PHP - Read File

There are a number of options in PHP for reading data from a file that has been opened with the fopen() function. The following built-in functions in PHP’s library can help us perform the read operation –

- **fgets()** – gets a line from the file pointer.
- **fgetc()** – returns a string with a single character from the file pointer.
- **fread()** – reads a specified number of bytes from the file pointer.
- **fscanf()** – reads data from the file and parses it as per the specified format.

The fgets() Function

The **fgets()** function can return a line from an open file. This function stops returning on a new line at a specified length or EOF, whichever comes first and returns **false** on failure.

```
fgets(resource $stream, ?int $length = null): string|false
```

Here, the **$stream** parameter is a file pointer or handle to the file opened with the fopen() function with read or read/write mode, and **$length** is an optional parameter specifying the number of bytes to be read.

The read operation ends when "length-1" bytes are read or a newline is encountered, whichever is first.

Example

The following code reads the first available line from the "hello.txt" file –

```
<?php
   $file = fopen("hello.txt", "r");
   $str = fgets($file);
   echo $str;
   fclose($file);
?>
```

It will produce the following **output** –

```
Hello World
```

Example

You can put the fgets() function in a loop to read the file until the end of file is reached.

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

It will produce the following **output** −

```
Hello World
TutorialsPoint
PHP Tutorials
```

Here, we have used the **feof()** function which returns true if the file pointer is at EOF; otherwise returns false.

The fgetc() Function

The fgetc() function returns a single character read from the current position of the file handle. It returns **false** when EOF is encountered.

```
fgetc(resource $stream): string|false
```

Here, the **$stream** parameter is a file pointer or handle to the file opened with the fopen() function with read or read/write mode.

Example

The following code displays the first character read from the "hello.txt" file −

```
<?php
   $file = fopen("hello.txt", "r");
   $str = fgets($file);
   echo $str;
   fclose($file);
?>
```

It will produce the following **output** −

```
H
```

Example

You can also put the **fgetc()** function inside a loop to read the file character by character until it reaches EOF.

```
<?php
   $file = fopen("hello.txt", "r");
   while(! feof($file)) {
      $char = fgetc($file);
      if ($char == "\n")
      echo "<br>";
      echo $char;
   }
   fclose($file);
?>
```

It will produce the following **output** −

```
Hello World
TutorialsPoint
PHP Tutorials
```

The fread() Function

The fread() function in PHP is a binary-safe function for reading data from a file. While the fgets() function reads only from a text file, the fread() function can read a file in binary mode.

```
fread(resource $stream, int $length): string|false
```

Here, the **$stream** parameter is a file pointer or handle to the file opened with the fopen() function with binary read or read/write mode (**rb** or **rb+**). The **$length** parameter specifies number of bytes to be read.

If the **$length** parameter is not given, PHP tries to read the entire file until EOF is reached, subject to the chunk size specified.

Example

The following code reads a text file –

```php
<?php
   $name = "hello.txt";
   $file = fopen($name, "r");
   $data = fread($file, filesize($name));
   echo $data;
   fclose($file);
?>
```

It will produce the following **output** –

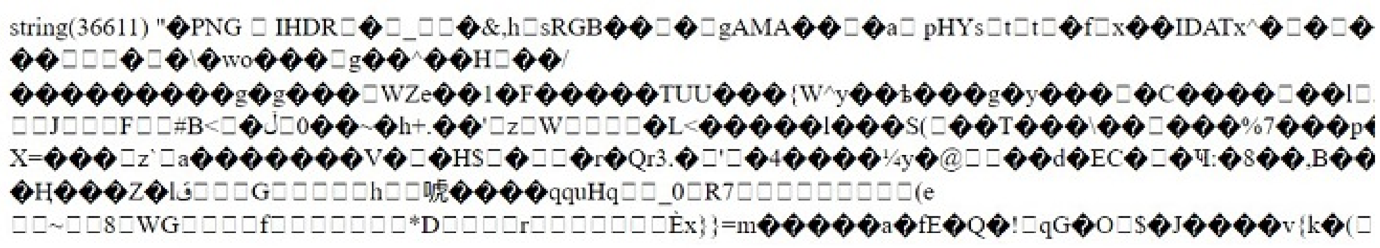
```
Hello World TutorialsPoint PHP Tutorials
```

Example

You can also read a non-ASCII file such as an image file opened in **rb** mode.

```php
<?php
   $name = "welcome.png";
   $file = fopen($name, "rb");
   $data = fread($file, filesize($name));
   var_dump($data);
   fclose($file);
?>
```

The browser displays the "var_dump" information as the following –



```
string(36611) "�PNG □ IHDR□�□_□□�&,h□sRGB��□�□gAMA��□�a□ pHYs□t□t□�f□x��IDATx^�□��□�
��□□□□�□�\�wo����□g��^��H□��/
����������g�g����□WZe��1�F������TUU���{W^y��ƀ���g�y���□�C�����□��l□.
□□J□□□F□□#B<□�J□0��~�h+.��'□z□W□□□□□�L<������l����S(□���T���\��□����%7���p�
X=���□z`□a���������V�□�HS□�□□□�r�Qr3.�□'□�4�����¼y�@□□□��d�EC�□�Ч:�8��.B��
�Ḩ���Z�lڨ□□□G□□□□□□h□□唬����qquHq□□_0□R7□□□□□□□□□□□(e
□□~□□8□WG□□□□f□□□□□□□□*D□□□□r□□□□□□□□Èx}}=m������a�fE�Q�!□qG�O□$�J�����v{k�(□
```

The fscanf() Function

The fscanf() function in PHP reads the input from a file stream and parses it according to the specified format, thereby converts it into the variables of respective types. Each call to the function reads one line from the file.

```
fscanf(resource $stream, string $format, mixed &...$vars): array|int|false|null
```

Here, the **$stream** parameter is the handle to the file opened with the fopen() function and in read mode. And, **$format** is a string containing one or more of the following formatting specifiers −

- **%%** − Returns a percent
- **%b** − Binary number
- **%c** − The character according to the ASCII value
- **%f** − Floating-point number
- **%F** − Floating-point number
- **%o** − Octal number
- **%s** − String
- **%d** − Signed decimal number
- **%e** − Scientific notation
- **%u** − Unsigned decimal number
- **%x** − Hexadecimal number for lowercase letters
- **%X** − Hexadecimal number for uppercase letters

$vars is an optional parameter that specifies variables by reference which will contain the parsed values.

Assuming that the "employees.txt" file is available in the same directory in which the PHP script given below is present. Each line in the text file has **name, email, post** and **salary** of each employee, separated by tab character.

Example

The following PHP script reads the file using the format specifiers in fscanf() function −

```
<?php
   $fp = fopen("employees.txt", "r");
   while ($employee_info = fscanf($fp, "%s\t%s\t%s\t%d\n")) {
      list ($name, $email, $post, $salary) = $employee_info;
      echo "<b>Name</b>: $name <b>Email</b>:
      $email <b>Salary</b>: Rs. $salary <br>";
   }
   fclose($fp);
?>
```

It will produce the following **output** −

```
Name: Ravishankar Email: ravi@gmail.com Salary: Rs. 40000
Name: Kavita Email: kavita@hotmail.com Salary: Rs. 25000
Name: Nandkumar Email: nandu@example.com Salary: Rs. 30000
```