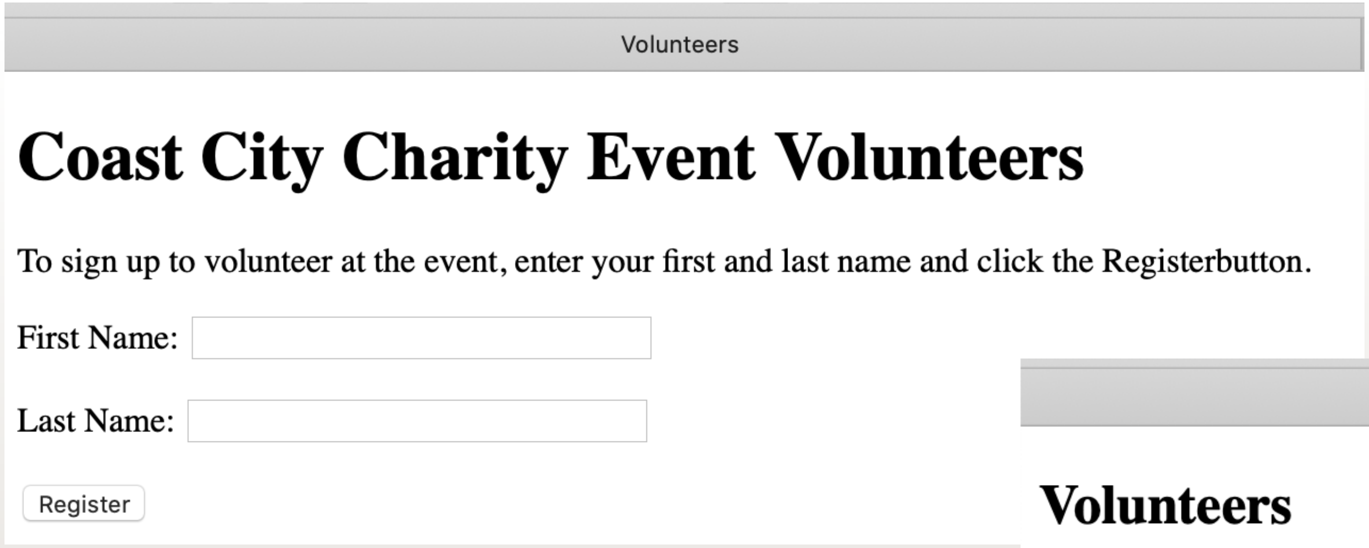
```
$myfile = file_get_contents("my_file.txt");
echo $ myfile;
```

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

```
readfile(”my_file.txt");
```

- 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`, to assign the lines into the elements in the array

WRITING/READING AN ENTIRE FILE - EXAMPLE

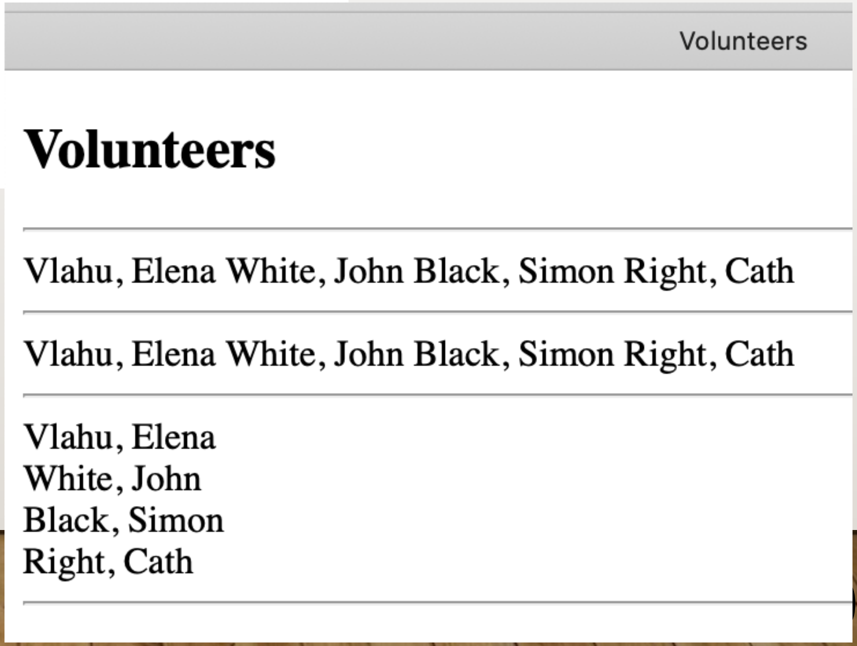


Coast City Charity Event Volunteers

To sign up to volunteer at the event, enter your first and last name and click the Registerbutton.

First Name:

Last Name:



Volunteers

Vlahu, Elena White, John Black, Simon Right, Cath

Vlahu, Elena White, John Black, Simon Right, Cath

Vlahu, Elena
White, John
Black, Simon
Right, Cath

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
- Using a file stream involves 3 steps:
  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

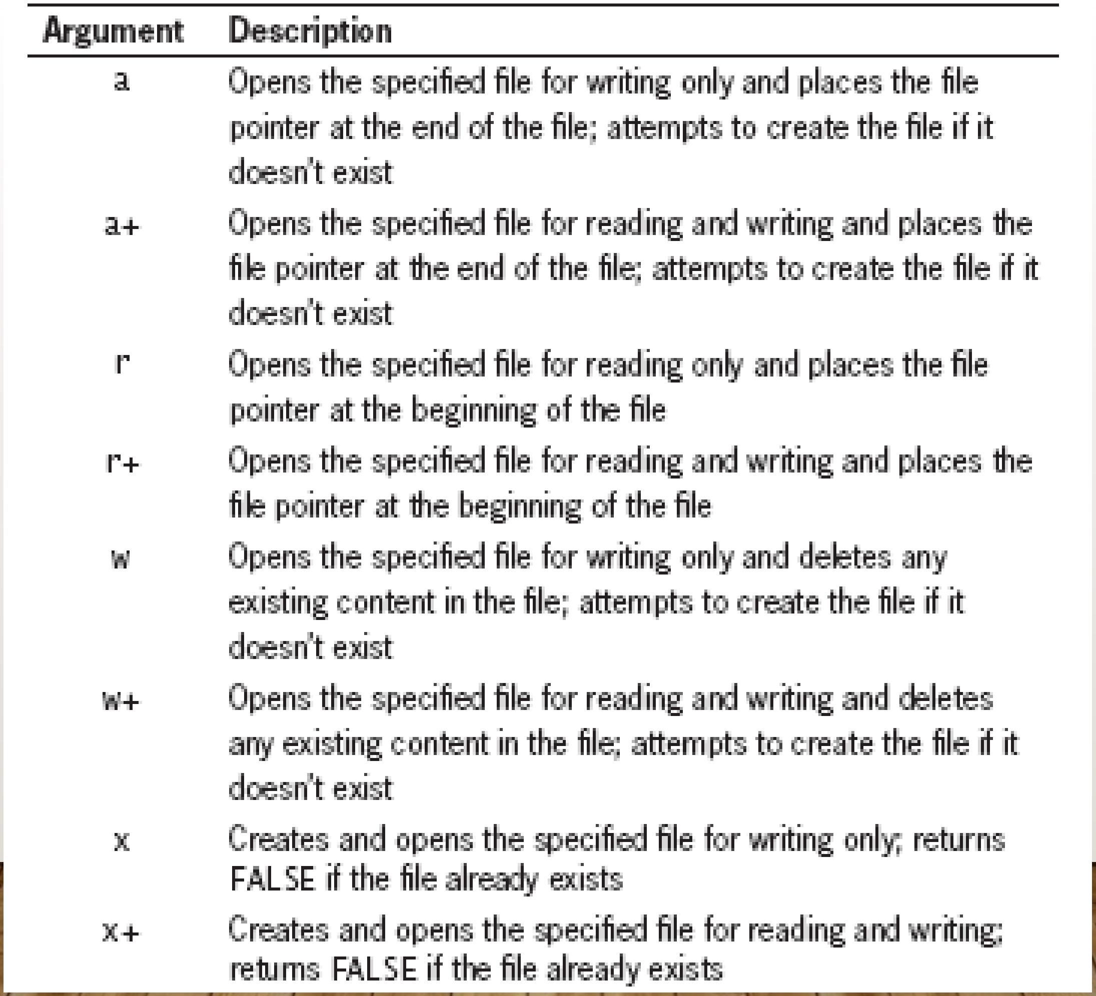
OPENING A FILE STREAM

- 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", "method");
```

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

METHOD ARGUMENTS FOR THE FOPEN() FUNCTION

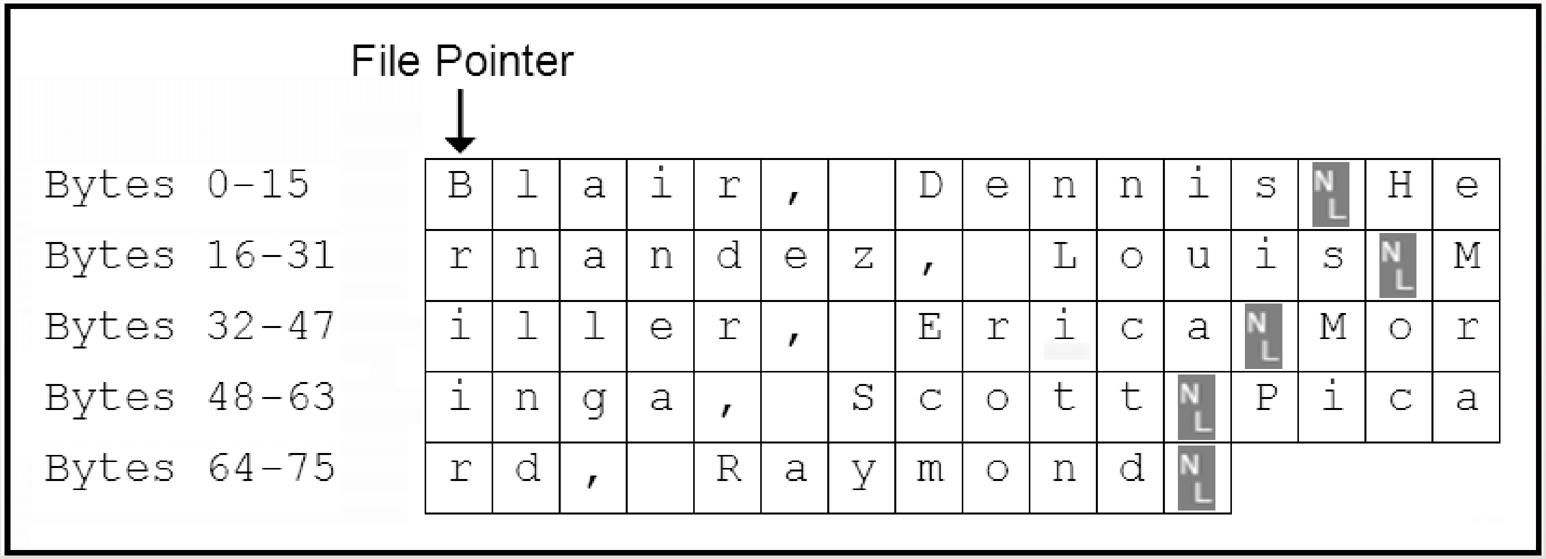


Argument	Description
a	Opens the specified file for writing only and places the file pointer at the end of the file; attempts to create the file if it doesn't exist
a+	Opens the specified file for reading and writing and places the file pointer at the end of the file; attempts to create the file if it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

PHP Programming with MySQL, 2011, Cengage Learning.

OPENING A FILE STREAM (CONTINUED)

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

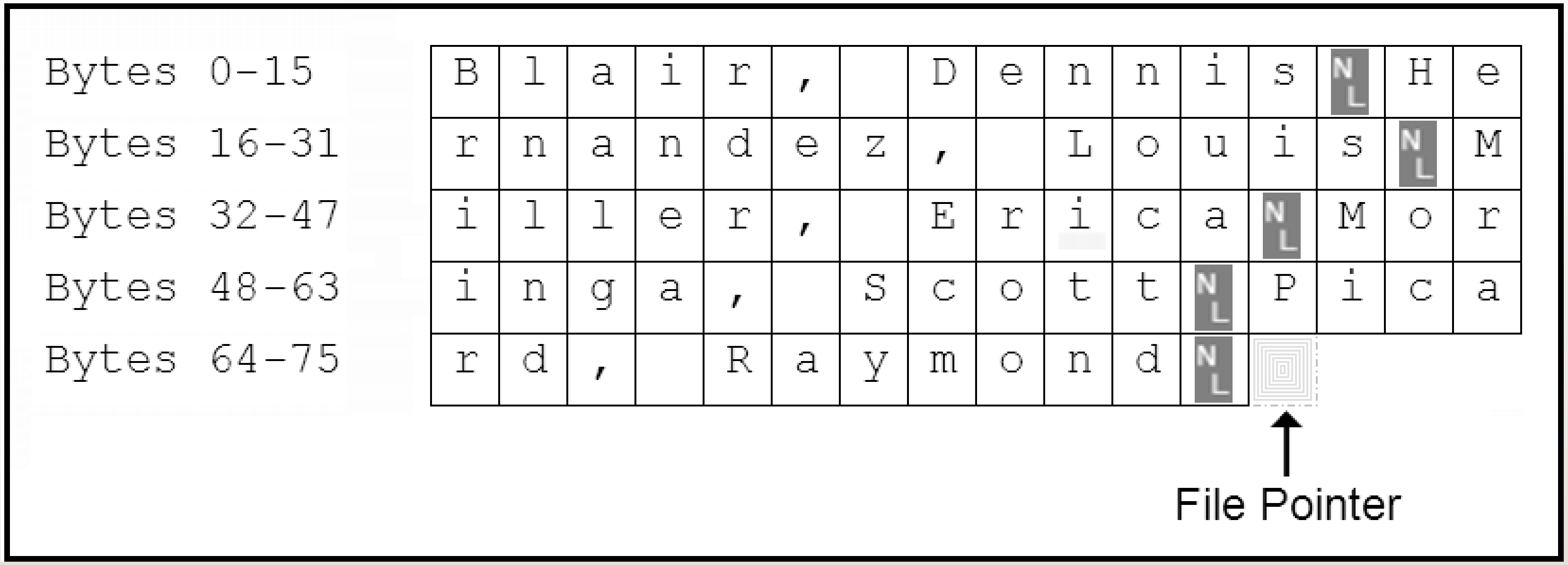


Bytes 0-15	B	l	a	i	r	,		D	e	n	n	i	s	NL	H	e
Bytes 16-31	r	n	a	n	d	e	z	,		L	o	u	i	s	NL	M
Bytes 32-47	i	l	l	e	r	,		E	r	i	c	a	NL	M	o	r
Bytes 48-63	i	n	g	a	,		S	c	o	t	t	NL	P	i	c	a
Bytes 64-75	r	d	,		R	a	y	m	o	n	d	NL				

PHP Programming with MySQL, 2011, Cengage Learning.

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				

PHP Programming with MySQL, 2011, Cengage Learning.

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`

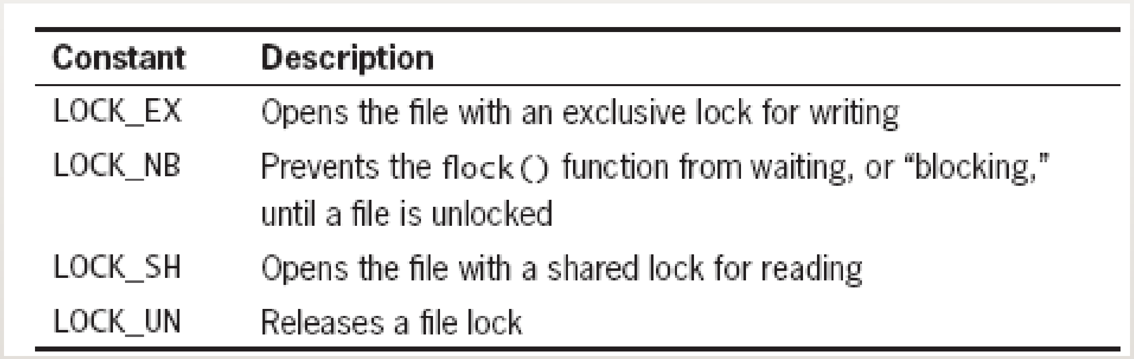
EXAMPLE

```php
$VolunteersFile = fopen("volunteers.txt", "ab");
fwrite($VolunteersFile, "Blair, Dennis\n");
fwrite($VolunteersFile, "Hernandez, Louis\n");
fwrite($VolunteersFile, "Miller, Erica\n");
fwrite($VolunteersFile, "Morinaga, Scott\n");
fwrite($VolunteersFile, "Picard, Raymond\n");
fclose($VolunteersFile);
```

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)
```

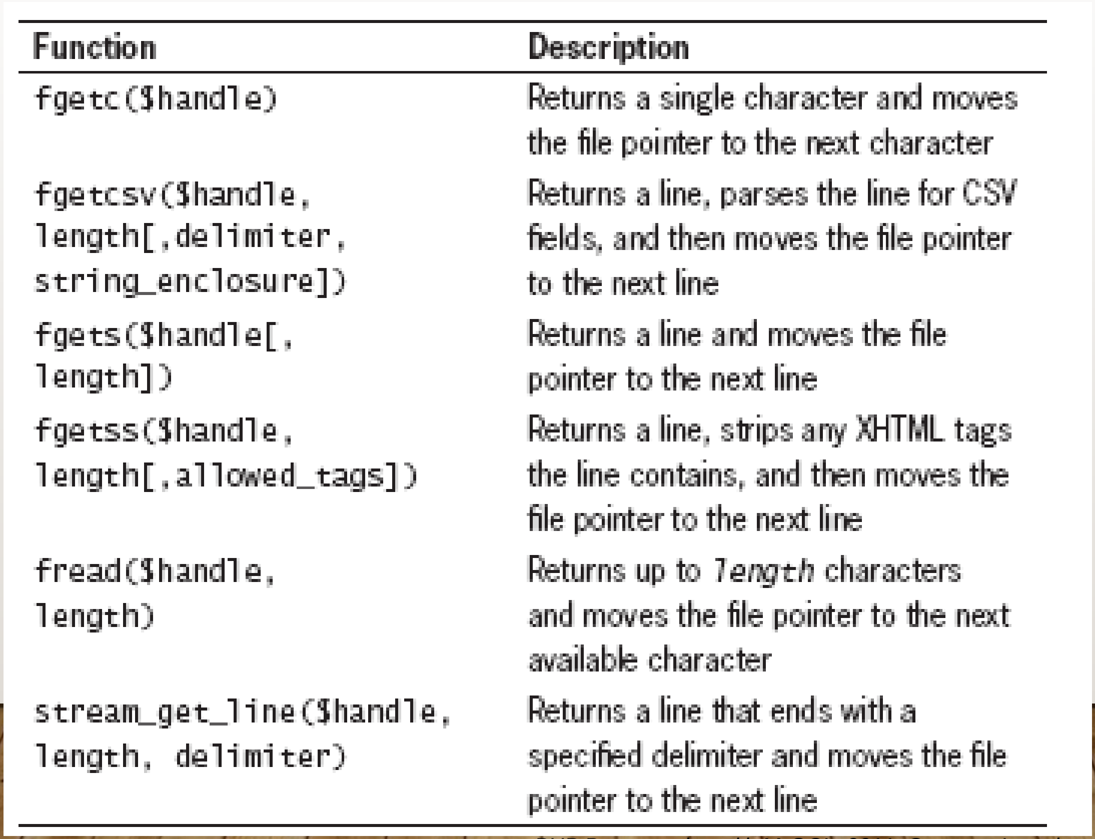


Constant	Description
LOCK_EX	Opens the file with an exclusive lock for writing
LOCK_NB	Prevents the flock() function from waiting, or "blocking," until a file is unlocked
LOCK_SH	Opens the file with a shared lock for reading
LOCK_UN	Releases a file lock

PHP Programming with MySQL, 2011, Cengage Learning.

READING DATA INCREMENTALLY

- The `fgets()` function uses the file pointer to 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 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

PHP Programming with MySQL, 2011, Cengage Learning.

READING DATA INCREMENTALLY (CONTINUED)

- You must use `fopen()` and `fclose()` with the functions listed in Table
- The `feof()` function returns a value of TRUE when a file pointer reaches the end of a file
  - it accepts a single argument containing the handle for the open file

READING/WRITING INCREMENTALLY IN THE FILE – EXAMPLE

- Visitors feedback

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 - EXAMPLE

```
if (copy("$Source/$Entry", "$Destination/$Entry"))
   echo "One file copied\n";
else
   echo "Could not copy the file \n";
```

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 (it does not work unless the directory is empty)
- 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