PHP - Write File

PHP's built-in function library provides two functions to perform write operations on a file stream. These functions are **fwrite()** and **fputs()**.

To be able to write data in a file, it must be opened in write mode (w), append mode (a), read/write mode (r+ or w+) or binary write/append mode (rb+, wb+ or wa).

The fputs() Function

The fputs() function writes a string into the file opened in a writable mode.

```
fputs(resource $stream, string $string, int $length)
```

Here, the **$stream** parameter is a handle to a file opened in a writable mode. The **$string** parameter is the data to be written, and **$length** is an optional parameter that specifies the maximum number of bytes to be written.

The fputs() function returns the number of bytes written, or **false** if the function is unsuccessful.

Example

The following code opens a new file, writes a string in it, and returns the number of bytes written.

</> Open Compiler

```
<?php
   $fp = fopen("hello.txt", "w");
   $bytes = fputs($fp, "Hello World\n");
   echo "bytes written: $bytes";
   fclose($fp);
?>
```

It will produce the following **output** –

```
bytes written: 12
```

Example

If you need to add text in an earlier existing file, it must be opened in append mode (**a**). Let us add one more string in the same file in previous example.

```
<?php
   $fp = fopen("hello.txt", "a");
   $bytes = fputs($fp, "Hello PHP");
   echo "bytes written: $bytes";
   fclose($fp);
?>
```

If you open the "hello.txt" file in a text editor, you should see both the lines in it.

Example

In the following PHP script, an already existing file (hello.txt) is read line by line in a loop, and each line is written to another file (new.txt)

It is assumed thar "hello.txt" consists of following text –

```
Hello World
TutorialsPoint
PHP Tutorials
```

Here is the PHP code to create a copy of an existing file –

```
<?php
   $file = fopen("hello.txt", "r");
   $newfile = fopen("new.txt", "w");
   while(! feof($file)) {
      $str = fgets($file);
      fputs($newfile, $str);
   }
   fclose($file);
   fclose($newfile);
?>
```

The newly created "new.txt" file should have exactly the same contents.

The fwrite() Function

The frwrite() function is a counterpart of fread() function. It performs binary-safe write operations.

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

Here, the **$stream** parameter is a resource pointing to the file opened in a writable mode. Data to be written to the file is provided in the **$data** parameter. The optional **$length** parameter may be provided to specify the number of bytes to be written. It should be **int**, writing will stop after length bytes have been written or the end of data is reached, whichever comes first.

The fwrite() function returns the number of bytes written, or **false** on failure along with E_WARNING.

Example

The following program opens a new file, performs write operation and displays the number of bytes written.

```
<?php
   $file = fopen("/PhpProject/sample.txt", "w");
   echo fwrite($file, "Hello Tutorialspoint!!!!!");
   fclose($file);
?>
```

Example

In the example code given below, an existing file "welcome.png" in opened in binary read mode. The fread() function is used to read its bytes in "$data" variable, and in turn written to another file "new.png" −

```
<?php
   $name = "welcome.png";
   $file = fopen($name, "rb");
   $newfile = fopen("new.png", "wb");
   $size = filesize($name);
   $data = fread($file, $size);
   fwrite($newfile, $data, $size);
   fclose($file);
   fclose($newfile);
?>
```

Run the above code. The current directory should now have a copy of the existing "welcome.png" file.