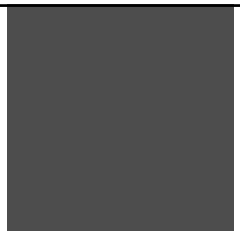


Web Application Development : PHP Strings, Files, Directories and Arrays

Week 3

SWINBURNE UNIVERSITY OF TECHNOLOGY

Content of this Lecture

- Handling form submissions and processing
- Manipulating, parsing and comparing strings
- Managing files and directories
- Working with files: open, close, write, read
- Working with arrays
- Associative and multidimensional arrays

2

Handling Form Submissions

- Form data is submitted in `name=value` pairs, based on the `name` and `value` attributes of each element

```
<form action="processOrder.php" method="get" > ...
<input type="text" name="book_title" value="technical" />
<input type="text" name= "number_of_copies" value= "1" />
 ...
</form>
```

- A query string is a set of `name=value` pairs appended to a target URL. Each `name=value` pair is separated with ampersands (&)

```
books_title=technical&number_of_copies=1
```

- A question mark (?) and a query string are automatically appended to the URL of a server-side script for any forms that are submitted with the `get` method

```
processOrder.php?books_title=technical&number_of_copies=1
```

3

Determining if Form Variables Contain Values

- Use the `isset()` function to determine whether a variable has been declared and initialised (or "set")
- Use the `empty()` function to determine whether a variable is empty
- Use the `is_numeric()` function to test whether a variable contains a numeric string

```
if (isset($_GET['height']) && isset($_GET['weight'])) {
  if (is_numeric($_GET['weight']) &&
    is_numeric($_GET['height'])) {
      $BodyMass = $_GET['weight'] / ($_GET['height']
             * $_GET['height']) * 703;
      printf("<p>Your body mass index is %d.</p>", $BodyMass);
   }
   else echo "<p>You must enter numeric values!</p>";
}
```

Note: Use `$_POST` when `method=post` is used instead of `get` for `form` submission.

4

Using `mail()` Function

- The syntax for the `mail()` function is:

```
mail(recipient(s), subject, message[, additional_headers])
```

- The `mail()` function returns a value of true if a message was delivered successfully or false if it was not

```
$to = "cchua@swin.edu.au";
$subject = "This is the subject";
$message = "This is the message.";
$headers = "From: Caslon Chua <cchua@swin.edu.au>";
mail($to, $subject, $message, $headers);
```

5

Constructing Text Strings

- A text string contains zero or more characters surrounded by double or single quotation marks
- Text strings can be used as literal values or assigned to a variable

```
echo "<p>Dr. Livingstone, I presume?</p>";
$explorer = "Henry M. Stanley";
echo $explorer;
```

- To include a quoted string within a literal string surrounded by double (single) quotation marks, you surround the quoted string with single (double) quotation marks

```
$explorerQuote = '<p>"Dr. Livingstone, I presume?"</p>';
$explorerQuote = "<p>'Dr. Livingstone, I presume?'</p>";
```

6

Combining Strings

- Concatenation operator **.**

```
$destination = "Paris";
$location = "France";
$destination = "<p>" . $destination . " is in
   France.</p>";
echo $destination;
```

- Concatenation assignment operator **.=**

```
$destination = "<p>Paris";
$destination .= " is in France.</p>";
echo $destination;
```

7

Adding Escape Characters and Sequences

- An escape character tells the compiler or interpreter that the character that follows it has a special purpose
- In PHP, the escape character is the backslash \

```
echo '<p>Marilyn Monroe\'s real name was Norma Jean
Baker.</p>'; // output >>> Marilyn Monroe's real name was Norma Jean Baker.
echo "<p>\"Dr. Livingstone, I presume?\" asked Henry M.
Stanley.</p>"; // output >>> "Dr. Livingstone, I presume?" asked Henry M. Stanley.
```

- Do not add a backslash before an apostrophe if you surround the text string with double quotation marks

```
echo "<p>Marilyn Monroe's real name was Norma Jean
Baker.</p>"; // output >>> Marilyn Monroe's real name was Norma Jean Baker.
```

8

Adding Escape Characters and Sequences

- The escape character combined with one or more other characters is called an escape sequence

Table of PHP escape sequences within double quotation marks

Escape Sequence	Description
`\\`	Inserts a backslash
`\$`	Inserts a dollar sign
`\r`	Inserts a carriage return
`\"`	Inserts a double quotation mark
`\t`	Inserts a horizontal tab
`\n`	Inserts a new line
`\regular expression`	Inserts a character in hexadecimal notation that matches the regular expression

9

Simple and Complex String Syntax

- Simple string syntax uses the value of a variable within a string by including the variable name inside a text string with double quotation marks

```
$vegetable = "broccoli";
echo "<p>Do you have any $vegetable?</p>";
```

How about: `echo "<p>Do you have any $vegetables?</p>";`

```
  //causes an error, variable not declared.
```

- When variables are placed within curly braces inside of a string, it is called complex string syntax

```
$vegetable = "carrot";
echo "<p>Do you have any {$vegetable}s?</p>";
  //output is: Do you have any carrots?
```

10

Comparing Strings using Comparison Operators

```
$loc01 = "Miami";
$loc02 = "Havana";
if ($loc01 == $loc02) echo "<p>Same location.</p>";
else echo "<p>Different location.</p>";
```

```
$firstLetter = "A";
$secondLetter = "B";
if ($secondLetter > $firstLetter)
  echo "<p>The second letter is higher in the
       alphabet than the first letter.</p>";
else
  echo "<p>The second letter is lower in the
       alphabet than The first letter.</p>";
```

11

ASCII American Standard Code for Information Interchange

- Numeric representations of English characters
- ASCII values range from 0 to 255
- Lowercase letters are represented by the values 97 (“a”) to 122 (“z”)
- Uppercase letters are represented by the values 65 (“A”) to 90 (“Z”)
- Since lowercase letters have higher values than uppercase letters, they are evaluated as being “greater” than the uppercase letters

Note: UTF-8 is a strict superset of ASCII with the same physical encoding for ASCII characters

12

String Comparison Functions

- The `strcasecmp()` function performs a case-insensitive comparison of two strings
- The `strcmp()` function performs a case-sensitive comparison of two strings
- Most string comparison functions compare strings based on their ASCII values: returns <0 (if smaller); =0 (if same); >0 (if larger)
- The `similar_text()` function returns the number of characters that two strings have in common
- The `levenshtein()` function returns the number of characters you need to change for two strings to be the same

13

Parsing Strings

- The `strlen()` function returns the total number of characters in a string

```
echo strlen(' ab cd '); // 7
```

- The `str_word_count()` function returns the number of words inside a string

```
$title = "The Cask of Amontillado";
echo "<p>" . str_word_count($title) . " words in total.</p>";
```

- The `strpos()` function performs a case-sensitive search and if found, returns the position of the first occurrence of one string (the 2nd parameter) in another string (the 1st parameter); otherwise, returns `false`

```
$email = "president@whitehouse.gov";
echo strpos($email, '@'); // 9
echo strpos($email, 'p'); // 0 for the 1st character
```

14

Finding Substrings

- The `strchr()` function searches the first occurrence of a string (the 2nd parameter) inside another string (the 1st parameter), and returns the rest of the string (from the matching point), or `false`, if the string to search for is not found

```
echo strchr("Hello world!!!","world"); // world!!!
```

- The `strrchr()` function searches from the end of a string
- The `substr()` function returns a substring from a string (the 1st parameter). The starting position is specified by the 2nd parameter and the length is specified by the optional 3rd parameter.

```
$email = "president@whitehouse.gov";
$nameEnd = strpos($email, "@");
echo substr($email, 0, $nameEnd); // president
echo substr($email, $nameEnd+1); // whitehouse.gov
```

15

Replacing Substrings

- The `str_replace()` / `str_ireplace()` functions perform case-sensitive / case-insensitive replacement of all occurrences of a substring (the 1st parameter) by a replacement substring (the 2nd parameter) in a string (the 3rd parameter)

```
$email = “john@swin.edu.au";
echo str_replace(“john", “mary", $email);// mary@swin.edu.au
```

- The substr_replace(*string, replacement_string, start_position[, length]*) function perform replacement of a substring starting from the specified start position (and the optional length)

16

Dividing Strings into Smaller Pieces

- The `strtok()` function breaks a string into smaller strings, called **tokens** one by one

```
$variable = strtok(string, separators);
```

- After the first call, this function only needs the separators in the subsequent calls as it keeps track of where it is in the current string

```
<?php
$string = "Hello world. Beautiful day today.";
$token = strtok($string, ". ");

while ($token != false)
{
  echo "$token<br />";
  $token = strtok(". ");
}
?>
```

Output:
Hello
world
Beautiful
day
today

17

Converting Between Strings and Arrays

- The `str_split()` function splits each character (or each set of fixed number of characters) in a string into an array element

```
$array = str_split(string[, length]);
```

- The `explode()` function splits a string into an indexed array at a specified separator

```
$array = explode(separator, string);
```

- Note: the 1st parameter is separator, where the characters in the separator are treated as a substring, not individual characters.

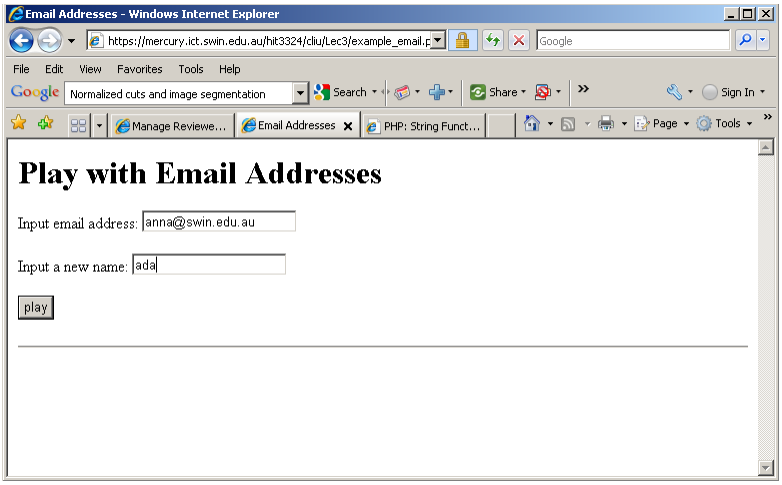
```
$string = "Hello world. Beautiful day today.";
$sentence = explode(". ", $string);
```

Output:
Hello world
Beautiful day today.

- The `implode()` function combines an array's elements into a single string, separated by specified characters

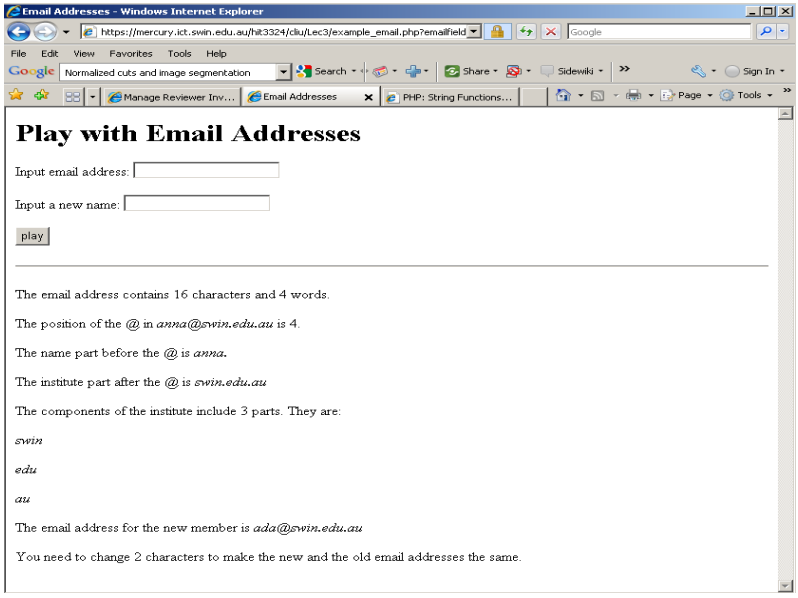
18

Examples – Play with Email Addresses



19

Examples – Some String Operations



20

How to Implement These Operations

```
<?php
  if(isset($_GET['emailfield']) && isset($_GET['namefield'])) {
     $email = $_GET['emailfield'];
     echo "<p> The email contains " . strlen($email) . " chars and " . str_word_count($email) . " words.</p>";
     $newName = $_GET['namefield'];
     $nameEnd = strpos($email, "@");
     echo "<p> The position of the @ in <em>$email</em> is $nameEnd.</p>";
     $name = substr($email, 0, $nameEnd);
     echo "<p> The name part before the @ is <em>$name.</em></p>";
     $institute = substr($email, $nameEnd+1);
     echo "<p> The institute part after the @ is <em>$institute</em></p>";
     $instSegment = explode(".", $institute);
     $cnt = count($instSegment);
     echo "<p> The components of the institute include $cnt parts. They are:</p>";
     for ($i=0; $i<$cnt; $i++)  echo "<p><em>" . $instSegment[$i] . "</em></p>";
     echo "<p> Email for the new member is <em>". str_replace($name, $newName, $email). "</em></p>";
     echo "You need to change " . levenshtein($name, $newName) . " characters to make the new and the old
email addresses the same.</p>";
  }
?>
```

21

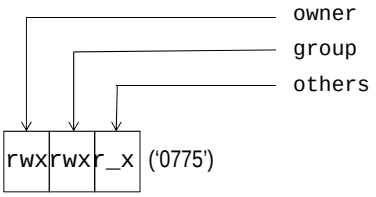
More String Functions

- http://www.w3schools.com/php/php_ref_string.asp
- http://php.net/manual/en/ref.strings.php

22

Windows & Unix/Linux File and Directory

- File is used to store data permanently for retrieval later
  - May use different end of line '\r' '\n' characters
- Directory (in Unix/Linux), also referred to as a folder (in Windows) is a virtual container within an electronic file system
- Path delimiting character
  - Windows uses '\', e.g. 'hit3323\assign1'
  - Unix/Linux uses '/', e.g. 'hit3323/assign1'
- Unix/Linux has access permissions for directories/files, e.g.



23

PHP Directory Functions

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

24

Reading Directories

- Open a handle to the directory with the **`opendir()`** function
- To iterate through the entries in a directory, use the **`readdir()`** function to return the file and directory names from the open directory
- Use the **`closedir()`** function to close a directory handle

```
$dir = "../data";
$dirOpen = opendir($dir);
while ($curFile = readdir($dirOpen)) {
    echo $curFile , "<br />";
}
closedir($dirOpen);
```

- Use the **`scandir()`** function to returns an indexed array containing the names of files and directories in the specified directory

```
$dir = "../data";
$dirEntries = scandir($dir);
foreach ($dirEntries as $entry) {
          echo $entry , "<br />";
}
```

25

Creating Directories

- The **`mkdir()`** function creates a new directory. It returns true on success, or false on failure.

```
mkdir(path, mode[, recursive, context]);
```

- On mercury, suppose the current directory is username/hit3323/www/htdoc. We create

```
mkdir("Lec2", 0777);
mkdir("../data/assignment1", 02770);
```

Note: we need to change mode of username/hit3323/www/data

```
…/www> chmod 02770 data
```

For mercury File Permissions, see:
https://csg.ict.swin.edu.au/livecsg/help/Mercury_Web_Server

26

Obtaining File and Directory Information

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

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
filesize(*filename*)	Returns the size of the file in bytes

27

Copying, Renaming and Removing

- Use the **`copy()`** function to copy a file with PHP, it returns true if successful or false if not

```
copy(source, destination)
```

- Use the **`rename()`** function to rename a file or directory with PHP, it returns true if successful or false if not

```
rename(old_name, new_name)
```

- Use the **`unlink()`** function to delete a file and the **`rmdir()`** function to delete a directory, they return 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 copy/rename/delete it

28

Opening and Closing a File

- 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
- Usually a three stage process:
  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

29

Opening and Closing a File

- 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

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

```
$bowlersFile = fopen("bowlers.txt", "a");
$newBowler = "Doe, John\n";
fwrite($bowlersFile, $newBowler);
fclose($bowlersFile);
```

30

Mode of `fopen()`

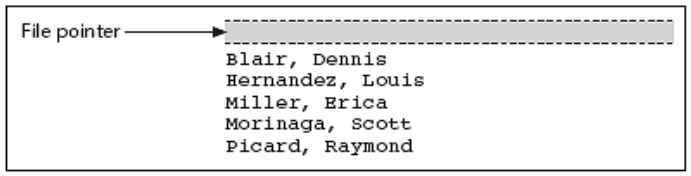
Mode parameter of the **`fopen()`** function

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 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

31

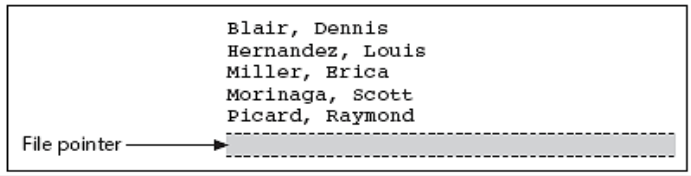
File Pointer

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



Location of the file pointer when the **`fopen()`** function uses ***mode*** "r+"

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



Location of the file pointer when the **`fopen()`** function uses ***mode*** "a+"

32

Writing Data to a File

- PHP supports two basic functions for writing data to text files:
  - **`file_put_contents()`** function *writes an entire file* or *appends* a text string to a file
  - **`fwrite()`** function *incrementally writes* data to a text file
- Escape sequences used to identify the end of a line:
  - UNIX/Linux platforms use the `\n` carriage return
  - Macintosh platforms use `\r` carriage return *(OS X is Linux based)*
  - Windows uses both the `\n` newline and the `\r` carriage return escape sequence `\n\r`

33

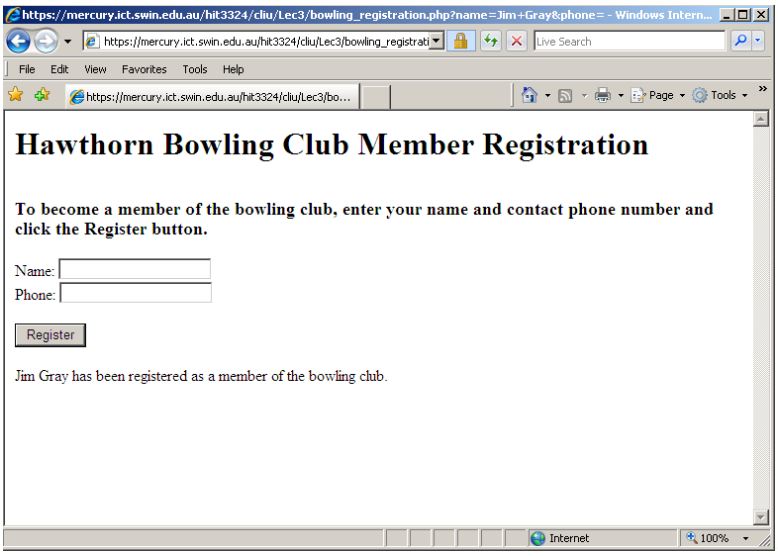
Writing Data Once into a File

- The **`file_put_contents()`** function *writes an entire file* or *appends* a text string to a file

  `file_put_contents(`*`filename`*`, `*`string`*`[, `*`options`*`])`
- Note: no file open/close needed - integrated
- For the 3rd parameter
  - 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

34

Example – Bowling Registration



35

Recording Member Info

```
<HTML XMLns="http://www.w3.org/1999/xHTML">
<body>
<H1>Hawthorn Bowling Club Member Registration</H1><br/>
<H3>To become a member … … click the Register button.</H3>
<form> Name: <input type="text" name="name"> <br/>
       Phone: <input type="text" name="phone"> <br/><br/>
       <input type="submit" value="Register" /> <br/>
</form>
</body>
<?php
  if(isset($_GET['name']) && isset($_GET['phone'])) {
    $bowlerName = $_GET['name']; $bowlerPhone = $_GET['phone'];
    $bowlerInfo = $bowlerName ., " . $bowlerPhone ."\n";
    $file = "../../data/bowlers.txt";
    if(file_put_contents($file, $bowlerInfo, FILE_APPEND) > 0)
      echo "<p>{$_GET['name']} has been registered as a member
         of the bowling club.</p>";
    else echo "<p>Registration error!</p>";
  }
?>
</HTML>
```

36

`addslashes()` 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>";
}
```

Note: 'mercury' PHP settings currently use *magic_quotes_gpc* which applies magic quotes, i.e., adds a backslash (\) to a quote in any user-submitted data so do not need addslashes()

37

`stripslashes()` Function

- To prevent the display of escaped characters, use the `stripslashes()` function

```
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>";
```

38

Writing Data Incrementally

- Use the **`fwrite()`** function to *incrementally write* data to a text file. **`fputs()`** is an alias for **`fwrite()`**

```
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

39

Locking Files

- Use the **`flock()`** function, to prevent multiple users from modifying a file simultaneously

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

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

40

Reading an Entire File

- The `file_get_contents()` function reads the contents of a file into a string, e.g.,

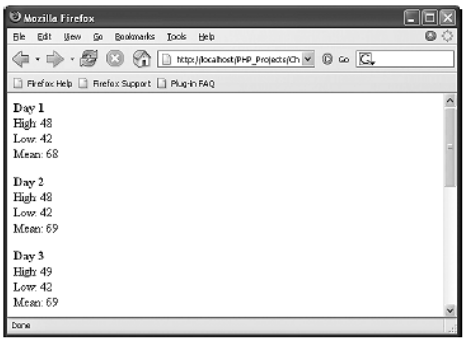
```
$sfWeather = file_get_contents("sfweather.txt");
```

- The `readfile()` function prints the contents of a file along with the file size to a Web browser
- The `file()` function reads the contents of a file into an *indexed array*. It automatically recognises whether the lines in a text file end in `\n`, `\r`, or `\r\n`
- The `fread()` function reads the contents of a file into a string up to a maximum number of bytes

41

Example of `file()` Function

```
$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);
$januaryTemps = file("sfjanaverages.txt");
for ($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 Data Incrementally

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

43

Reading Data Incrementally

- You must use `fopen()` and `fclose()` with the functions listed in the table in the previous slide.
- The commonly used `fgets()` function uses the file pointer to iterate through a text file
- Each time you call any of these functions, 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
- Often combined with the **`feof()`** function

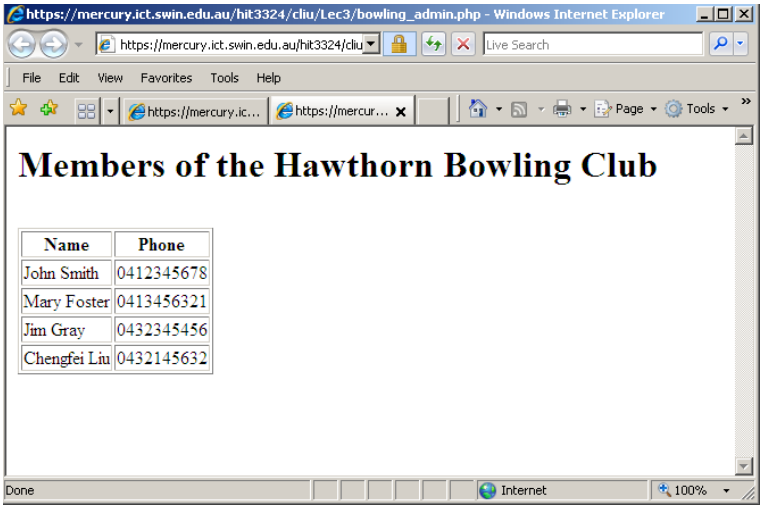
44

Example - Reading Data Incrementally

```
$handle = fopen("sfjanaverages.txt", "r");
while (! feof($handle) ) {
  $curLine = fgets ($handle);
   $curDay = explode(", ", $curLine);
      echo "<p><strong>Day " . ($i + 1)
               . "</strong><br/>";
      echo "High: {$curDay[0]}<br />";
      echo "Low: {$curDay[1]}<br />";
      echo "Mean: {$curDay[2]}</p>";
}
fclose ($handle);
```

45

Back to the Bowling Club



46

Listing all Members

```
<HTML XMLns="http://www.w3.org/1999/xHTML">
<body>
<H1>Members of the Hawthorn Bowling Club</H1>
<br/>
<?php
  $file = "../../data/bowlers.txt";
  if(!file_exists($file))
    echo "No registered member found!";
  else {
    $bowlers=file($file);
    echo "<table border='1'><th>Name</th><th>Phone</th>";
    for($i=0;$i<count($bowlers);$i++) {
      $curBowler = explode(",",$bowlers[$i]);
      echo "<tr><td>".$curBowler[0]."</td>";
      echo "<td>".$curBowler[1]."</td></tr>";
    }
    echo "</table>";
  }
?>
</body>
</HTML>
```

47

Adding and Removing Elements

- From the beginning of an array
  - The **`array_shift()`** function removes the first element from the beginning of an array
  - The **`array_unshift()`** function adds one or more elements to the beginning of an array. Pass the name of an array followed by comma-separated values for each element you want to add
- From the end of an array
  - The **`array_pop()`** function removes the last element from the end of an array
  - The **`array_push()`** function adds one or more elements to the end of an array.

```
$hospitalDepts = array("Anesthesia", "Molecular Biology",
                       "Neurology", "Pediatrics");
array_pop($hospitalDepts);  //removes Pediatrics from end
array_push($hospitalDepts, "Psychiatry", "Pulmonary Diseases");
// adds these to the end of array
```

48

Adding and Removing Elements

- The **`array_splice()`** function adds or removes array elements within an array, and renumbers the indexes

```
array_splice(array_name, starting element,
   elements_to_delete, values_to_insert);
```

- To add an element within an array, include a value of 0 as the 3rd parameter
- To add more than one element within an array, pass the `array()` construct as the 4th parameter, separate the new `array()` element values by commas
- Delete array elements by omitting the 4th parameter from the `array_splice()` function. *If the 3rd parameter is also omitted, all elements starting from the specified position are deleted*

49

Examples of `array_splice()`

```
$hospitalDepts = array(
      "Anesthesia",              // first element (0)
      "Molecular Biology",       // second element (1)
      "Neurology",               // third element (2)
      "Pediatrics");             // fourth element (3)

//  Add two new elements between "Neurology" and  "Pediatrics"
array_splice($hospitalDepts, 3, 0,
   array("Ophthalmology","Otolaryngology"));

//  Delete 2nd & 3rd elements ("Molecular Biology" and "Neurology")
array_splice($hospitalDepts, 1, 2);
```

50

Declaring and Initialising Associative Arrays

- With associative arrays, you specify an element's key by using the array operator (=>)
- The syntax for declaring and initialising an associative array:
  - `$array_name = array(key=>value, ...);`
    e.g.
    ```
    $ProvinceCapitals = array("Ontario"=>"Toronto",
    "Alberta"=>"Edmonton", ...);
    ```
  - `$array_name[key] = value;`
    e.g.
    ```
    $provinceCapitals["Ontario"] = "Toronto";
    $provinceCapitals["Alberta"] = "Edmonton";
    ...
    ```
- The sytax to refer to an element in an associate array
  e.g. `echo $provinceCapitals["Ontario"];`

51

Output of Associative Arrays

```
$territorialCapitals["Nunavut"] = "Iqaluit";
$territorialCapitals["Northwest Territories"] = "Yellowknife";
$territorialCapitals[] = "Whitehorse"; // next indexed element
print_r($territorialCapitals);
```

Output of array with associative and indexed elements
Mixed use of keys and indexes

52

Iterating Through an Array

- The internal array pointer refers to the currently selected element in an array

Array pointer iteration functions

Function	Description
`current(array)`	Returns the current array element
`each(array)`	Returns the key and value of the current array element and moves the internal array pointer to the next element
`end(array)`	Moves the internal array pointer to the last element
`key(array)`	Returns the key of the current array element
`next(array)`	Moves the internal array pointer to the next element
`prev(array)`	Moves the internal array pointer to the previous element
`reset(array)`	Resets the internal array pointer to the first element

53

Iterating Through an Array (continued)

```
$provinceCapitals = array(
  "NewFounderland and Labrador"=>"St. John's",
  "Prince Edward Island"=>"Charlottetown",
  "Nova Scotia"=>"Halifax",
  "New Brunswick"=>"Fredericton",
  "Quebec"=>"Quebec City",
  "Ontario"=>"Toronto",
  "Manitoba"=>"Winnipeg",
  "Saskatchewan"=>"Regina",
  "Alberta"=>"Edmonton",
  "British Columbia"=>"Victoria");
  foreach ($provinceCapitals as $capital) {
     echo "The capital of ",
      key($provinceCapitals), " is $capital<br />";
  }
```

Correct as follows ☺

```
   foreach ($provinceCapitals as $capital) {
      echo "The capital of ",
       key($provinceCapitals), " is $capital<br />";
      next($provinceCapitals);
   }
```

54

Determining if a Value or a Key Exists

- The **`in_array()`** function returns *true* if a given value exists in an array
- The **`array_search()`** function determines whether a given value exists in an array and returns the *index* or *key* of the first matching element if the value exists, or returns *false* if the value does not exist

```
if (in_array("Neurology", $hospitalDepts))
   echo "<p>The hospital has a Neurology department.</p>";
```

- The **`array_key_exists()`** function determines whether a given index or key exists
  - The first parameter represents the key to search for
  - The second parameter represents the name of the array in which to search

55

Example - Determining if a Key Exists

```
$gamePieces["Dancer"] = "Daryl";
$gamePieces["Fat Man"] = "Dennis";
$gamePieces["Assassin"] = "Jennifer";
if (array_key_exists("Fat Man", $gamePieces)) {
      echo "<p>{$gamePieces["Fat Man"]} is already
              'Fat Man'.</p>";
} else {
      $gamePieces["Fat Man"] = "Don";
      echo "<p>{$gamePieces["Fat Man"]} is now
              'Fat Man'.</p>";
}
```

56

Returning a Portion of an Array

- The **`array_slice()`** function returns a portion of an array and assigns it to another array

```
new_array = array_slice(array_name, starting element,
elements_to_return);
```

```
$topGolfers = array("Tiger Woods", "Vijay Singh", "Ernie
Els", "Phil Mickelson", "Retief Goosen", "Padraig
Harrington", "David Toms", "Sergio Garcia", "Adam Scott",
"Stewart Cink");
$TopFiveGolfers = array_slice($TopGolfers, 1, 3);
echo "<p>The three selected golfers are:</p><p>";
for ($i = 0; $i < count($topFiveGolfers); $i++) {
   echo "{$topFiveGolfers[$i]}<br />";
}
echo "</p>";
// output: Vijay Singh, Ernie Els, Phil Mickelson in three lines
```

57

Sorting Arrays

The most commonly used array sorting functions are:

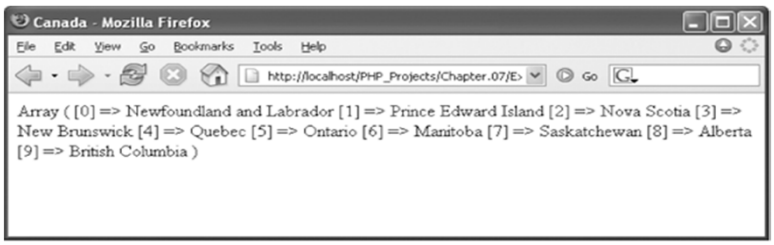
- **`sort()`** and **`rsort()`** for *indexed arrays*
  - `sort()` sorts an indexed array by value and renumbers the indexes
  - `rsort()` performs a reverse sort
- **`ksort()`** and **`krsort()`** for *associative arrays by key*

58

Combining Arrays

- To append one array to another, use the addition (+) or the compound assignment operator (+=). Only array elements with unique keys are appended. Duplicated indexes/keys are ignored

```
$provinces = array("Newfoundland and Labrador",
   "Prince Edward Island", "Nova Scotia", "New Brunswick",
   "Quebec", "Ontario", "Manitoba", "Saskatchewan", "Alberta",
   "British Columbia");
$territories = array("Nunavut", "Northwest Territories",
   "Yukon Territory");
$canada = $provinces + $territories;
print_r($canada); // territories ignored due to duplicate indexes
```



59

Combining Arrays (continued)

- + and += works best on associative arrays, especially if the arrays involved do not have any common keys.
- For example

```
$arr1 = array ("one"=>"apple", "two"=>"banana");
$arr2 = array ("three"=>"cherry", "four"=>"grapes");
$arr3 = $arr1 + $arr2;
print_r($arr3);
```

Output

```
Array ( [one] => apple [two] => banana [three] => cherry
 [four] => grapes )
```

60

Combining Arrays - Examples

- To merge two or more arrays use the **`array_merge()`** function. Duplicated associative keys overwrite, elements of numeric keys are appended.

```
new_array = array_merge($array1, $array2, $array3, ...);
```

Example, given

```
$arr1 = array ("one"=>"apple", "two"=>"banana");
$arr2 = array ("three"=>"cherry", "two"=>"grapes");
// Duplicate keys ignored
$arr3 = $arr1 + $arr2;
print_r($arr3);
```

Output: Array ([one] => apple [two] => banana [three] => cherry)

```
// Duplicate keys overwritten
$arr4 = array_merge($arr1, $arr2);
print_r($arr3);
```

Output: Array ([one] => apple [two] => grapes [three] => cherry)

61

Combining Arrays (continued)

- array_merge works best with arrays having numeric keys
- For example

```
$provinces = array("Newfoundland and Labrador",
   "Prince Edward Island", "Nova Scotia", "New Brunswick",
   "Quebec", "Ontario", "Manitoba", "Saskatchewan", "Alberta",
   "British Columbia");
$territories = array("Nunavut", "Northwest Territories",
   "Yukon Territory");
$canada = array_merge ($provinces, $territories);
print_r($canada);  //territories appended
```

Output:

Array ([0] => Newfoundland and Labrador [1] => Prince Edward Island [2] => Nova Scotia [3] => New Brunswick [4] => Quebec [5] => Ontario [6] => Manitoba [7] => Saskatchewan [8] => Alberta [9] => British Columbia [10] => Nunavut [11] => Northwest Territories [12] => Yukon Territory)

62

Comparing Arrays

- The **`array_diff()`** function returns an array of elements that exist in one array but not in any other arrays to which it is compared

```
new_array = array_diff($array1, $array2, $array3, ...);
```

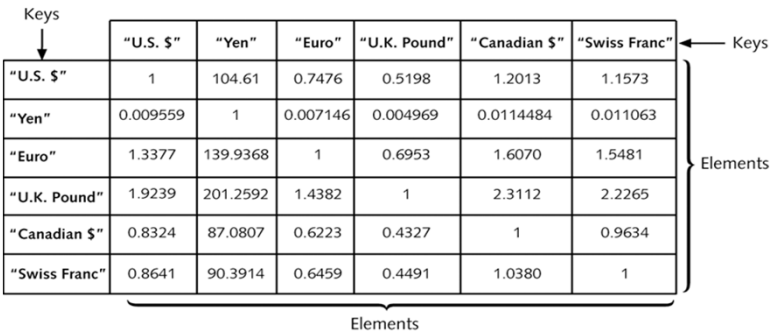
- The **`array_intersect()`** function returns an array of elements that exist in all of the arrays that are compared

```
new_array = array_intersect($array1, $array2, $array3,
...);
```

63

Multi-dimensional Indexed Arrays

- A multi-dimensional array consists of multiple indexes or keys
- A *two-dimensional* array has two sets of indexes or keys



	"U.S. $"	"Yen"	"Euro"	"U.K. Pound"	"Canadian $"	"Swiss Franc"
"U.S. $"	1	104.61	0.7476	0.5198	1.2013	1.1573
"Yen"	0.009559	1	0.007146	0.004969	0.0114484	0.011063
"Euro"	1.3377	139.9368	1	0.6953	1.6070	1.5481
"U.K. Pound"	1.9239	201.2592	1.4382	1	2.3112	2.2265
"Canadian $"	0.8324	87.0807	0.6223	0.4327	1	0.9634
"Swiss Franc"	0.8641	90.3914	0.6459	0.4491	1.0380	1

Elements and keys in the `$exchangeRates[row][col]` array

64

Creating Two-Dimensional Indexed Arrays

```
$usDollars = array(1, 104.61, 0.7476, 0.5198, 1.2013, 1.1573);
$yen = array(0.009559, 1, 0.007146, 0.004969, 0.011484, 0.011063);
$euro = array(1.3377, 139.9368, 1, 0.6953, 1.6070, 1.5481);
$ukPound = array(1.9239, 201.2592, 1.4382, 1, 2.3112, 2.2265);
$canadianDollar = array(0.8324, 87.0807, 0.6223, 0.4327, 1, 0.9634);
$swissFranc = array(0.8641, 90.3914, 0.6459, 0.4491, 1.0380, 1);
$exchangeRates = array($usDollars, $yen, $euro, $ukPound,
   $canadianDollar, $swissFranc);
```

```
$exchangeRates = array(
    array(1, 104.61, 0.7476, 0.5198, 1.2013, 1.1573),  // U.S. $
    array(0.009559, 1, 0.007146, 0.004969, 0.011484, 0.011063), // Yen
    array(1.3377, 139.9368, 1, 0.6953, 1.6070, 1.5481), // Euro
    array(1.9239, 201.2592, 1.4382, 1, 2.3112, 2.2265), // U.K. Pound
    array(0.8324, 87.0807, 0.6223, 0.4327, 1, 0.9634), // Canadian $
    array(0.8641, 90.3914, 0.6459, 0.4491, 1.0380, 1)  // Swiss Franc
);
```

65