# CHAPTER V: WORKING WITH FILES AND DIRECTORIES IN PHP

**Introduction**

Working with files and directories is an essential aspect of system management and programming. This chapter covers key concepts such as file types, permissions, directory manipulation, file I/O operations, and file management techniques in PHP. Each section provides detailed explanations and practical examples to ensure comprehensive understanding.

## 1. File Types and Permission

**File Types:**

- **Regular Files:** Text files, program source code, images, PDF documents, or binary data.
- **Directory Files:** Containers that store other files and directories.
- **Special Files:** Includes device files used for hardware interactions.
- **Symbolic Links:** Reference files that point to other files or directories.

**File Permissions:**

- **Read (r):** Allows users to view the file's content.
- **Write (w):** Enables users to modify or delete the file.
- **Execute (x):** Grants permission to run the file as a program.

**PHP Command for Changing Permissions:**

```
chmod("filename.pdf", 0755); // Owner can read, write, and execute; others can only read and execute
```

## 2. Reading and Creating Directories

**Reading Directory Content:**

```
$dir = "./";
if (is_dir($dir)) {
  if ($dh = opendir($dir)) {
    while (($file = readdir($dh)) !== false) {
      echo "Filename: $file<br>";
    }
    closedir($dh);
  }
}
```

**Explanation:**

- is_dir() checks if the given path is a valid directory.
- opendir() opens the directory for reading.
- readdir() iterates through the directory's contents.

**Creating Directories in PHP:**

```
mkdir("new_directory", 0777, true); // 0777 gives full permissions, 'true' creates nested directories if
needed
```

**3. Retrieving the File Information**

**Sample Code to Retrieve File Information:**

```
$file = "example.pdf";

if (file_exists($file)) {

    echo "File size: " . filesize($file) . " bytes";<br>

    echo "Last modified: " . date("F d Y H:i:s.", filemtime($file));

}
```

**Explanation:**

- file_exists() checks if the file exists.
- filesize() returns the file's size in bytes.
- filemtime() retrieves the file's last modification date.

**4. Downloading the File**

**Downloading Using PHP:**

```
$file_url = 'https://example.com/sample.pdf';

header('Content-Type: application/pdf');

header('Content-Disposition: attachment; filename="downloaded_sample.pdf"');

readfile($file_url);
```

**Explanation:**

- header() sets the file type and forces download with a specified filename.
- readfile() reads the file content directly from the URL and sends it to the browser for download.

**5. Writing and Reading Entire File**

**Writing to a File:**

```
file_put_contents("example.json", json_encode(["name" => "Sample", "status" => "Active"]));
```

**Reading from a File:**

```
$content = file_get_contents("example.json");

$data = json_decode($content, true);

print_r($data);
```

**Explanation:**

- file_put_contents() writes text directly to a file (overwrites existing content).
- file_get_contents() reads the entire content of a file as a string.
- json_encode() and json_decode() are used for handling structured data.

**6. Copying and Moving Files**

**Copying Files Using PHP:**

```
copy("source.pdf", "destination.pdf");
```

**Moving Files:**

```
rename("source.pdf", "new_directory/source.pdf");
```

**Explanation:**

- copy() duplicates the file to the specified location.
- rename() moves or renames a file in one step.

**7. Renaming and Removing Files**

**Renaming Files:**

rename("old_name.docx", "new_name.docx");

**Removing Files:**

unlink("file_to_delete.docx");

**Explanation:**

- rename() renames a file or directory.
- unlink() permanently deletes a file.

**Important Note:** Ensure that files are closed before attempting to rename or delete them to prevent errors.

**Conclusion**

Mastering file and directory operations is crucial for efficient programming and system management. By understanding file types, permissions, and common file manipulation techniques, developers can enhance their coding practices and ensure efficient file handling. Detailed use of PHP functions ensures flexibility in creating dynamic web applications that manage files effectively.

## Working Example:

```
<?php
// Complete File Management System in PHP

// 1. Create a Directory
function createDirectory($dirName) {
    if (!file_exists($dirName)) {
        mkdir($dirName, 0777, true);
        echo "Directory '$dirName' created successfully.<br>";
    } else {
        echo "Directory '$dirName' already exists.<br>";
    }
}

// 2. Write Data to a File
function writeFile($filename, $content) {
    file_put_contents($filename, $content);
    echo "Data written to '$filename' successfully.<br>";
}
```

```
// 3. Read Data from a File
function readFileContent($filename) {
    if (file_exists($filename)) {
        echo "Content of '$filename':<br>";
        echo nl2br(file_get_contents($filename)) . "<br>";
    } else {
        echo "File '$filename' does not exist.<br>";
    }
}

// 4. Copy a File
function copyFile($source, $destination) {
    if (copy($source, $destination)) {
        echo "File copied from '$source' to '$destination'.<br>";
    } else {
        echo "Failed to copy file.<br>";
    }
}

// 5. Move a File
function moveFile($source, $destination) {
    if (rename($source, $destination)) {
        echo "File moved to '$destination'.<br>";
    } else {
        echo "Failed to move file.<br>";
    }
}

// 6. Delete a File
function deleteFile($filename) {
    if (file_exists($filename)) {
        unlink($filename);
        echo "File '$filename' deleted successfully.<br>";
    } else {
        echo "File '$filename' does not exist.<br>";
    }
}

// Example Usage
createDirectory("test_dir");
writeFile("test_dir/sample.txt", "Hello, this is a sample file.");
```

```php
readFileContent("test_dir/sample.txt");

copyFile("test_dir/sample.txt", "test_dir/copy_sample.txt");

moveFile("test_dir/copy_sample.txt", "test_dir/moved_sample.txt");

deleteFile("test_dir/moved_sample.txt");

?>
```