In PHP programming, file is a place on your physical disk where information is stored.

Why files are needed?

- When a program is terminated, the entire data is lost. Storing in a file will preserve your data even if the program terminates.
- If you have to enter a large number of data, it will take a lot of time to enter them all. However, if you have a file containing all the data, you can easily access the contents of the file using few commands in PHP.
- You can easily move your data from one computer to another without any changes.

Types of Files

There are two types of files you should know about.

1. Text files
2. Binary files

1. Text files

Text files are the normal .txt files that you can easily create using Notepad or any simple text editors. When you open those files, you will see all the contents within the file as plain text. You can easily edit or delete the contents. They take minimum effort to maintain, are easily readable, and provide least security and takes bigger storage space.

2. Binary files

Binary files are mostly the .bin, exe, com, obj, etc files in your computer. Instead of storing data in plain text, they store it in the binary form (0's and 1's). They can hold higher amount of data, are not readable easily and provides a better security than text files.

File Operations

In PHP, A binary file is a series of characters or bytes for PHP attaches no special meaning. Any structure to the data is determined by the application that reads from or writes to the file. A text file, in contrast, is assumed to have only printable characters and a small set of control or formatting characters. The Formatted characters are the binary equivalents of the escape sequences listed in following table.

Escape Sequences	Meaning	Decimal Code	Octal	Hexadecimal
\r	Line Feed	10	012	0A or \x0A
\n	Carriage return	13	015	00 or \x0F
\t	Horizontal tab	9	011	09 or \x09
\v	Vertical tab	11	013	0B or \x0B
\f	Form feed	12	012	0C or \x0C

Different operating system use different escape sequences to identify the end of a line. UNIX/Linux platforms use the \n carriage return escape sequence, and Macintosh application usually use the \r line feed escape sequence, and windows operating system use the \n carriage return escape sequence followed by the \r line feed escape sequence.

- Escape sequence "**\n"** use for ***end- of- line*** in Unix/Linux Operating System.
- Escape sequences "**\n\r"** combine use for ***end- of- line*** in Windows Operating System.
- Escape sequence "***\r"*** use for ***end-of- line*** in Macintosh OS.

You can perform four major operations on the file, either text or binary:

1. Creating a new file
2. Opening an existing file
3. Closing a file
4. Reading from and writing information to a file

Working with files

When working with files in PHP, you need to declare a pointer of type file. This declaration is needed for communication between the file and program.

Syntax: ***$filehandler = fopn(filename, mode)***

PHP provides a number of functions that helps to perform basic file operations. Following are the functions:

Functions	Description
fopen()	Create a new file or open an existing file
fclose()	Closes a file
fwrite()	Writes the contents of string to the file stream pointed to by handle.
fread ()	Read content or string from file
fgetc()	Reads a character from a file
fgets()	Read string or line from a file
fprintf ()	Writes set formatted data to a file
fscanf ()	Reads formatted data from a file
file_exists()	File check in file system exist return value true or false
feof ()	It checks if the ***"end-of-file"*** has been reached or not return true or false value.

Opening a file: (for creation and edit)

To open a file you need to use the ***fopen ()*** function, which returns a ***FileHandler.*** Once you have opened a file, you can use the file handler pointer to let the script server r perform input and output functions on the file.

Syntax: ***$FileHandler = fopen(filename, mode);***

Mode	Description
r	Opens a text file in reading mode
w	Opens or create a text file in writing mode.
a	Opens a text file in append mode
r+	Opens a text file in both reading and writing mode
w+	Opens a text file in both reading and writing mode
a+	Opens a text file in both reading and writing mode
rb	Opens a binary file in reading mode
wb	Opens or create a binary file in writing mode
ab	Opens a binary file in append mode
rb+	Opens a binary file in both reading and writing mode
wb+	Opens a binary file in both reading and writing mode
ab+	Opens a binary file in both reading and writing mode

Example:

```
<?php
        $Filehandler = fopen("\\temp\\abc.txt", "r");
?>
```

This code will open test.txt for reading in text mode. To open a file in a binary mode you must write “rb” to the mode string.

Closing a File:

To close a file, use the **fclose** ( ) function.

Syntax: *fclose($FileHandler);*

Here ***fclose( )*** function closes the file and returns zero on success.

The ***feof()*** function in PHP is an inbuilt function which is used to test for the ***end-of-file*** on a file handler. It checks if the ***“end-of-file”*** has been reached or not return true or false value. The ***feof()*** function is used for looping through the content of a file if the size of content is not known beforehand.

Syntax: ***feof($FileHandler);***

Input and Output operation on File

Reading and writing to a binary/text file

The Functions ***fread()*** and ***fwrite()*** are used for reading from and writing to a file on the disk respectively in case of binary files.

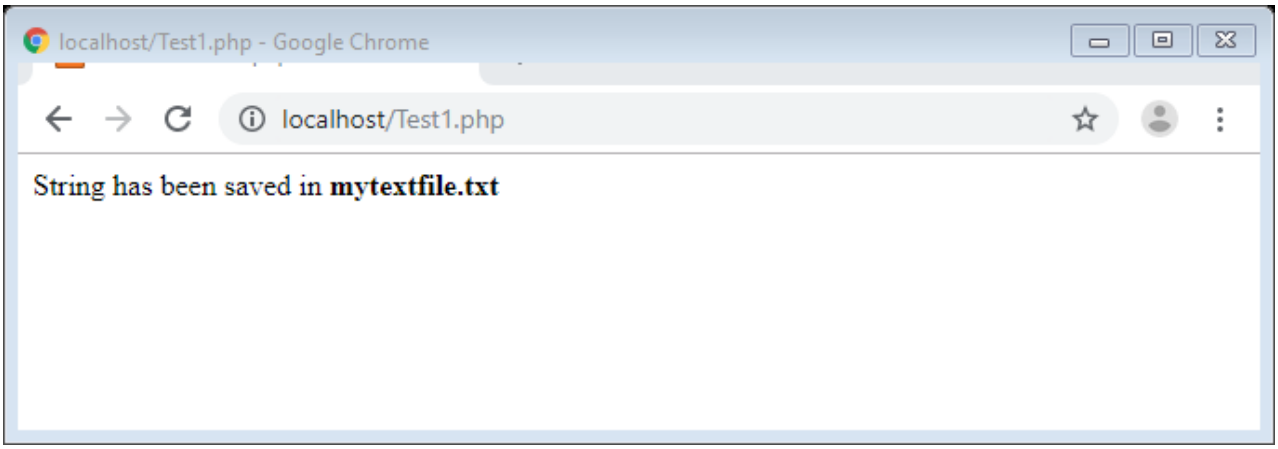
fwrite() writes the contents of string to the file stream pointed to by handle. It has three parameters.

Syntax : fwrite (filepointer, string, length)

- ***filepointer*** A file system pointer resource that is typically created using ***fopen ()***.
- ***string*** The string that is to be written.
- ***length*** If the length argument is given, writing will stop after length bytes have been written or the end of string is reached, whichever comes first.

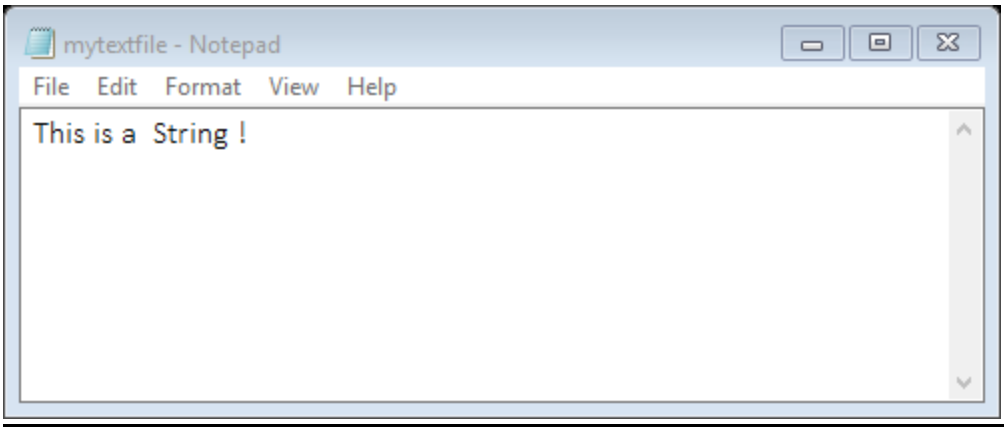
Example : Writing to a binary file using fwrite()

```
<?php
   $filehandler = fopen("mytextfile.txt","w");
   fwrite($filehandler, "This is a  String !");
   echo "String has been saved in mytextfile.txt<br>";
  fclose($filehandler);
?>
```



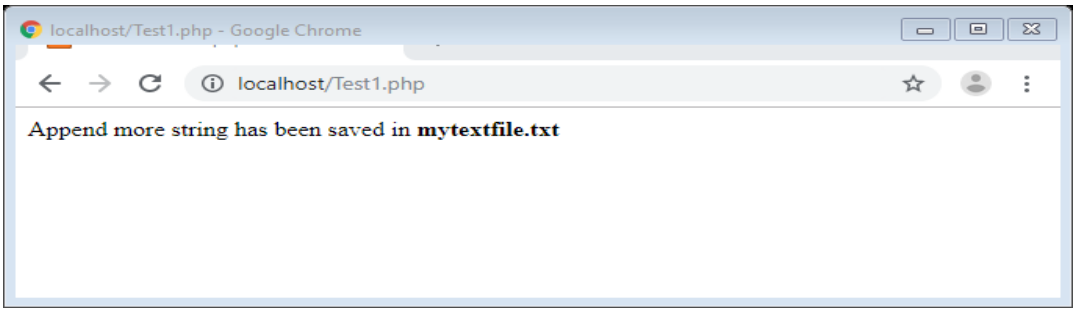
After this program execute on browser, we go to the ***htdocs*** folder, where see ***mytextfile.txt*** will be create in the ***htdocs*** folder. We open this file on the notepad than display the content,

"This is a String !";

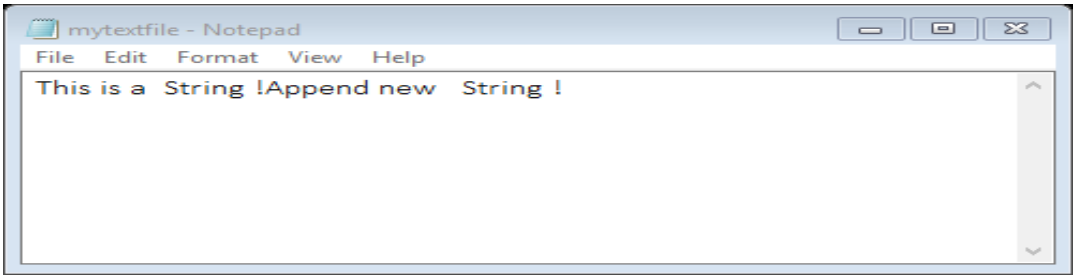


Example : Append data to a binary file using fwrite()

```
<?php
 $filehandler = fopen("mytextfile.txt","a");
  fwrite($filehandler, "Append new   String !");
  echo "Append more string has been saved in <b>mytextfile.txt </b> <br>";
  fclose($filehandler);
?>
```



Again, this program execute on browser, we go to the ***htdocs*** folder, where see ***mytextfile.txt*** will be ubdate in the ***htdocs*** folder. We open this file on the notepad than display the content "This is a string ! Append new String !".

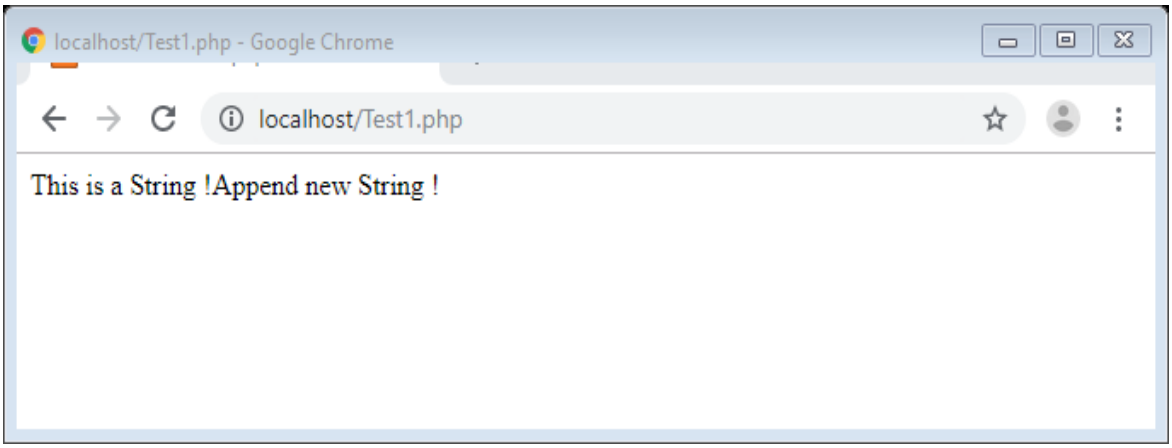


The ***fread ()*** reads up to length bytes from the file pointer . Reading stops as soon as one of the conditions are length bytes have been read , EOF (end of file) is reached and a packet becomes available or the socket timeout occurs (for network streams), or FALSE on failure.

Syntax : ***fread ($filepointer, length)***

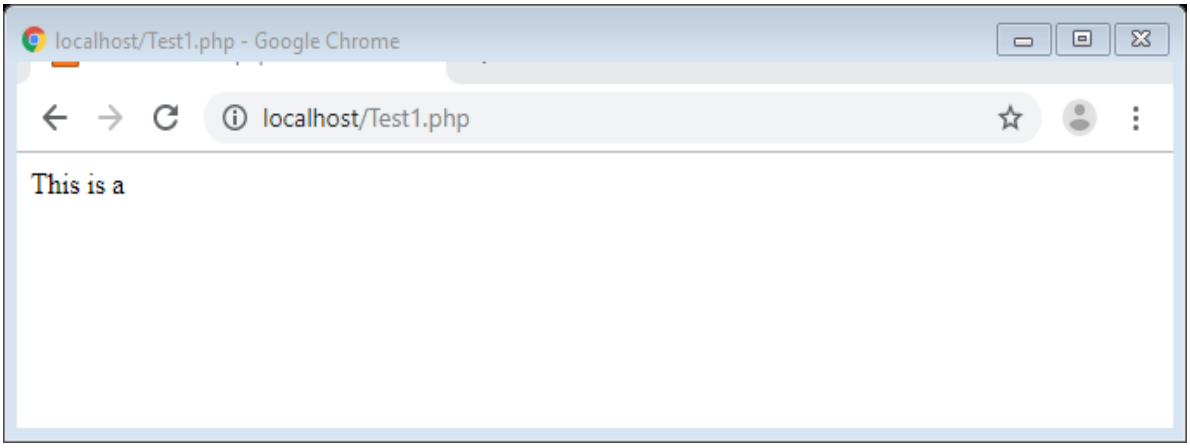
Example : Reading from a binary/Text file using fread()

```
<?php
    $filehandler = fopen("mytextfile.txt","r");
    echo fread($filehandler,filesize("mytextfile.txt"));   // read entire file contents.
    fclose($filehandler);
?>
```



Example: Read only 10 bytes

```
<?php
  $filehandler = fopen("mytextfile.txt","r");
  echo fread($filehandler,"10");
  fclose($filehandler);
?>
```

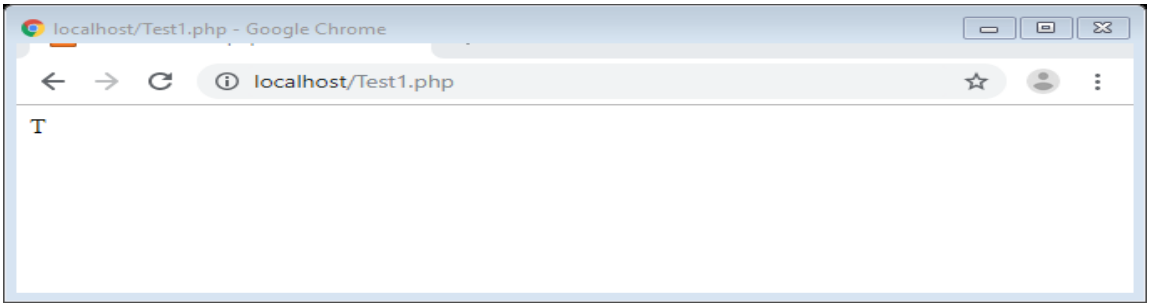


fgetc() is simplest functions used to read individual characters to a file.

Syntax: $varaible = fgetc(filehandler)

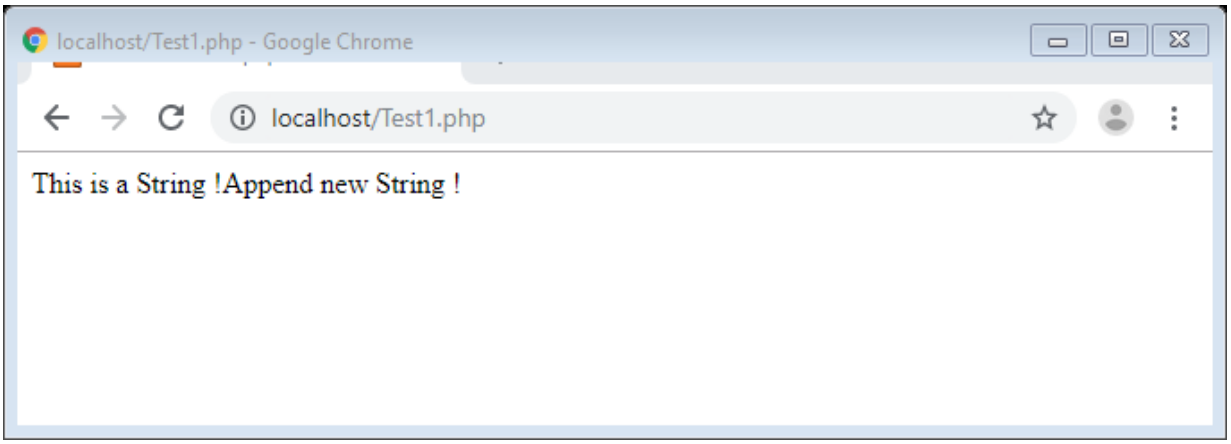
Example: Read one character from mytextfile.txt file.

```
<?php
 $filehandler = fopen("mytextfile.txt","r");
 echo fgetc($filehandler);     // only read one character from mytext.txt file.
 fclose($filehandler);
?>
```



Example: Read entire file character by character from mytextfile.txt file.

```
<?php
 $filehandler = fopen("mytextfile.txt","r");
 while (! feof ($filehandler))  // read character by character  from mytext.txt file until end of file.
 {
     echo fgetc($filehandler);
 }
 fclose($filehandler);
?>
```

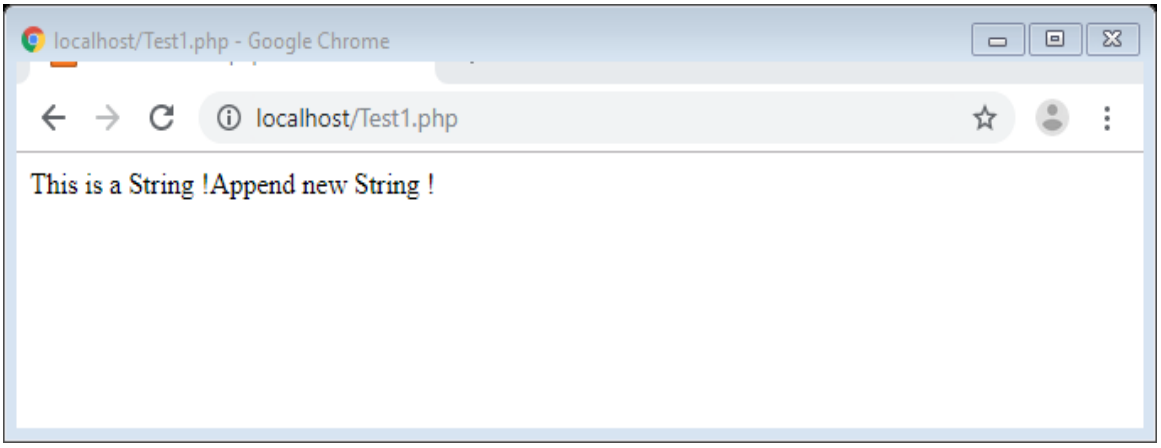


fgets() function returns a line from an open file. The ***fgets()*** function stops returning on a new line, at the specified length, or at eof, whichever comes first. This function returns FALSE on failure.

Syntax: fgets (filepointer, [length])

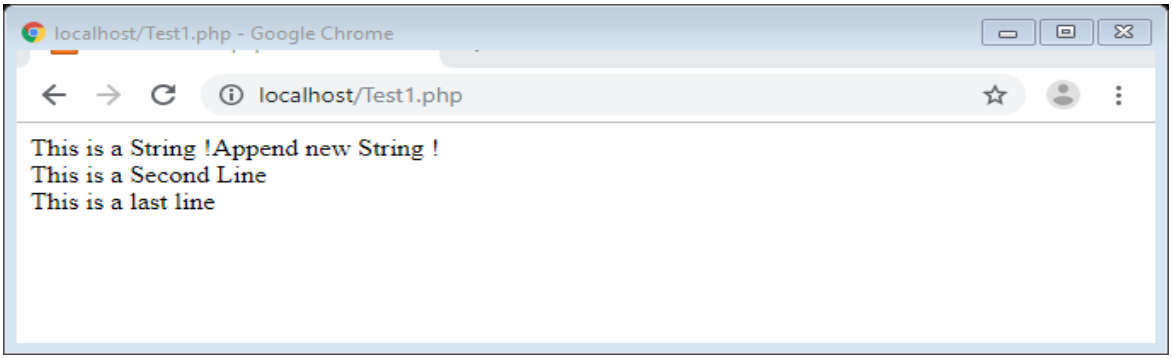
Example -1

```
<?php
  $filehandler = fopen("mytextfile.txt","r");
  echo fgets($filehandler);
  fclose($filehandler);
?>
```



Example-2: *Read complete data from file but line by line.*

```
<?php
   $filehandler = fopen("mytextfile.txt","r");
   while(! feof($filehandler))
   {
     echo fgets($filehandler). "<br />";
   }
  fclose($filehandler);
?>
```



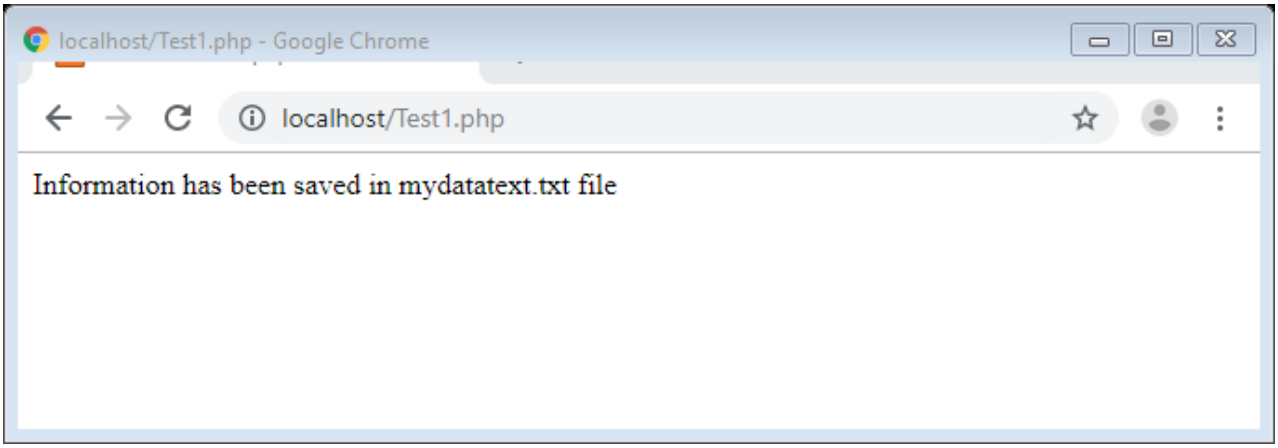
Reading and Writing formatted data from File using fprintf() and fscanf() functions

fprintf() function writes a formatted string to a specified output stream (example: file or database).The arg1, arg2, ++ parameters will be inserted at percent (%) signs in the main string. This function works "step-by-step". At the first % sign, arg1 is inserted, at the second % sign, arg2 is inserted, etc.

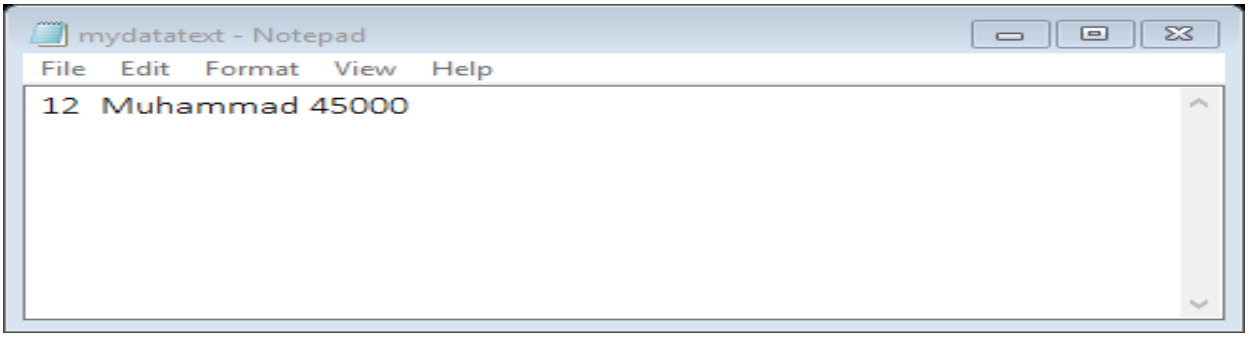
Syntax: fprintf($filehandler,"formatted spacifier , .. ", $arg1,..)

Example: using fprintf () function

```
<?php
  $code = 12;
  $name = "Muhammad";
  $salary =45000;
  $filehandler = fopen("mydatatext.txt","w");
  fprintf($filehandler, "%u  %s %u\n" , $code, $name, $salary);
  echo "Information has been saved in mydatatext.txt file<br>";
  fclose($filehandler);
?>
```



And we open ***mytextfile.txt*** on notpad, we will see

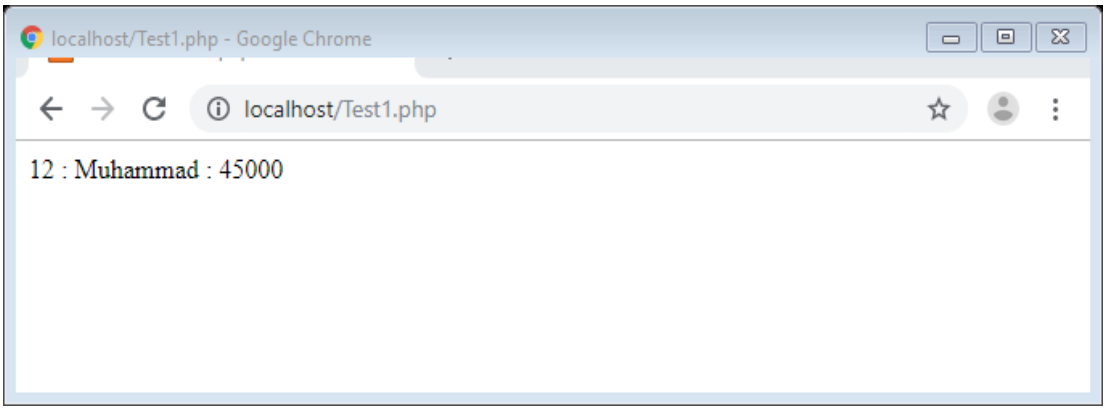


fscanf() function parses the input from an open file according to the specified format.

Syntax: fscanf($filehandler,"formatted spacifier , .. ", $arg1,..)

Example: using fscanf () function

```
<?php
  $filehandler = fopen("mydatatext.txt", "r");
  while ($userinfo = fscanf($filehandler, "%u %s %u\n"))
  {
    list ($code, $name, $salary) = $userinfo;
    echo "$code  : $name :  $salary  <br>";
  }
  fclose($filehandler);
?>
```



file_exists () function in PHP is an inbuilt function which is used to check whether a file or directory exists or not. The ***file_exists ()*** function in PHP accepts only one parameter *path*. It specifies the path of the file or directory you want to check.

Syntax: ***file_exists(path)***

Example

```
<?php
   if  ( file_exists("mytextfile.txt") )
         echo " Given File is Present ";
   else
         echo "Given file is Not Exist";
?>
```

Note: The ***file_exists()*** function For files larger than 2gb, some of the file system functions may give unexpected results since PHP's integer type is signed and many platforms use 32bit integers.

Working with files and folders of File Systems

In PHP, you should become familiar with how PHP works with directories. By knowing how to create, read and move between directories, you can read the names of files and directories that exist within any specified directory for which you have the appropriate permissions. To read the content of directory, you use the PHP functions listed are following.

Functions	Description
getcwd()	Get the current working folder path.
chdir()	Move on or change in specific folder.
chroot()	Move on the root or main directory or folder.
opendir()	Open a specific folder
closedir()	Close a folder.
readdir()	Read a file or folder name from the specified folder or directory.
rewinddir()	Reset the folder handler to the beginning of the folder.
scandir()	Returns an indexed array containing the name of the files and folder or directory in the specific folder/directory.

getcwd() function is use to in PHP Get the current working directory:

Syntax: *getcwd()*

Example:

```
<?php
    echo "My working current folder/directory is : " . getcwd();
 ?>
```

chroot() function changes the root directory of the current process to directory, and changes the current working directory to "/". This function requires root privileges, and is only available to GNU and BSD systems, and only when using the CLI, CGI or Embed SAPI. This function is not implemented on Windows platforms.

Syntax: *chroot(directory)*

Example

```
<?php
   // Change root directory
   chroot("/path/to/chroot/");

    // Get current directory
   echo getcwd();
?>
```

Opendir() Open specific directory handler. To iterate through the

Syntax: *$folderhandler = opendir(path)*

Example:

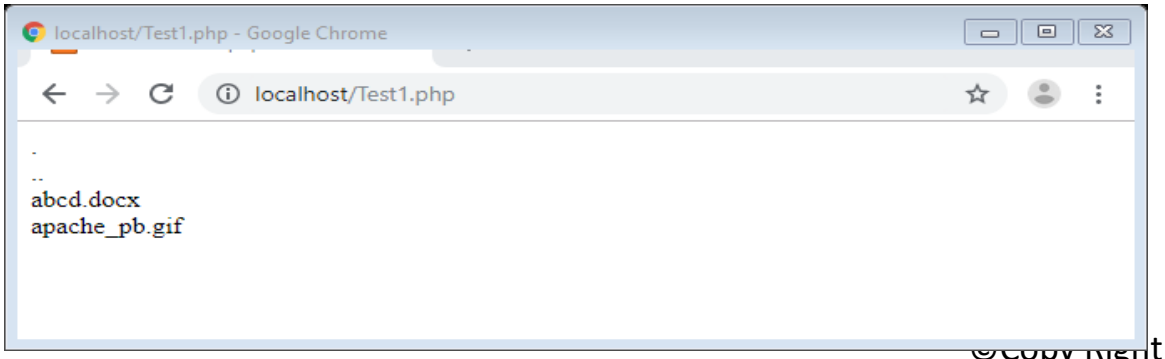
```
<?php
    $Dir="/xampp/htdocs";   // set specific path
    $DirOpen =opendir($Dir);  //   open specific folder

   $CurrentFile=readdir($DirOpen);  // read first . Directory
   echo  $CurrentFile . "<br>";

   $CurrentFile=readdir($DirOpen);  // read second .. Directory
   echo  $CurrentFile . "<br>";

   $CurrentFile=readdir($DirOpen);  // read first file or folder
   echo  $CurrentFile . "<br>";

  $CurrentFile=readdir($DirOpen); // read second file or folder
  echo  $CurrentFile . "<br>";
  closedir($DirOpen);   // close  current directory
?>
```

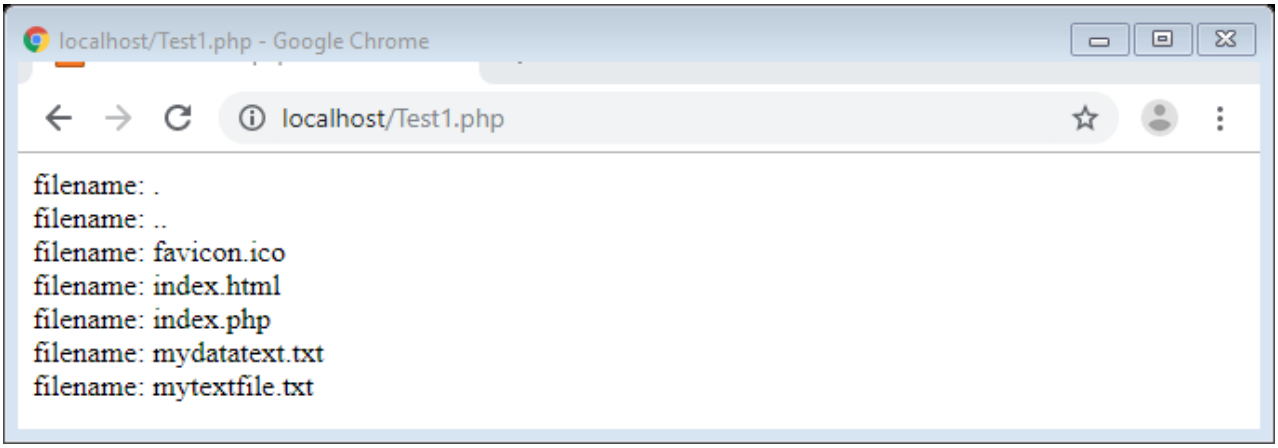


The ***readdir()*** function in PHP is an inbuilt function which is used to return the name of the next entry in a directory. The method returns the filenames in the order as they are stored in the file name system. [d***ir]*** this parameter , Specifies the directory handle resource previously opened with opendir(). If this parameter is not specified, the last link opened by opendir() is assumed

Syntax : *$filenalevariable = readdir(dir)*

Example

```
<?php
  $dir = opendir("mywebfolder");
  while (($file = readdir($dir)) !== false) {
    echo "filename: " . $file . "<br />";
  }
  closedir($dir);
?>
```



Scandir() function in PHP is an inbuilt function which is used to return an array of files and directories of the specified directory. The ***scandir()*** function lists the files and directories which are present inside a specified path.

Syntax: *scandir(*directory, order, context*)*

Directory: specific path name

Order : read files and folder ascending or descending order

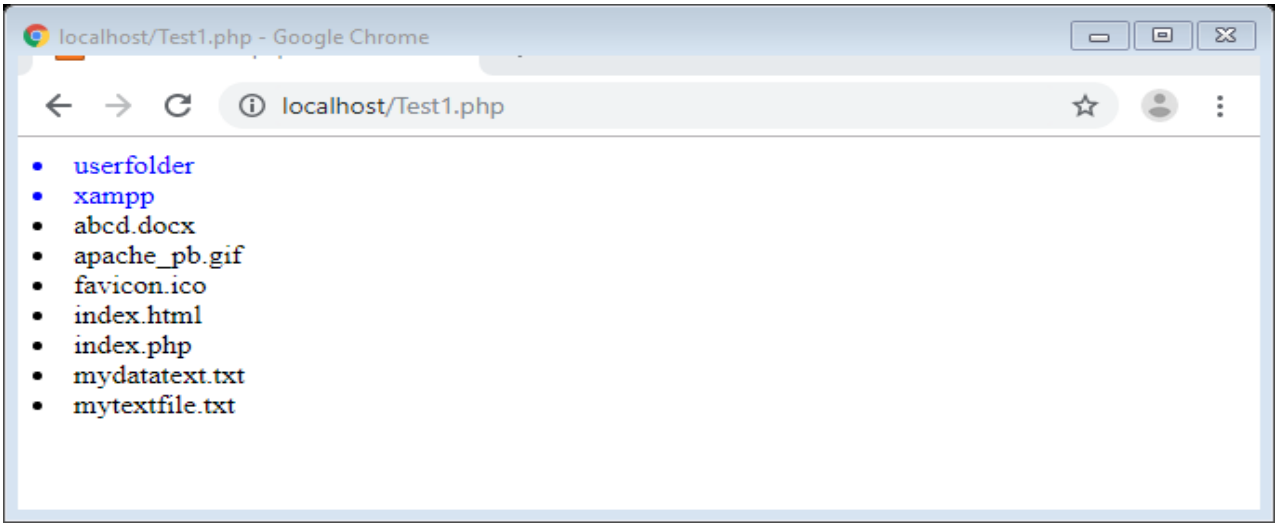
0 for ascending 1 for descending order.

context : Context is a set of options that can modify the behavior of a stream.

Example

```
<?php
$dir = 'Mywebfolder';
$directories = array();
```

```
$files_list  = array();
$files = scandir($dir);
foreach($files as $file)
{
  if(($file != '.') && ($file != '..'))
  {
      if(is_dir($dir.'/'.$file))
      {
        $directories[]  = $file;
      }else
     {
       $files_list[]    = $file;
     }
  }
}
foreach($directories as $directory)
{
  echo '<li style="color:blue">'.$directory.'</li>';
}
foreach($files_list as $file_list)
{
  echo '<li>'.$file_list.'</li>';
}
?>
```



Exercise

Theory Questions

1. What is the difference between the binary and a text file?
2. What are the different end-of-line markers for windows, Macintosh and UNIX/Linux.
3. What are the three typical permissions for files and Directories?
4. What function returns an indexed array containing the names of files and Directories in the specified directory?
5. What function is used to create and remove a directory with syntax and example.
6. What type of form input element is used to choose the file to upload?
7. What is the name of auto-Global array that contains the uploaded file information?
8. What function is used to send the contents of a file to the client Web browser?
9. What is a file stream?
10. How do you move a file in PHP?

Practical Questions

1. Write a C program to store the following record of 10 employees in a file.
    - Emp_code (it should be in the format 01, 02, 03 ……. and so on).
    - Name
    - Designation
    - Salary

a) Search and display for employee with Emp_code=03.

b) Delete Emp_code=06 from the records.

c) Modify the data of employee having Emp_code=01 and increase his salary by Rs.2000.

d) Display the record of Emp_code=01.

e) Display all the data of the file.

Objective MCQ's

1. Which of the following escape sequences is used on Macintosh platform?
    a) \n
    b) \r
    c) \n\r
    d) \r\n

2. Which of the following functions sorts directory entries?
    a) scandir()
    b) readdir()
    c) opendir()
    d) sortdir()

3. Which of the following function reads the contents of a file into string?
   a) file()
   b) fread()
   c) readfile()
   d) gets()

4. Which of the following function return a value of TRUE when a file pointer reaches the end of a file?
   a) is_end()
   b) end()
   c) eof()
   d) feof()

5. Which of the following statements creates a directory named “students” at the same level as the current directory.
   a) mkdir(“/students”);
   b) mkdir(“students”);
   c) mkdir(“../students”);
   d) mkdir(“/students/”);
6. PHP provides the _____ function for changing the permissions of a file/folder within PHP.
   a) decoct()
   b) rmdir()
   c) chmod()
   d) mkdir()
7. Get the current working directory, which function will be use?
   a) readdir()
   b) scandir()
   c) chroot()
   d) getcwd()
8. Return the file type , which function use?
   a) filetime()
   b) filesize()
   c) filetype()
   d) filename()
9. PHP uses the _____ function to return header information to the client Web browser.
   a) sethead()
   b) webheader()
   c) headerinformatio()
   d) header()
10. The PHP _____ function reads a file from disk and sends it directly to the Web browser.
    a) readfile()
    b) fgets()
    c) read_file()
    d) read()