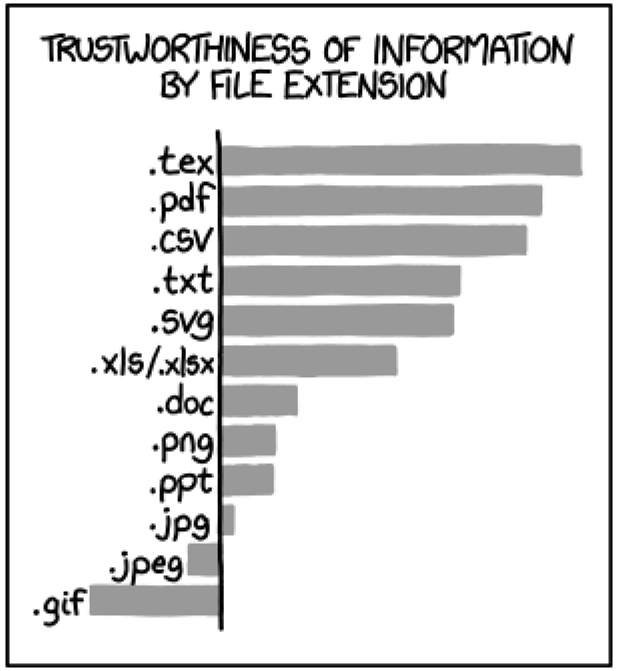
CSE 154

LECTURE 16: FILE I/O; FUNCTIONS



Query strings and parameters

```
URL?name=value&name=value...
```

```
http://www.google.com/search?q=Romney
http://example.com/student_login.php?username=obourn&id=1234567
```

- **query string**: a set of parameters passed from a browser to a web server
  - often passed by placing name/value pairs at the end of a URL
  - above, parameter username has value obourn, and sid has value 1234567
- PHP code on the server can examine and utilize the value of parameters
- a way for PHP code to produce different output based on values passed by the user

Query parameters: $_GET, $_POST

```
$user_name = $_GET["username"];
$id_number = (int) $_GET["id"];
$eats_meat = FALSE;
if (isset($_GET["meat"])) {
  $eats_meat = TRUE;
}                                                               PHP
```

- $_GET["parameter name"] or $_POST["parameter name"] returns a GET/POST parameter's value as a string
- parameters specified as http://....?name=value&name=value are GET parameters
- test whether a given parameter was passed with isset

PHP file I/O functions

function name(s)	category
file, **file_get_contents**, **file_put_contents**	reading/writing entire files
basename, file_exists, filesize, fileperms, filemtime, is_dir, is_readable, is_writable, disk_free_space	asking for information
copy, rename, unlink, chmod, chgrp, chown, mkdir, rmdir	manipulating files and directories
glob, **scandir**	reading directories

Reading/writing files

contents of foo.txt	**file("foo.txt")**	**file_get_contents("foo.txt")**
Hello how r u? I'm fine	array("Hello\n", # 0 "how r u?\n", # 1 "\n", # 2 "I'm fine\n" # 3)	"Hello\n how r u?\n # a single \n # string I'm fine\n"

- file function returns lines of a file as an array (\n at end of each)
- `file_get_contents` returns entire contents of a file as a single string
- `file_put_contents` writes a string into a file

The file function

```
# display lines of file as a bulleted list
$lines = file("todolist.txt");
foreach ($lines as $line) {     # for ($i = 0; $i < count($lines); $i++)
  print $line;
}                                                                         PHP
```

- file returns the lines of a file as an array of strings
- each ends with \n ; to strip it, use an optional second parameter:

```
$lines = file("todolist.txt", FILE_IGNORE_NEW_LINES);                 PHP
```

- common idiom: foreach or for loop over lines of file

Splitting/joining strings

```
$array = explode(delimiter, string);
$string = implode(delimiter, array);                                PHP
```

```
$s  = "CSE 190 M";
$a  = explode(" ", $s);      # ("CSE", "190", "M")
$s2 = implode("...", $a);    # "CSE...190...M"                      PHP
```

- explode and implode convert between strings and arrays
- for more complex string splitting, you can use regular expressions (later)

Example with explode

```
Martin D Stepp
Jessica K Miller
Victoria R Kirst
```

contents of input file names.txt

```
foreach (file("names.txt") as $name) {
  $tokens = explode(" ", $name);
  ?>
  <p> author: <?= $tokens[2] ?>, <?= $tokens[0] ?> </p>
  <?php
}
```

```
author: Stepp, Marty

author: Miller, Jessica

author: Kirst, Victoria
```

output

Unpacking an array: list

```
list($var1, ..., $varN) = array;                                PHP
```

```
Allison Obourn
(206) 685 2181
570-86-7326                          contents of input file personal.txt
```

```
list($name, $phone, $ssn) = file("personal.txt");
...
list($area_code, $prefix, $suffix) = explode(" ", $phone);      PHP
```

- the odd list function "unpacks" an array into a set of variables you declare
- when you know a file or line's exact length/format, use file and list to unpack it

Reading directories

function	description
glob	returns an array of all file names that match a given pattern (returns a file path and name, such as "foo/bar/myfile.txt")
scandir	returns an array of all file names in a given directory (returns just the file names, such as "myfile.txt")

- glob can accept a general path with the * wildcard character (more powerful)

glob example

```
# reverse all poems in the poetry directory
$poems = glob("poetry/poem*.dat");
foreach ($poems as $poemfile) {
  $text = file_get_contents($poemfile);
  file_put_contents($poemfile, strrev($text));
  print "I just reversed " . basename($poemfile) . "\n";
}                                                          PHP
```

- glob can match a "wildcard" path with the * character
  - `glob("foo/bar/*.doc")` returns all .doc files in the foo/bar subdirectory
  - `glob("food*")` returns all files whose names begin with "food"
- the `basename` function strips any leading directory from a file path
  - `basename("foo/bar/baz.txt")` returns "`baz.txt`"

scandir example

```
<ul>
  <?php foreach (scandir("taxes/old") as $filename) { ?>
    <li>I found a file: <?= $filename ?></li>
  <?php } ?>
</ul>
```
PHP

```
• .
• ..
• 2007_w2.pdf
• 2006_1099.doc
```
output

- `scandir` includes current directory (".") and parent ("..") in the array
- don't need `basename` with `scandir`; returns file names only without directory

Reading/writing an entire file

```php
# reverse a file
$text = file_get_contents("poem.txt");
$text = strrev($text);
file_put_contents("poem.txt", $text);                    PHP
```

- file_get_contents returns entire contents of a file as a string
  - if the file doesn't exist, you will get a warning and an empty return string
- file_put_contents writes a string into a file, replacing its old contents
  - if the file doesn't exist, it will be created

Appending to a file

```
# add a line to a file
$new_text = "P.S. ILY, GTG TTYL!~";
file_put_contents("poem.txt", $new_text, FILE_APPEND);          PHP
```

old contents	new contents
Roses are red, Violets are blue. All my base, Are belong to you.	Roses are red, Violets are blue. All my base, Are belong to you. P.S. ILY, GTG TTYL!~

- file_put_contents can be called with an optional third parameter to append (add to the end) rather than overwrite