PHP – File Permissions

The concept of permissions is at the core of Unix/Linux file system. The permissions determine who can access a file and how one can access a file. File permissions in Linux are manipulated by the **chmod command**, which can be run inside the Linux terminal. PHP provides the **chmod() function** with which you can handle file permissions programmatically.

PHP's chmod() function is effective only when you are working on a Linux OS. It doesn't work on Windows, as Windows OS has a different mechanism of controlling file permissions.

To view the permissions enabled on a file, obtain the list of files using the "**ls -l**" command (long listing)

```
mvl@GNVBGL3:~$ ls -l

-rwxr-xr-x 1 mvl mvl 16376 May  5 21:52 a.out
-rw-r--r-- 1 mvl mvl    83 May  5 21:52 hello.cpp
-rwxr-xr-x 1 mvl mvl    43 Oct 11 14:50 hello.php
-rwxr-xr-x 1 mvl mvl    43 May  8 10:01 hello.py
drwxr-xr-x 5 mvl mvl  4096 Apr 20 21:52 myenv
```

The first column contains permission flags of each file. Third and fourth columns indicate the owner and group of each file, followed by size, date and time, and the file name.

The permissions string has ten characters, their meaning is described as follows –

Position	Meaning
1	"d" if a directory, "-" if a normal file
2, 3, 4	read, write, execute permission for user (owner) of file
5, 6, 7	read, write, execute permission for group
8, 9, 10	read, write, execute permission for other (world)

The characters in the permission string have following meaning –

Value	Meaning
-	Flag is not set.

r	File is readable.
w	File is writable. For directories, files may be created or removed.
x	File is executable. For directories, files may be listed.

If you consider the first entry in the above list –

```
-rwxr-xr-x 1 mvl mvl 16376 May  5 21:52 a.out
```

The "a.out" file is owned by the user "mvl" and group "mvl". It is a normal file with "read/write/execute" permissions for the owner, and "read/ execute" permissions for the group as well as others.

The binary and octal representation of permission flags can be understood with the following table –

Octal Digit	Binary Representation (rwx)	Permission
0	000	none
1	001	execute only
2	010	write only
3	011	write and execute
4	100	read only
5	101	read and execute
6	110	read and write
7	111	read, write, and execute (full permissions)

The chmod() Function

The chmod() function can change permissions of a specified file. It returns **true** on success, otherwise **false** on failure.

```
chmod(string $filename, int $permissions): bool
```

The chmod() function attempts to change the mode of the specified file (**$filename**) to that given in permissions.

The second parameter **$permissions** is an octal number with four octal digits. The first digit is always zero, second specifies permissions for the owner, third for the owner's user group and fourth for everybody else. Each digit is the sum of values for each type of permission.

1	Execute Permission
2	Write Permission
4	Read Permission

The default value of **$permissions** parameters is **0777**, which means the directory is created with execute, write and read permissions enabled.

Example

Take a look at the following example –

```
<?php

   // Read and write for owner, nothing for everybody else
   chmod("/PhpProject/sample.txt", 0600);

   // Read and write for owner, read for everybody else
   chmod("/PhpProject/sample.txt", 0644);

   // Everything for owner, read and execute for everybody else
   chmod("/PhpProject/sample.txt", 0755);

   // Everything for owner, read for owner's group
   chmod("/PhpProject/sample.txt", 0740);
?>
```

The chown() Function

The chown() function attempts to change the owner of the file filename to a new user. Note that only the superuser may change the owner of a file.

```
chown(string $filename, string|int $user): bool
```

Example

Take a look at the following example –

```php
<?php

   // File name and username to use
   $file_name= "index.php";
   $path = "/PhpProject/backup: " . $file_name ;
   $user_name = "root";

   // Set the user
   chown($path, $user_name);

   // Check the result
   $stat = stat($path);
   print_r(posix_getpwuid(fileowner($path)));
?>
```


The chgrp() Function

The chgrp() function attempts to change the group of the file filename to group.

```
chgrp(string $filename, string|int $group): bool
```

Only a **superuser** may change the group of a file arbitrarily; other users may change the group of a file to any group of which that user is a member.

Example

Take a look at the following example –

```php
<?php
   $filename = "/PhpProject/sample.txt";
   $format = "%s's Group ID @ %s: %d\n";
   printf($format, $filename, date('r'), filegroup($filename));
   chgrp($filename, "admin");
   clearstatcache();    // do not cache filegroup() results
   printf($format, $filename, date('r'), filegroup($filename));
?>
```

It will produce the following **output** –

```
/PhpProject/sample.txt's Group ID @ Fri, 13 Oct 2023 07:42:21 +0200: 0
/PhpProject/sample.txt's Group ID @ Fri, 13 Oct 2023 07:42:21 +0200: 0
```