COMP 2021

Unix and Script Programming

File Handling in PHP

Data Storage: Flat File vs. Database

- Both provide long-term storage of data (information)
- Pros. of Flat File
  - Available and versatile: create and save data in any operating system's file system. Ready to be read by a variety of other programs.
  - Easy to use: no extra software needed.
  - Smaller: use less disk space

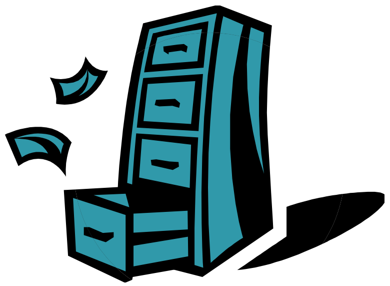
- Pros. Of Database
  - Security: provide a security layer of its own, in addition to the security provided by the operating system. A database protects the data from outside intrusion better than a flat file.
  - Accessibility of data: allow complex data structure, specifying types and relationships among the data.
  - Ability to handle multiple users

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

Files and PHP

- PHP provides file handling in
  - File and directory manipulations
  - File upload
    - From forms



PHP File Input/Output

File I/O Tasks

- Opening a file / Creating a new file
- Reading data from a file
  - Entire file
  - line-by-line
  - character by character
- Manipulating file contents
- Getting file information
- Checking end-of-file (EOF)
- Writing data to a file
- Closing a file

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

Open/Close a File

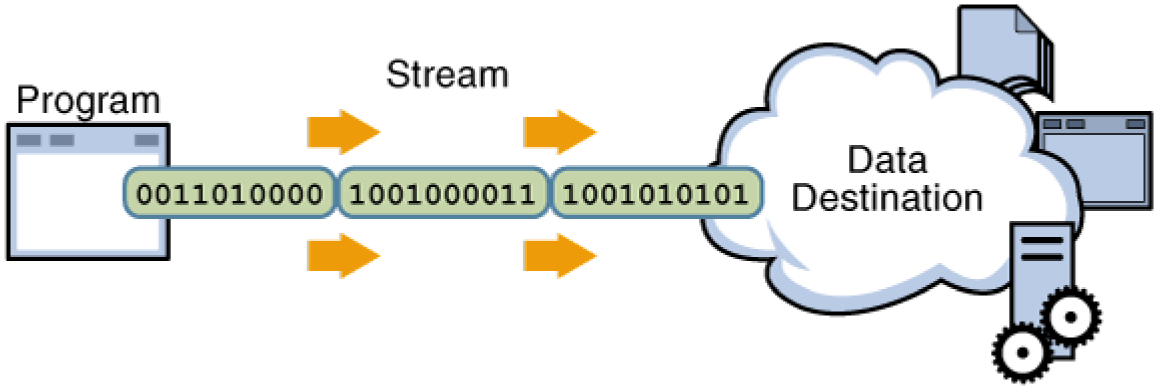
- Given the file path, a file is opened with fopen() as a "stream", and PHP returns a "handle" to the file that can be used to reference the open file in other functions.
- Each file is opened in a particular mode.
- FALSE is returned if can not open the file
- A file stream is closed with fclose() or when your script ends.

```php
<?php
    $filehandle = fopen("c:\\folder\\resource.txt", "r");
    fclose($filehandle);
?>
```

PHP

File Stream

- 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
  - E.g. binary iutput stream



File Handle

- A **file 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

- More technical definition
  - A temporary reference (typically a number) assigned by the operating system to a **file** that an application has asked it to open. The **handle** is used throughout the session to access the **file**.

File Open Modes

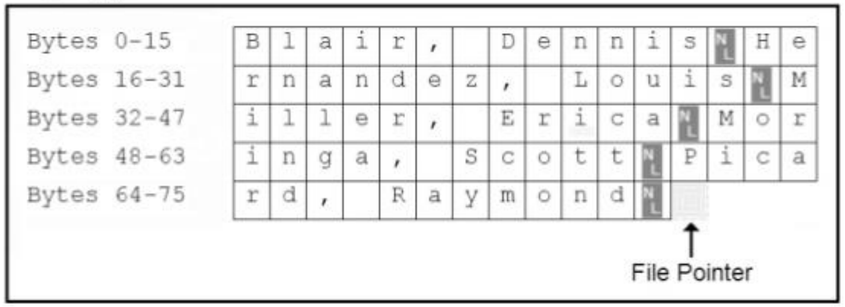
‘r’	Open for reading only. Start at beginning of file.
‘r+’	Open for reading and writing. Start at beginning of file.
‘w’	Write only. Opens and clears the contents of file; or creates a new file if it doesn't exist
‘w+’	Read/Write. Opens and clears the contents of file; or creates a new file if it doesn't exist
‘a’	Open writing, but start at END of current content.
‘a+’	Open for reading and writing, start at END and create file if necessary.
‘x’	Write only. Creates a new file. Returns FALSE and an error if file already exists
‘x+’	Read/Write. Creates a new file. Returns FALSE and an error if file already exists

File Pointer

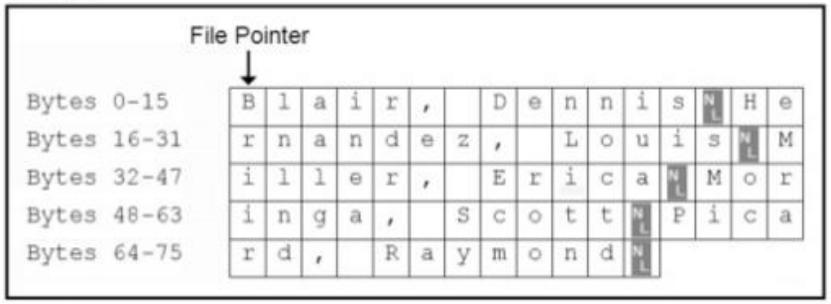
- A **file pointer** is a special type of variable that refers to the currently selected line or character in a file

- E.g. initial location of the file pointer when open file with “a+” and “r+” mode

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



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



Read the Entire File

- Different ways to read the entire file
  - `file()` returns lines of a file as an **array**
    - Try `<?php print_r(file(“hello.txt"));?>`
  - `file_get_contents()` returns entire contents of a file as a string
  - `readfile()` display the contents and file size on the browser

contents of hello.txt	file(“hello.txt")	file_get_contents(“hello.txt")

`Hello`
`how are`
`you?`
 	
`I'm fine`
	`array(`
	`"Hello\n",    #0`
	`"how are\n", #1`
	`"you?\n",     #2`
	`"\n",        #3`
	`"I'm fine\n" #4 )`
		`"Hello\n`
		`how are\n`
		`you?\n`
		`\n`
		`I'm fine\n"`

Example: Read/Write an Entire File

```php
<?php
#filereverse.php
    $text = file_get_contents("palindrome.txt");
    $text = strrev($text);
    echo "$text";
    file_put_contents("palindrome.txt", $text);
?>
```

PHP

- `file_get_contents()` returns entire contents of a file as a string
- `file_put_contents()` writes a string into a file, replacing any prior contents

Example: Open a File and Read

```php
<?php
#  fileopen.php

     $filename = "CapitalCity.txt";
     $file = fopen( $filename, "r" ) or exit(“Unable to
open file”);

     $filesize = filesize($filename);
     # read whole file, you can specify other size
     $filetext = fread($file, $filesize);
     fclose($file);

     echo ("$filename");
     echo( "File size : $filesize bytes" );
     echo ( "$filetext" );
?>
```

PHP

File Read Functions

`fread(file,length)`	Reads from an open file. Stops at the end of the file or when reaches the specified length, whichever comes first. Returns the read string, or FALSE on failure.
`feof(file)`	Checks if the "end-of-file" (EOF) has been reached. Returns TRUE if an error occurs, or EOF has been reached. Otherwise it returns FALSE.
`fgetc(file)`	Returns a single character from an open file. Slow and should not be used on large files. Use fgets() to grab a line, then fgetc() on that line instead for large file.
`fgets(file, length)`	Returns a line from an open file.
`fgetss(file,length, tags)`	Returns a line, with HTML and PHP tags removed, from an open file.
`filesize(filename)`	Returns the file size in bytes.

Example: Read a File Line-by-Line

```
<?php
#  filereadline.php
    $filename = "CapitalCity.txt";
    $file = fopen( $filename, "r" ) or exit("Unable to
open file");
    $linenum = 0;

    while (!feof($file))
    {
        $line = fgets($file);
        $linenum++;
        echo "$linenum : $line \n";
    }
     fclose($file);
?>
```

PHP

Write to a File

- Open a file in write mode and write the contents
  - `fwrite(file, data)`
  - `fputc(file, char)` - writes character by character
  - `fputs(file, line)` - writes line by line

```
<?php
#filewrite.php
    $filename = fopen("marvel.txt", "w") or die("Unable to
open file!");
    $txt = "Captin American\n";
    fwrite($filename, $txt);
    $txt = "Thor\n";
    fwrite($filename, $txt);
    fclose($filename);
?>
```

PHP

Example: Append to a File

```
<?php
#fileappend.php
    # method 1: a+ enables read and append
    $file = fopen("palindrome.txt", "a+") or die("Unable
to open file!");
    $new_text = "Madam in Eden, I'm Adam\n";
    fwrite($file, "\n". $new_text);
    # method 2:
    $new_text = "A Santa lived as a devil at NASA\n";
    file_put_contents("palindrome.txt", $new_text,
FILE_APPEND);
    # show updated file contents
    fread("palindrome.txt");
?>
```

PHP

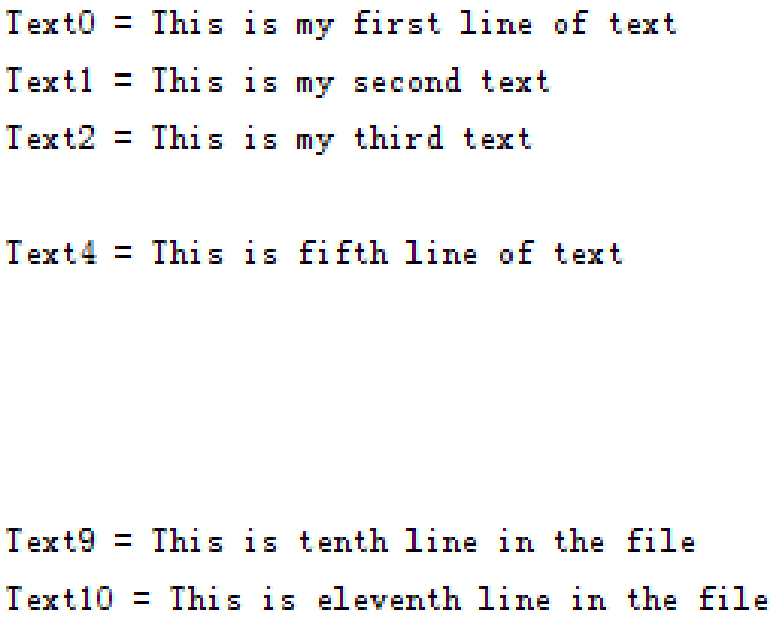
Example: Copy between Two Files

```php
<?php
#filecopy.php
$file = "CaptialCity.txt";
$newfile = 'CapticalCity.txt.bak';

copy($file, $newfile) or die("failed to copy $file...\n");
?>
```

PHP

Example: Insert into File 1

- Format of fileinsert1.txt



```
Text0 = This is my first line of text
Text1 = This is my second text
Text2 = This is my third text

Text4 = This is fifth line of text




Text9 = This is tenth line in the file
Text10 = This is eleventh line in the file
```

- Task: update the 8th line

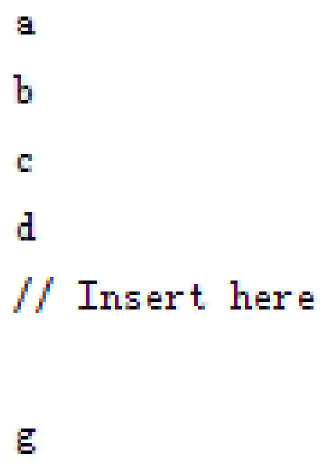
Example: Insert into File 1(cont.)

```
<?php
#fileinsert1.php
#Update the 8th line of a file. File lines start from 0.

    $filename = "fileinsert1.txt";
    $new_line = "Text7 = This is eighth line in the file";
    $element_number_to_replace = 7;
    $line_array = file($filename);

    $line_number_to_insert = 7;
    $line_array[$line_number_to_insert] = $new_line;
    $all_content = implode("\n", $line_array);
    file_put_contents($filename,$all_content);
?>
                                                        PHP
```

- **Implode(): join array elements with a glue string**
- `string implode ( string $glue , array $pieces )`

Example: Insert into File 2

- Format of fileinsert2.txt



```
a
b
c
d
// Insert here


g
```

- Task: Insert at the specified location

Example: Insert into File 2 (cont.)

```php
<?php
#fileinsert2.php
#insert to a location specified by tag

    $filename = "fileinsert2.txt";
    $file = fopen($filename, "r+") or die("Unable to open
file");
    $filecontents = fread($file, filesize($filename));
    fseek($file, strpos($filecontents, "// Insert here"));
    fwrite($file, "Insert successfully\n");
    fclose($file);
?>
                                                          PHP
```

- fseek(): seeks on a file pointer
- `int fseek(resource $filehandle , int $offset)`

Manipulate File Content

- reads file lines and unpacks the lines into variables
  - Either use file() to get the lines of a file as an array of strings
  - Or use while loop to read line-by-line
  - Each string ends with \n (can strip the \n off)

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

  - Split string contents into several variables

Example: File Manipulation

```php
<?php
#filenaminpulate.php
    $srcfile = fopen("readfrom.txt", "r");
    $destfile = fopen("writeto.txt", 'a');

    while(!feof($srcfile)) {
        $line = fgets($srcfile);
        $lineAry = explode("=", $line);
        if (strlen($lineAry[0]) >= 3)
            fwrite($destfile, $line);
    }

    fclose($srcfile);
    fclose($destfile);

?>
```

PHP

Split/Join String

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

```
$str = "COMP 2021 L1";
$arr = explode(" ", $s); # ("COMP", “2021", “L1")
$str2 = implode("...", $a); # “COMP...2021...L1"


                                                    PHP
```

- `explode` and `implode` convert between strings and arrays

Unpacking an Array: `list`

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

```php
$values = array("cindy", "18", "f");
list($username, $age, $gender) = $values;
```
PHP

- the list() function accepts a comma-separated list of variable names as parameters
- use this to quickly "unpack" an array's contents into several variables

Example: explode and list

```
Harry Potter, J.K. Rowling
The Lord of the Rings, J.R.R. Tolkien
Dune, Frank Herbert
```

contents of input file explode.txt

```
<?php
#explode.php
    foreach (file("explode.txt") as $book) {
        list($title, $author) = explode(",", $book);
        echo "$Book: $title, Author: $author\n";
}
?>
```

PHP

How many arrays are used in this example?

Reading Directories

Function	Description
scandir	returns an array of all file names in a given directory (returns just the file names, such as "`myfile.txt`")
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`")

Example: `scandir`

```
<ul>
<?php
      $folder = "comp2021/notes";
      foreach (scandir($folder) as $filename) {
?>
      <li> <?= $filename ?> </li>
<?php
}
?>
</ul>
```

Embedded PHP

```
• .
• ..
• comp2021Lec1.pdf
• comp2021hw1.doc
```

output

Example: glob

```php
<?php
# glob.php
$poems = glob("poetry/poem*.txt");
foreach($poems as $poemfile) {
    $text = file_get_contents($poemfile);
    file_put_contents($poemfile, strrev($text));
    print "Reversed: ". basename($poemfile) . "\n";
}
?>
```
PHP

- glob can match a "wildcard" path with the * character
- the basename() function strips any leading directory from a file path

Include File

Include files: `include`

```php
<?php
#function.php
    function doit(){ echo "did it"; }
?>
```
PHP

```php
<?php
#main.php
    include("function.php");
    doit();
?>
```
PHP

- Inserts the entire contents of the given file into the PHP script's output page
- Encourages modularity
- Useful for defining reused functions needed by multiple pages

Example: include

```
<a href="http://www.example.com/index.php">Home</a>
- <a href="http://www.example.com/about.php">About
Us</a> - <a
href="http://www.example.com/links.php">Links</a> -
<a href="http://www.example.com/contact.php">Contact
Us</a> <br />
```

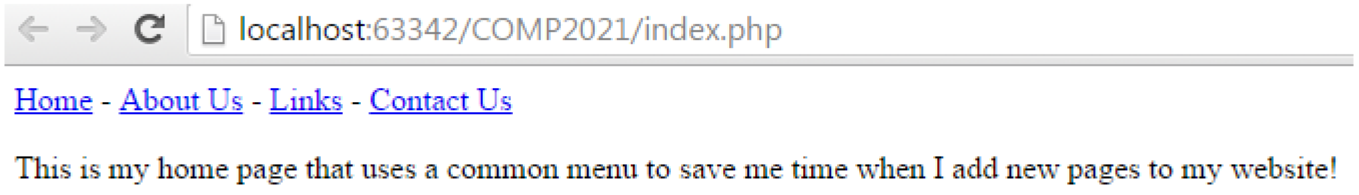
menu.php

```
<html>
<body>
<?php include("menu.php"); ?>
<p>This is my home page that uses a common menu to
save me time when I add new pages to my website!</p>
</body>
</html>
```

index.php

Example: include (cont.)

- Visitor's view of index.php

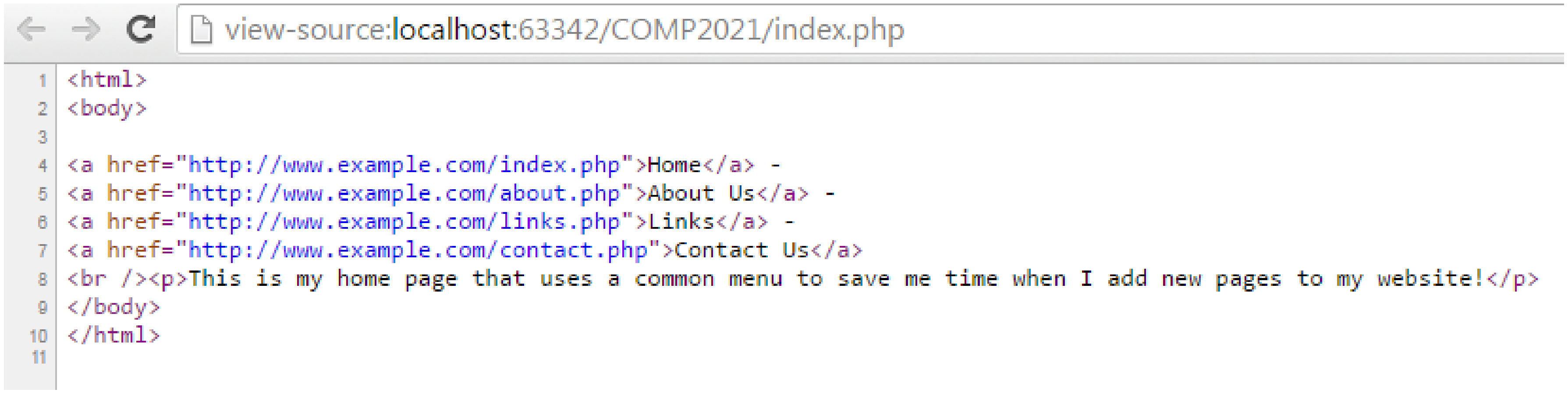


localhost:63342/COMP2021/index.php

Home - About Us - Links - Contact Us

This is my home page that uses a common menu to save me time when I add new pages to my website!

- View Source of index.php to a visitor



view-source:localhost:63342/COMP2021/index.php

```
<html>
<body>

<a href="http://www.example.com/index.php">Home</a> -
<a href="http://www.example.com/about.php">About Us</a> -
<a href="http://www.example.com/links.php">Links</a> -
<a href="http://www.example.com/contact.php">Contact Us</a>
<br /><p>This is my home page that uses a common menu to save me time when I add new pages to my website!</p>
</body>
</html>

```

Include Error

```php
<?php
       include("noFileExistsHere.php");
       echo "Hello World!";
?>
```

PHP

```
Warning: include(noFileExistsHere.php): failed to open
stream: No such file or directory in
C:\Users\cindy\PhpstormProjects\COMP2021\require.php on
line 2

Warning: include(): Failed opening 'noFileExistsHere.php'
for inclusion (include_path='C:\xampp\php\PEAR') in
C:\Users\cindy\PhpstormProjects\COMP2021\require.php on
line 2
Hello World!
```

Error Message

a warning does not prevent PHP script from running

Include() vs. Require()

```php
<?php
       require("noFileExistsHere.php");
       echo "Hello World!";
?>
```

PHP

```
Warning: include(noFileExistsHere.php): failed to open
stream: No such file or directory in
C:\Users\cindy\PhpstormProjects\COMP2021\require.php on
line 2

Fatal Error: include(): Failed opening
'noFileExistsHere.php' for inclusion
(include_path='C:\xampp\php\PEAR') in
C:\Users\cindy\PhpstormProjects\COMP2021\require.php on
line 2
```

Error Message

echo statement is not executed

Upload File

File Upload UI: Form

```
<html>
<body>
<h3> FILE UPLOADING </h3>
<hr>
<form action="fileupload.php" method="post"
enctype="multipart/form-data">
    <label for="file">Filename:</label>
    <input type="file" name="file" id="file" />
    <br />
    <input type="submit" name="submit" value="upload" />
</form>
</body>
</html>
```

HTML

- Add a file upload to form as an input tag with type of file
- Must also set the `enctype` attribute of the form

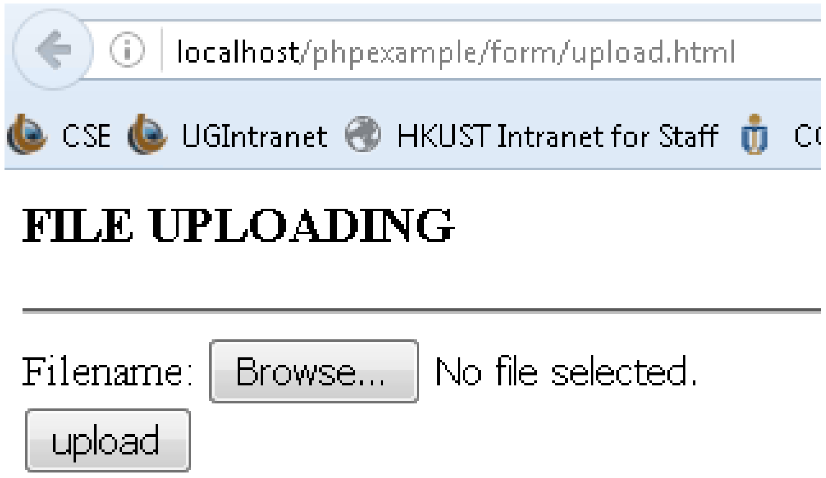
Processing an Uploaded File in PHP

- Uploaded files are placed into global array `$_FILES`, not `$_REQUEST`
- each element of `$_FILES` is itself an associative array, containing:
  - `name:` the local filename that the user uploaded
  - `type:` the MIME type of data that was uploaded, such as image/jpeg
  - `size :` file's size in bytes
  - `tmp_name :` a filename where PHP has temporarily saved the uploaded file
    - to permanently store the file, move it from this location into some other file

Processing an Uploaded File in PHP (cont.)

```
<?php
#fileupload.php
if ($_FILES["file"]["error"] > 0)
{
    echo "Error: " . $_FILES["file"]["error"] . "<br />";
}
else
{
    echo "File uploaded! <br />";
    echo "Upload: " . $_FILES["file"]["name"] . "<br />";
    echo "Type: " . $_FILES["file"]["type"] . "<br />";
    echo "Size: " . ($_FILES["file"]["size"] / 1024) . " 
Kb<br />";
    echo "Stored in: " . $_FILES["file"]["tmp_name"];
   move_uploaded_file($_FILES["file"]["tmp_name"], 
"userupload.txt");
}
?>
```

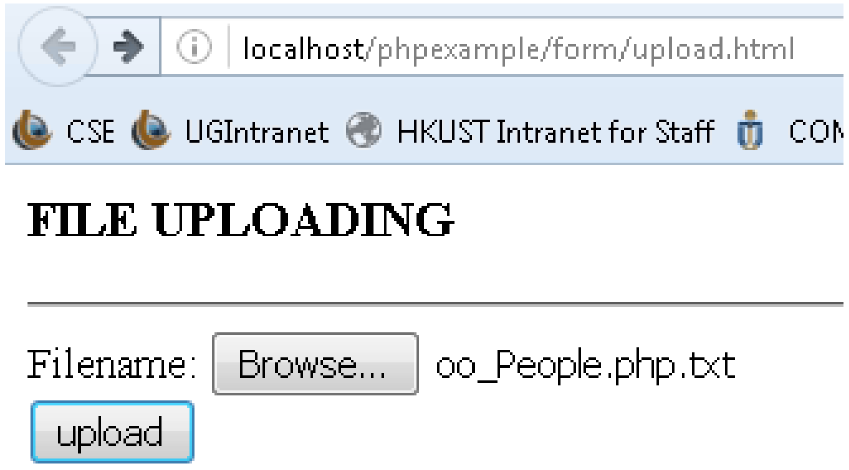
moves from a temporary file location to a more permanent file



FILE UPLOADING

Filename: Browse... No file selected.

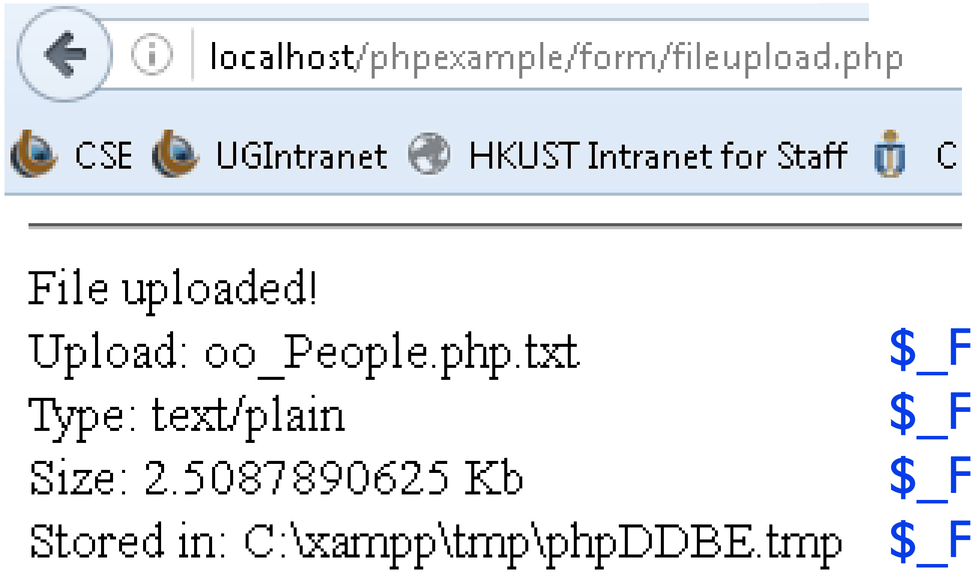
upload



FILE UPLOADING

Filename: Browse... oo_People.php.txt

upload



```
File uploaded!
Upload: oo_People.php.txt
Type: text/plain
Size: 2.5087890625 Kb
Stored in: C:\xampp\tmp\phpDDBE.tmp
```

$_FILES["file"]["name"]
$_FILES["file"]["type"]
$_FILES["file"]["size"]
$_FILES["file"]["tmp_name"]