PHP - Delete File

PHP doesn’t have either a **delete** keyword or a **delete()** function. Instead, it provides the **unlink()** function, which when called, deletes a file from the filesystem. It is similar to Unix/C unlink function.

If the delete operation could not be completed, PHP returns false and shows an E_WARNING message.

```
unlink(string $filename, ?resource $context = null): bool
```

The mandatory string parameter to unlink() function is a string that refers to the file to be deleted.

Example

The following code demonstrates a simple use of the unlink() function −

```
<?php
   $file = "my_file.txt";

   if (unlink($file)) {
      echo "The file was deleted successfully.";
   } else {
      echo "The file could not be deleted.";
   }
?>
```

Deleting the Symlink to a File

The unlink() function can also delete a symlink to a file. However, deleting a symlink doesn’t delete the original file. A symlink is a shortcut to an existing file.

In Windows, open a command prompt with administrator privilege and use the **mlink** command with **/h** switch to create a symlink to a file. (**/j** switch is used for symlink to a folder)

```
mklink /h hellolink.lnk hello.txt
Hardlink created for hellolink.lnk <<===>> hello.txt
```

In Ubuntu Linux, to create a symbolic link to a file, you would use the following command –

```
ln -s /path/to/original_file /path/to/symlink
```

To create a symbolic link to a directory, you would use the following command –

```
ln -s /path/to/original_directory /path/to/symlink
```

In PHP, there is also a symlink() function for the purpose.

```
symlink(string $target, string $link): bool
```

Example

Create a symlink with the following code –

```
<?php
   $target = 'hello.txt';
   $link = 'hellolink.lnk';
   symlink($target, $link);

   echo readlink($link);
?>
```

Now delete the symlink created above –

```
unlink("hellolink.lnk");
```

If you check the current working directory, the symlink will be deleted, leaving the original file intact.

How to Rename a File in PHP

You can change the name of an existing file with the help of respective command from the console of an operating system. For example, the "**mv command** in Linux terminal or the "**rename command**" in Windows command prompt helps you to change the name of a file.

However, to rename a file programmatically, PHP’s built-in library includes a **rename()** function.

Here is the **syntax** of the rename() function −

```
rename(string $from, string $to, ?resource $context = null): bool
```

Both **$from** and **$to** strings are the names of files, existing and new respectively. The rename() function attempts to rename **$from** to **$to**, moving it between directories if necessary.

If you are renaming a **file** and **$to** already exists, then it will be overwritten. If you are renaming a **directory** and **$to** exists, then this function will emit a **warning**.

To change the name of "hello.txt" to "test.txt" −

```
<?php
   rename("hello.txt", "test.txt");
?>
```

You can also employ a little indirect approach for renaming a file. Make a copy of an existing file and delete the original one. This also renames "hello.txt" to "test.txt" −

```
copy("hello.txt", "test.txt");
unlink("hello.txt");
```