CSIS 3280: Lecture 4

Agenda

- Log errors
- Exception Handling
- File Operations in PHP

Logging the Errors

- Errors are common to happen when you code
  - Missing a (or } or ;
- Errors during execution can be displayed in the
  - Command line
  - Browser window
  - Log file
- You don’t want error to be displayed in the production environment. There are also cases where the app cannot even print out any error message, e.g., when you design an API
  - Error log file will become really useful. Combine the PHP log file with several UNIX commands to manage text file like ‘tail’, ‘cat’, ‘cut’, etc, you will be able to spot the error and fix it
- There are several directives that handle the error logging in php.ini, e.g., error_reporting, display_error, log_errors, error_log

Error Logging and Custom Log message

- WAMP server is considered to be a development environment. So, it will log all errors and put it in c:/wamp64/logs/php_error.log
- To log custom message, you can use the error_log() function

```
error_log("New user registered");
```

  The message and associated timestamp will be log in the destination file

```
[24-Apr-2014 12:15:07] New user registered
```

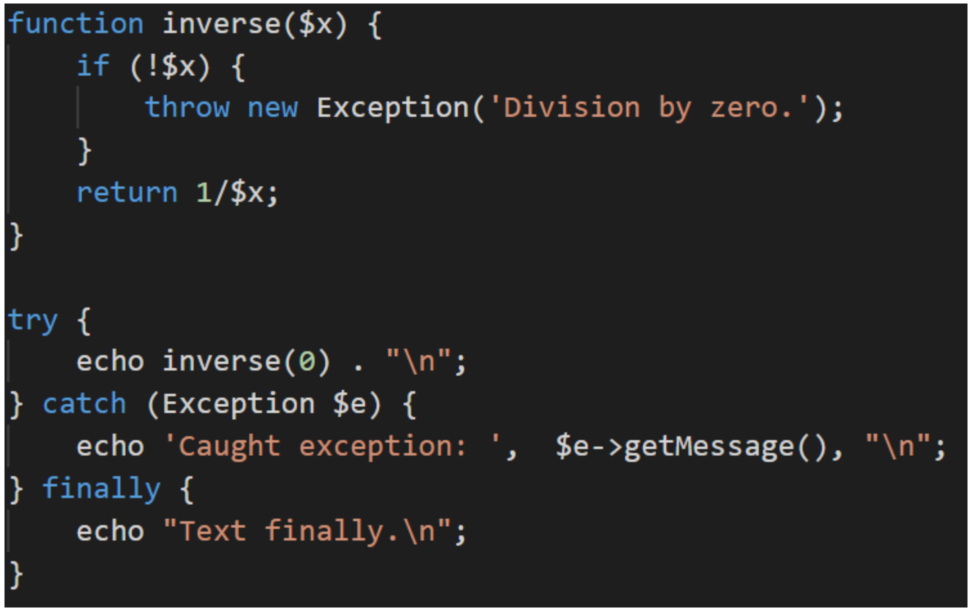
- You can also overwrite the action, e.g., log the error, send email, or append to a file, to a specific file

```
error_log("New user registered", 3, "/var/log/users.log");
```

I may ask you to submit the error log files of your project apps. Please look at the "error handling and logging" part of php.ini file and be familiar with the error log file, i.e., c:/wamp64/logs/php_error.log. You may be asked to set the log file with a different filename

Exception Handling

- Exception handling allow the code to react gracefully instead of crashing
- Exception handling is used to react to error that caused by different type of errors, for example: invalid calculation, file IO, database connection, etc.
- Similar to other programming language, php has try-catch-finally block. The following is an example of exception handling in file operation



```
function inverse($x) {
    if (!$x) {
        throw new Exception('Division by zero.');
    }
    return 1/$x;
}

try {
    echo inverse(0) . "\n";
} catch (Exception $e) {
    echo 'Caught exception: ',  $e->getMessage(), "\n";
} finally {
    echo "Text finally.\n";
}
```

More information about exception can be obtained in the following
https://www.php.net/manual/en/class.exception.php

File Handling

File operation

- Web applications are rarely 100% self-contained; that is, most rely on some sort of external data source (file or database) to do anything interesting
- Your PHP applications are running inside a server. That server is running on behalf of a user.
- Your file access and filesystem permissions are what the user for your webserver has.
- Better that the web server does not have administrative privileges on the machine
- This is the reason why web servers are run as a user
- If you are working with data, best to have it not accessible to the web

Opening and Closing a file

- Opening a file: fopen()
  - filehandle = fopen (string $filename , string $mode [, bool $use_include_path = false [,resource $context]]) –
  - returns false if file unable to be opened
  - Example: $fh =fopen("../myfile", 'a'); – fopen
- Relative Path: files stored
  - If it is console, relative to the current folder
  - If it is a web app, it is relative to document root
  - Your file folder ≠ web folder
- Document root: $_SERVER['DOCUMENT_ROOT']
  - C:\wamp64\www in wampserver
- Closing a file: fclose()
- **ALWAYS remember to CLOSE YOUR FILE**

File opening modes

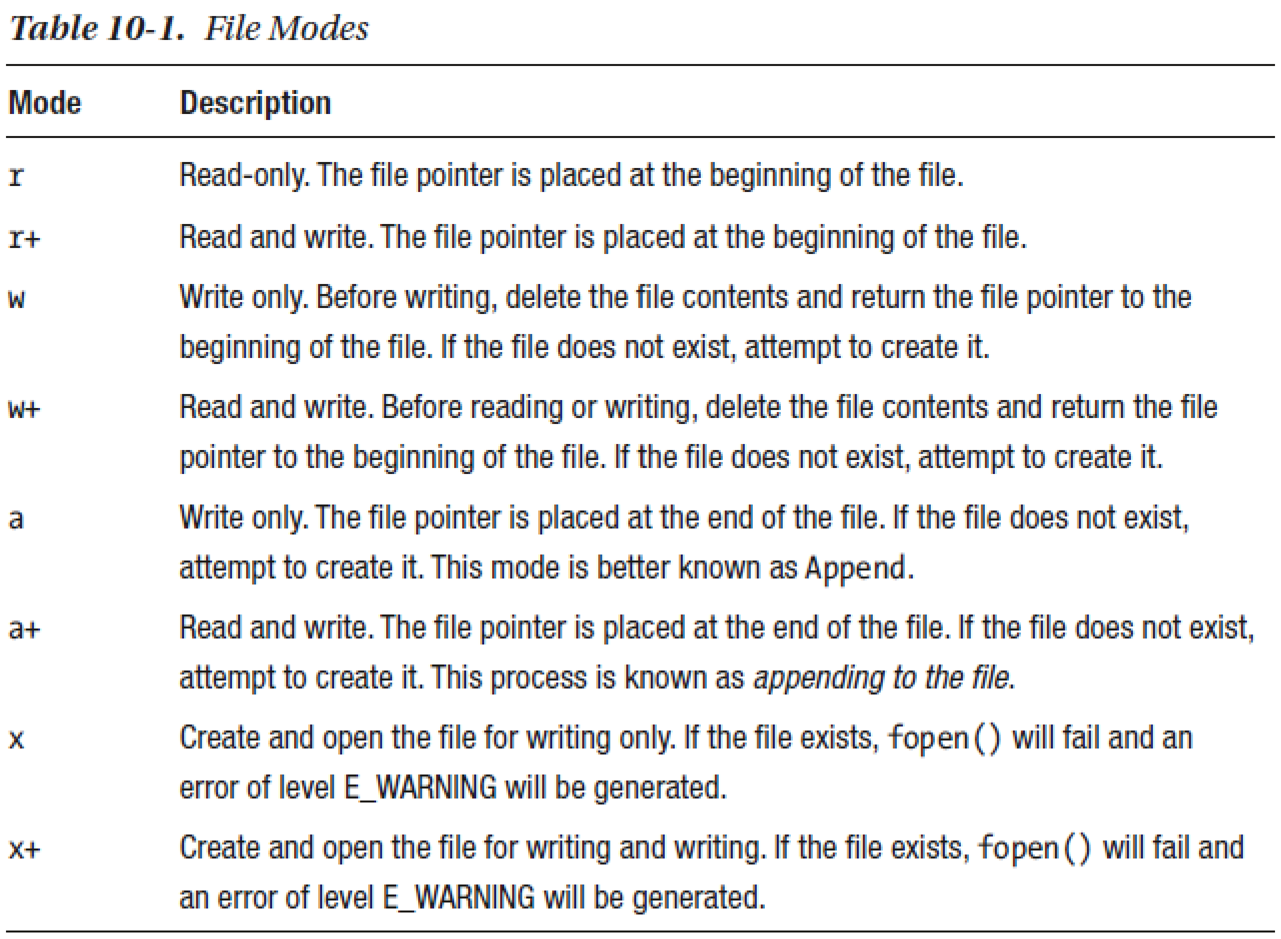


Table 10-1. File Modes

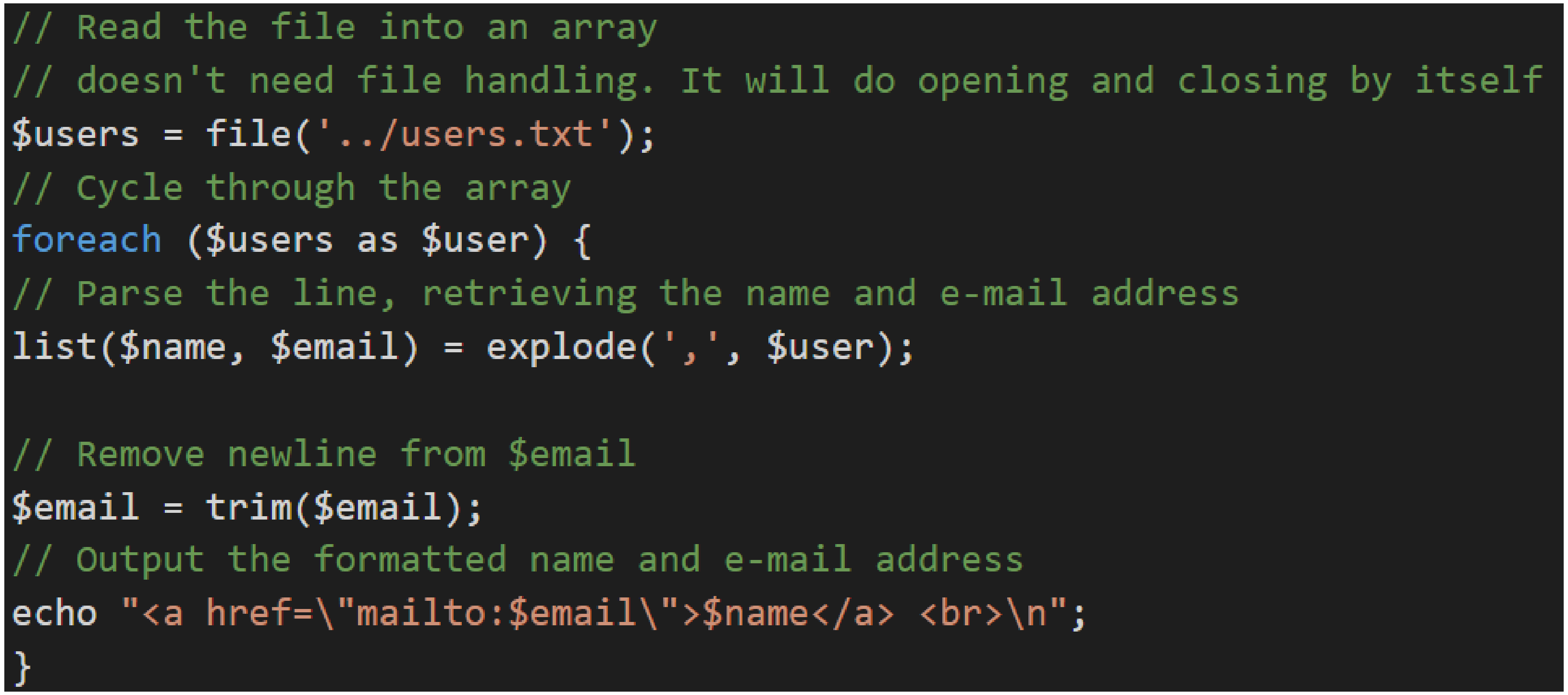
Mode	Description
`r`	Read-only. The file pointer is placed at the beginning of the file.
`r+`	Read and write. The file pointer is placed at the beginning of the file.
`w`	Write only. Before writing, delete the file contents and return the file pointer to the beginning of the file. If the file does not exist, attempt to create it.
`w+`	Read and write. Before reading or writing, delete the file contents and return the file pointer to the beginning of the file. If the file does not exist, attempt to create it.
`a`	Write only. The file pointer is placed at the end of the file. If the file does not exist, attempt to create it. This mode is better known as Append.
`a+`	Read and write. The file pointer is placed at the end of the file. If the file does not exist, attempt to create it. This process is known as *appending to the file*.
`x`	Create and open the file for writing only. If the file exists, `fopen()` will fail and an error of level E_WARNING will be generated.
`x+`	Create and open the file for writing and writing. If the file exists, `fopen()` will fail and an error of level E_WARNING will be generated.

Reading a file

- Use file handle to read contents
- First check if file exists
- Then open file for reading
- Read contents until end of file pointer is reached
  - Use functions for reading
  - fgets() – read a certain number of chars or EOL
  - fgetc() – read one char at a time
  - fread() – useful for binary file, ignores new line
  - fscanf() – read with a certain format, e.g., %d, %s

Functions that don’t need a handle

- There are several other functions to read the content of the file that will handle the opening and closing the file
  - file(), file_get_contents(), fgetcsv() and readfile()
  - Be careful in using these function especially if you don’t know the file size since all content will be dump into a variable → memory



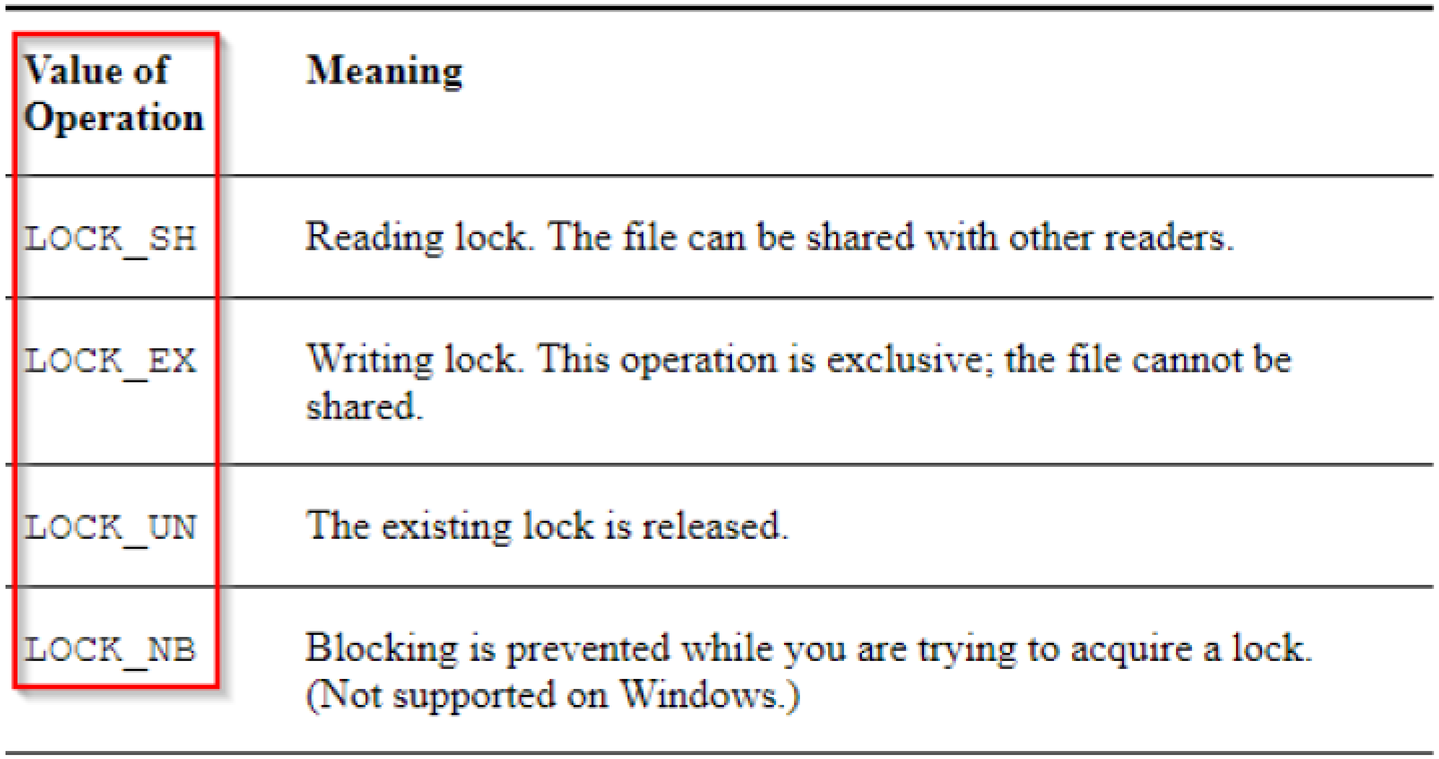
```php
// Read the file into an array
// doesn't need file handling. It will do opening and closing by itself
$users = file('../users.txt');
// Cycle through the array
foreach ($users as $user) {
// Parse the line, retrieving the name and e-mail address
list($name, $email) = explode(',', $user);

// Remove newline from $email
$email = trim($email);
// Output the formatted name and e-mail address
echo "<a href=\"mailto:$email\">$name</a> <br>\n";
}
```

Writing a file

- fwrite(filehandle, string, [int length])
  - Writes string to filehandle, and writes specified length if given, and returns how many bytes were written or FALSE
  - For example: fwrite($fp, $outputstring, strlen($outputstring));
  - strlen – gives number of bytes of the string (length in this case)
- file_put_contents(file name, string)
  - Does not require a filehandle, written directly to the file
- fputs(filehandle, string)
  - Writes a line of string to the file – Alias of fwrite but does not get length to write
  - Cannot be used with binary files
- fseek(resource *handle*, int *offset* [, int *whence*])
  - moves the pointer to the location specified by a provided offset value
  - whence can be SEEK_CUR, SEEK_END, SEEK_SET (= omit whence)
  - ftell() and rewind are other function related to file pointer

Locking files

- Lock files with flock($handle, CONSTANT);
- Locking is good to avoid conflicts

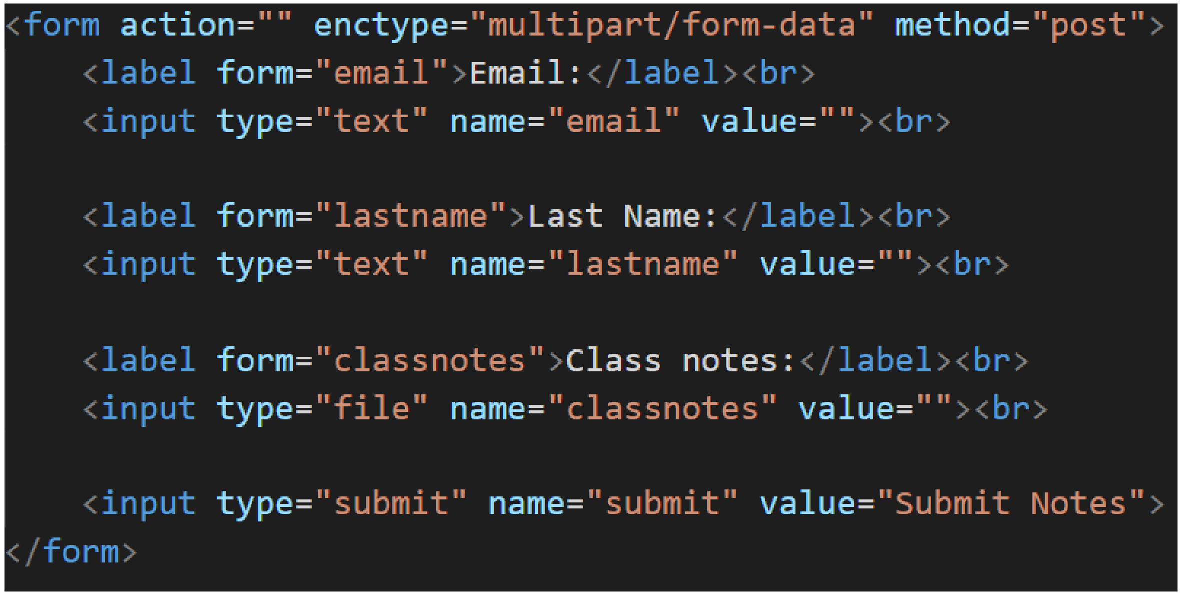


Value of Operation	Meaning
LOCK_SH	Reading lock. The file can be shared with other readers.
LOCK_EX	Writing lock. This operation is exclusive; the file cannot be shared.
LOCK_UN	The existing lock is released.
LOCK_NB	Blocking is prevented while you are trying to acquire a lock. (Not supported on Windows.)

File Upload

Uploading a File

- It is possible to transfer of any kind of file via HTTP, including images, video, docx, pdfs, executables, and other file types
- PHP's $_FILES superglobal array, used to handle file-upload
- PHP's built-in file-upload functions: is_uploaded_file() and move_uploaded_file()
- In order to handle file upload, the form must use POST method and use multipart/form-data for the enctype



```
<form action="" enctype="multipart/form-data" method="post">
    <label form="email">Email:</label><br>
    <input type="text" name="email" value=""><br>

    <label form="lastname">Last Name:</label><br>
    <input type="text" name="lastname" value=""><br>

    <label form="classnotes">Class notes:</label><br>
    <input type="file" name="classnotes" value=""><br>

    <input type="submit" name="submit" value="Submit Notes">
</form>
```

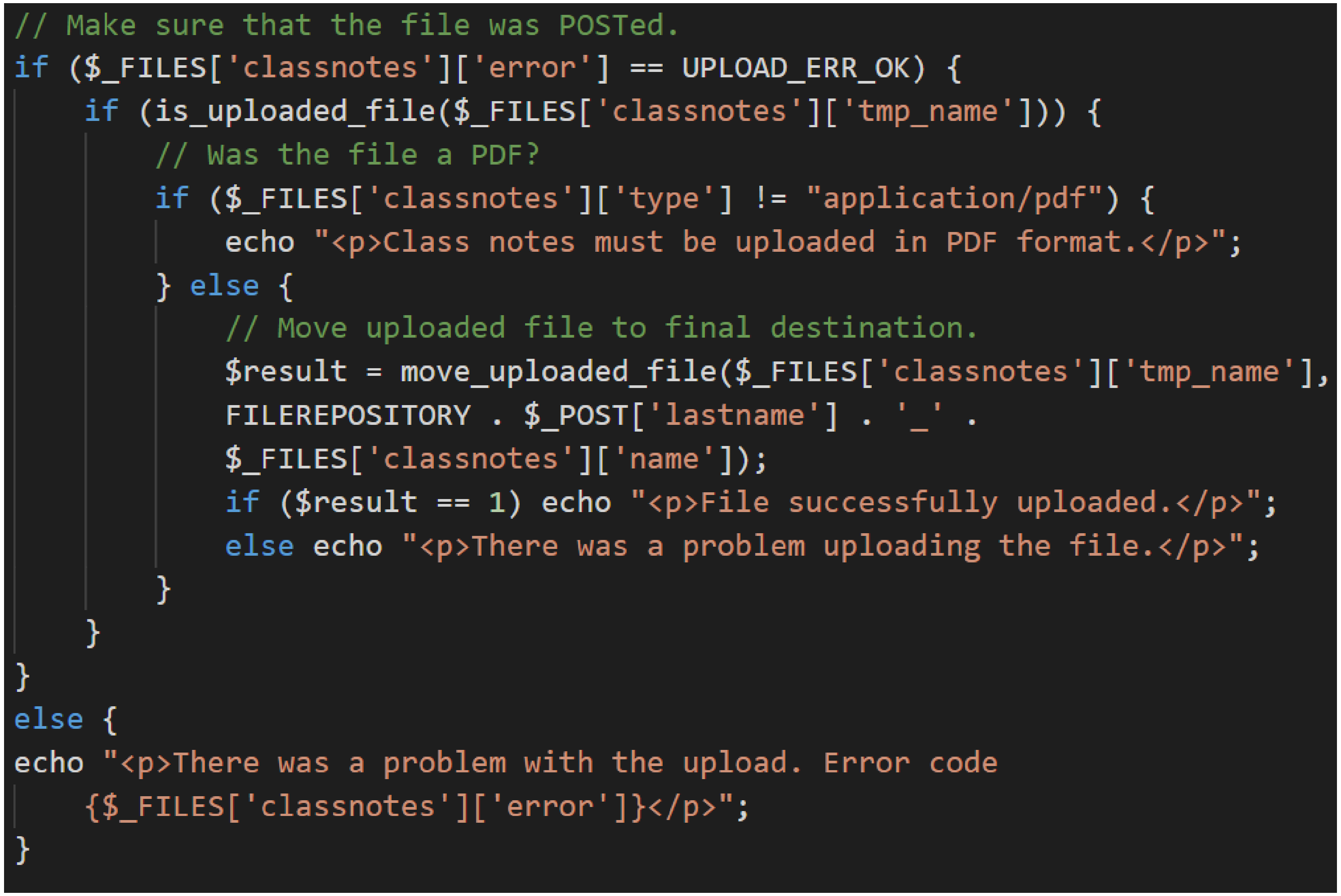
Uploading a File

- php.ini directives related to uploading file needs to be set appropriately
- The user running the webserver process must have the write permission to the folder
- Never write a file into the configuration folder of your webserver
- $_FILES array
  - $_FILES['userfile']['error'] – provide the result of the upload
  - $_FILES['userfile']['name'] – provide the original filename
  - $_FILES['userfile']['size'] – specifies the file size
  - $_FILES['userfile']['type'] – specifies the file MIME type
  - $_FILES['userfile']['tmp_name'] – specifies the temporary name when the file was uploaded

Uploading a File

- Process
  - Check if the form was submitted or not
  - Check whether the file was uploaded successfully
  - Check whether the file is indeed the one uploaded using HTTP_POST (not the file in the directory)
  - Check the file type (optional)
  - Check the file size (optional)
  - Rename the file (optional)
  - Move the file to the correct location in the server

Uploading a File



```php
// Make sure that the file was POSTed.
if ($_FILES['classnotes']['error'] == UPLOAD_ERR_OK) {
    if (is_uploaded_file($_FILES['classnotes']['tmp_name'])) {
        // Was the file a PDF?
        if ($_FILES['classnotes']['type'] != "application/pdf") {
            echo "<p>Class notes must be uploaded in PDF format.</p>";
        } else {
            // Move uploaded file to final destination.
            $result = move_uploaded_file($_FILES['classnotes']['tmp_name'],
            FILEREPOSITORY . $_POST['lastname'] . '_' .
            $_FILES['classnotes']['name']);
            if ($result == 1) echo "<p>File successfully uploaded.</p>";
            else echo "<p>There was a problem uploading the file.</p>";
        }
    }
}
else {
echo "<p>There was a problem with the upload. Error code
    {$_FILES['classnotes']['error']}</p>";
}
```

Problem with Files

- Files do not scale well
- Searching files is slow
- Concurrent access is problematic
- Security is a problem, limited to file permissions
- Use an RDBMS!

Lab 3

- Download Lab3 file from the Blackboard
- You need to submit it next week in the morning of the day of next class
- **Make sure to follow the submission guideline!**