Government Arts and Science College (Women),

Sathankulam – 628704

Class : II B.Sc. (Computer Science)

Semester : IV

Subject : Programming with PHP &MySQL

Topic : Unit III

Faculty : Mrs. A. Angeline Nancy Sophia M.Sc., M.Phil.,
Department of Computer Science

File Handling

- Opening files using fopen

  to start working with a file, we must first open that file

Syntax:

$filehandle = fopen(filename, mode [,use_include_path
[, zcontext]])

where filename is the name of file, mode indicates how to open the file, use_include_path is set to 1 or TRUE to include path, zcontext holds optional file context.

Modes:

r - open for reading only

r + - open for reading and writing

w - open for writing only. If the file does not exist, it create it

w+ - open for reading and writing. If the file does not exist, it create it

a – open for appending only

a+ - open for reading and writing, starting at the end of the file.

x – create and open for write only. If already exists it return false

x+ - create and open for reading and writing. If already exists it return false

Example

$handle = fopen("home/file.txt", "r");

When we open a file, we get a file handle which represents an open file. We use this handle to work with the file.

```
<html>
<head>
<title> Opening a file</title>
</head>
<body>
<h1>Opening a file</h1>
<?php
$handle = fopen("file.txt", "r");
if ($handle) {
        echo "File opened OK";
}
?>
</body>
</html>
```

Looping over a file's content with feof

To read multiple lines in a file we can use a while loop and feof function.

```
<html>
<head>
<title> Reading a file</title>
</head>
<body>
<h1>Reading a file</h1>
<?php
$handle = fopen("file.txt", "r");
while (!feof($handle)) {
. . .
}
?>
</body>
</html>
```

Closing a file

we should close the file to free up the resources connected with that file.

Syntax:

fclose($filehandle);

it returns true if file was closed successfully and false otherwise.

```
<html>
<head><title> Reading from a file</title>
</head>
<body>
<h1>Reading from a file</h1>
<?php
$handle = fopen("file.txt", "r");
while (!feof($handle)) {
    $text = fgets($handle);
echo $text, "<br>";
}
fclose($handle);
?>
</body></html>
```

Reading from a file using fgets

we use fgets function to get a string of text from a file.

Syntax:

fgets(handle [,length])

The function returns a string of up to length-1 bytes read from the file. Reading ends when length-1 bytes have been read, on a newline or encountering the end of file.

```
<html>
<head><title> Reading from a file</title>
</head>
<body>
<h1>Reading from a file</h1>
<?php
$handle = fopen("file.txt", "r");
while (!feof($handle)) {
        $text = fgets($handle);
echo $text, "<br>";
}
?>
</body></html>
```

Output

Reading from a file
Here
is
your
data

Reading character by character with fgetc

we can read individual characters from a text file using the fgetc function.

Syntax:

fgetc($filehandle);

```
<html>
<head><title> Reading characters from a file</title>
</head>
<body>
<h1>Reading characters from a file</h1>
```

```
<?php
$handle = fopen("file.txt", "r");
while ($char = fgetc($handle)) {
        if($char == "\n"){
                $char = "<br>";
}
fclose($handle);
?>
</body></html>
```

Output

Reading characters from a file
Here
is
your
data

Reading whole file at once

we can read the entire contents of a file with file_get_contents function.

Syntax:

file_get_contents(filename [,use_include_path
[,context [,offset [, maxlen]]]])

where filename is the name of the file, use_include_path is set to TRUE if we want to include path, context is context for operation, offset s the offset into the file at which to start reading, and maxlen is the maximum length of data to read.

Example

```
<html>
<head><title> Reading a whole file</title>
</head>
<body>
<h1>Reading a whole file</h1>
<?php
$text = file_get_contents("http://www.php.net");
$fixed_text = str_replace("\n", "<br>", $text);
```

```
echo $fixed_text;
}
?>
</body></html>
```

Output

Reading a whole file at once
Here
is
your
data

Reading a file into an array

we can use file function to read a file into an array all at once; each line becomes an element in an array.

Syntax:

file(filename [, use_include_path [, context]])

filename is the name of the file, use_include_path set to TRUE to include path for the file, context is a context for the operation. The function returns an array or FALSE if the operation failed.

Example

```
<html>
<head><title> Reading  a file into an array</title>
</head><body>
<h1> Reading  a file into an array </h1>
<?php
$data = file("file.txt");
foreach ($data as $number => $line){
        echo "Line Number: ", $line, "<br>";
}
?>
</body></html>
```

Output

Reading a file into an array
Line 0 :Here
Line 1 :is
Line 2 :your

Line 3:data

Checking if a file Exists

we can check if a file exists with the file_exists_function.

Syntax:

file_exists (filename)

it returns TRUE if the file exists, FALSE otherwise.

```
<html>
<head><title> Checking if a file exists</title>
</head>
<body>
<h1> Checking if a file exists </h1>
<?php
$filename = “does_not_exist.txt”);
if (file_exists($filename)){
$data = file($filename);
foreach ($data as $number => $line){
        echo “Line Number: “, $line, “<br>”;
}
}
else  {
        echo “The file $filename does not exist”;
}
?>
</body></html>
```

Output

Checking if a file exists

The file does_not_exist.txt does not exist

Getting File Size

we can get a file’s size using the filesize function. It returns an integer.

Syntax:

filesize (filename)

we just pass the filename to filesize and we get an integer or FALSE if the file doesn’t exist.

```
<html>
<head>
```

```
<title> Getting File size</title></head>
<body>
<h1>Getting file size </h1>
<?php
echo "The file file.txt is ", filesize("file.txt), "bytes long.";
?>
</body></html>
```

Output

Getting file size

The file file.txt is 24 bytes long.

Reading binary files

we can also read binary files using fread function

Syntax:

fread(handle, length)

this function reads up to length bytes from the file referenced by handle. We should open files for binary reading with mode 'rb'

Example

```
<html>
<head>
<title> Reading binary data</title></head>
<body>
<h1>Reading binary data</h1>
<?php
        $handle = fopen("file.txt", "rb");
        $text = fread($handle, filesize("file.txt"));
        $fixed_text = str_replace("\n", "<br>", $text);
Echo $fixed_text;
Fclose($handle);
?>
</body></html>
```

Output

Reading binary data

Here

is

your

data

Parsing file using fscanf

fscanf(handle, format)

this function takes a file handle and a format string.

actors.txt

Cary Grant
Hyrna Loy
Jimmy Stewart
June Allyson

in this file the actors first name and last names are separated by tabs.

```
<html>
<head>
<title> Parsing file</title>
</head>
<body>
<h1> Parsing Files</h1>
<?php
  $handle = fopen("actors.txt","r");
        while ($name = fscanf($handle, "%s\t"%s\n")){
                list($firstname, $lastname) = $name;
                echo $firstname, " ", $lastname, "<br>";
        }
fclose($handle);
?>
</body>
</html>
```

Output

Parsing Files
Cary Grant
Hyrna Loy
Jimmy Stewart
June Allyson

Parsing ini files with parse_ini_file

.ini -> initialization files

parse_ini_file(filename [,process_sections])

this function loads in the ini file specified in filename and return the settings in an associative array. By setting process_sections to TRUE we get a multidimensional array with section names and settings included. The default value is FALSE

Sample.ini

```
[first_section]
first_color = red
second_color = white
third_color = blue
[second_section]
file  = "/usr/local/code.data"
URL = "http://www.php.net"
```

```
<html>
<head>
<title> Parsing .ini files </title>
<body>
<h1> Parsing .ini files</h1>
<?php
  $array = parse_ini_files("sample.ini");
        foreach ($array as $key => $value) {
                echo "$key => $value <br>";
}
?>
</body>
</html>
```

Output

```
Parsing .ini files
first_color => red
second_color => white
third_color = >blue
file  => "/usr/local/code.data"
URL =>"http://www.php.net"
```

Getting file info using stat

stat function gives the information about a file

Syntax:

stat(filename)

0 (dev) device number

1 (ino) Inode number

2 (mode) Inode protection mode

3 (nlink) number of links

4 (uid) user id of the owner

5 (gid) group id of the owner

6 (rdev) Device type

7 (size) size in bytes

8 (atime) Time of last access

9 (mtime) time of last modification

10 (ctime) Time of last inode change

11 (blksize) block size of filesystem

12 (blocks) number of blocks allocated

```
<html>
<head>
<title> Using stat</title>
</head>
<body>
<h1>Using Stat</h1>
<?php
     $array = stat("file.txt");
    echo "The file is ", $array["size"]. "bytes long";
 ?>
</body>
</html>
```

Output

Using stat

The file is 24 bytes long

Setting File pointer Location using fseek

we can use file pointers to keep track of where it is in file and where the next read or write operation occurs from.

Syntax:

fseek(handle, offset, [start_point]);

handle is the handle of the file to set the file pointer in, offset is the number of bytes to set the pointer, start_point indicates a starting point for the pointer has the following constants:

SEEK_SET – beginning of the file

SEEK_CUR – current pointer location

SEEK_END – end of file

Copying files with copy function

we can copy files with the copy function

Syntax:

copy(source, destination)

source is the name of the source file, destination is the name of the name of the copy. It returns TRUE if successful, FALSE otherwise.

```
<html>
<head>
<title>Copying files</title>
</head>
<body>
<h1>Copying files</h1>
<?php
  $file = "file.txt";
  $copy = "copy.txt";
  if (copy($file, $copy)) {
      echo "Copied $file";
  }
  else {
     echo "could not copy $file.";
  }
 ?>
</body>
</html>
```

Output

Copying files

copied file.txt

Deleting files with unlink

we can delete a file using unlink function

Syntax:

unlink (filename [,context])

Filename is the name of the file and context is an optional context. It returns TRUE if the file was deleted, FALSE otherwise

```
<html>
<head>
<title>Deleting files</title>
</head>
<body>
<h1>Deleting files</h1>
<?php
  if (unlink("copy.txt")){
        echo "Deleted the file.";
  }
  else {
     echo "could not delete the file.";
  }
 ?>
</body>
</html>
```

Output

Deleting files

Deleted the file

Writing a File with fwrite

we use fwrite to write a string to a file.

Syntax:

fwrite(handle, string [,length])

we pass fwrite a file handle, the string to write and optionally the maximum length of data to write.

the function returns the number of bytes written or FALSE if there was an error.

```
<html>
<head>
<title>Writing Files</title>
</head><body>
<h1>Writing Files</h1>
<?php
        $handle = fopen("data.txt" , "w");
        $text = "Here \nis\nthe\ntext.";
        if (fwrite($handle, $text) == FALSE) {
                echo "cannot write data.txt.";
        }
else  {
                echo "Created data.txt.";
}
fclose($handle);
?>
</body></html>
```

Output

```
Here
is
the
text
```

Reading and writing Binary Files

we can write binary data with fwrite and read it with fread. We can use pack function to pack binary data into strings and unpack binary data using unpack function.

```
<html>
<head><title>Writing binary files</title></head>
<body>
<h1>Writing binary files</h1>
<?php
   $number = 512;
        $handle = fopen("data.dat", "wb");
        if (fwrite( $handle ,pack("L", $number)) == FALSE) {
```

```
                echo "Cannot write data.dat";
   }
  else  {
                echo "Created data.dat and stored $number.";
  }
fclose($handle);
?>
</body></html>
```

Output

Writing binary files

Created data.dat and stored 512

```
<html>
<head><title>Reading binary files</title></head>
<body>
<h1>Reading binary files</h1>
<?php
    $handle = fopen("data.dat", "rb");
    $data = fread($handle, 4);
   $array = unpack("Ldata", $data);
   $data = $array["data"];
         echo "Read this value from data.dat:", $data;
    fclose($handle);
?>
</body></html>
```

Output

Reading binary files

Read this value from data.dat : 512

Appending files with fwrite

we can also append data to files using fwrite if we open the file for appending explicitly.

```
<html>
<head><title>Appending to files</title></head>
<body>
<h1>Appending to files</h1>
```

```
<?php
        $handle = fopen("data.txt", "a");
  $text = \nHere\nis\nmore\ntext.";
        if (fwrite( $handle, $text == FALSE)  {
                echo "Cannot  append to data.txt";
   }
  else  {
                echo "Appended to data.txt.";
 }
fclose($handle);
?>
</body></html>
```

Output

```
Appending to files
Appended to data.txt
Here
is
the
text.
Here
is
more
text.
```


Writing a file all at once with file_put_contents

we can write entire file using file_put_contents function.

file_put_contents(filename, data [,flags [, context]])

filename is the name of the file, data is string of text to write, flags can be either FILE_USE_INCLUDE_PATH or FILE_APPEND and context is a file context.

the function returns the number of bytes that were written to files or FALSE if it couldn't write to the file.

```
<html>
<head><title>Writing files with file_put_contents</title></head>
<body>
<h1> Writing files with file_put_contents </h1>
```

```
<?php
  $text = "Here\nis\nthe\ntext.";
 if( file_put_contents("data.txt", $text)==FALSE) {
       echo "Cannot write data.txt.";
        }
else {
   echo "Wrote to data.txt.";
 }
?>
</body>
</html>
```

Output

Writing files with file_put_contents

Wrote to data.txt

Locking Files

In a multiuser environment, multiple users may be accessing our scripts at the same time which means multiple copies of the same script can operate at the same time. There may be conflicts as two scripts or two copies of the same script tries to write the same file at the same time. To fix that we use the file locking function.

flock(handle, operation [,&wouldblock])

handle is the handle of the file to lock and operation is:

LOCK_SH – to acquire shared lock

LOCK_EX – to acquire an exclusive lock

LOCK_UN – to release lock

The third argument is set to true if the lock would block

```
<html>
<head> <title> Locking and unlocking files</title> </head>
<body>
<h1> Locking and unlocking files </h1>
<?php
  $handle=fopen("data.txt", "w");
  $text = "Here\nis\nthe\ntext.";
```

```
 if (flock( $handle, LOCK_EX| LOCK_NB)) {
       echo “Locked the file. <br>”;
             if (fwrite($handle, $text) == FALSE) {
                   echo “Cannot write data.txt. <br>”;
             }
else {
 echo “Created data.txt. <br>”;
 }
flock($handle, LOCK_UN);
echo “Unlocked the file.<br>”;
}
else {
echo “Could not lock the file. <br>”;
}
fclose($handle);
?>
</body> </html>
```

Output

Locking and unlocking files

Locked the file
Wrote to data.txt
Unlocked the file