# Lab - 3 - File Permissions

### Objective

1. Verify read, write and execute permissions for file
2. Verify read, write and execute permissions for directories
3. Change `umask` and verify its effect of default permission
4. Calculate default permissions for a given `umask`.
5. Determine an appropriate `umask` for a desired level of access control.

### Reference

1. Chapter 3

### Submission

Complete the online quiz on file permission.

### Background

The file names used in this exercise such as `vanda`, `cattleya`, `laelia`, `oncidium`, `lycaste` and `dendrobium` are orchid genera.
The emphasis is on minimum permission required to perform a task. Given full read, write and execute permission a user can perform any tasks, the objective is to give the *minimum* permission.

### Requirements

A users account on Ubuntu.

### Procedure

Run all commands in user mode. Do not use the root account, unless explicitly specified in the exercise. Using a root account without a real need will limit your understanding of the concepts. If you get a permission denial error try to resolve the issue, instead to running the command with root privileges.

This lab has short exercises that grant and revoke permissions to your own files and directories. In a production system another user, such as an administrator will perform these tasks on files and directories that are used by other users and groups. You are assuming a role of an administrator.

**Exercise 1. Determine the `umask` for the account.** Type in `umask`, write the 4 digit octal number. Write the number in the space below. Discard the leading octal digit, you now have three octal digits. This will be the subtrahend. Subtract this number from 777, the minuend. You will be using octal arithmetic. Convert the difference from octal to binary. Put them into groups of three. Associate each group into `rwx` `rwx` `rwx`. Remove the execute permission. This will be your default file permission.

**Exercise 2. Confirm the default file permission using `touch`.** In your home directory create a file called `phal` using `touch phal`. Verify the file's permission using ls -l phal. Does the permission match the expected permission from the arithmetic you performed in the previous exercise?

**Exercise 3. Perform a write operation and confirm the files permission.** You can add some sample data to this file. You may use `vim` or redirect a command to `phal`.
For example, `date > phal`
Confirm that you have the read permission by typing `cat phal`, it should display the files contents.

**Exercise 4. Remove the write permission for the user** Remove the write permission from the file `phal`. Use the `chmod` command. Such as `chmod 464 phal` in absolute mode or `chmod u-w phal` in symbolic mode. Verify the change in permission using `ls -l phal`.
Add data to `phal` using a double redirection operator. `date >> phal`
Write down the error in the space provided. Verify that the earlier data is still readable, use `cat phal` to verify.

Put the write permission back using `chmod 664 phal` in absolute mode or `chmod u+w phal`.

Confirm that the write permission is restored and that you can now write to the file.
*Aside:* `vim` can override the write permission as we will see later. You can prevent an accidental overwrite in `vim` using `noclobber`.

**Exercise 5. Remove the read permission, without removing the write permission.** Confirm that you have restored the files permission to `rw-rw-r--`. Remove the write permission using chmod in absolute mode `chmod 264 phal` or use symbolic mode `chmod u-r phal`.

You will be able to write to the file but not read from it, even though you are the owner of the file. Type in `cat phal`. Write the error message in the space below.

You should be able to still write to the file using `date >> phal`.
Think of a use of such a file permission, not readable but only writable.

Restore the read file permission using `chmod 664 phal` in absolute mode or `chmod u+r phal`. Use `ls -l phal` to confirm. Now use `cat phal` to verify that the last input has been stored in the file.

**Exercise 6. Directory permission - Read, no write** The default permission for a directory with `umask` as `0002` is `rwxrwxr-x`. Create a directory called `orchid`. Verify the default file permission. Create a file in the `orchid` directory, call it `oncidium`. With the default permission a file will be created with no errors.
Remove the write permission leave the read permission for `orchid`, use `chmod`. Now attempt to create a new file within `orchid` called it `laelia`. With the write permission removed, `bash` will give an error. Write the error message.

Put the write permission back, and remove the read permission. Use `chmod u+w,u-r orchid`. Create a file call `cattleya` in the `orchid` directory, use `touch`. bash will create a file inthe directory because you have write permission but will not let you view the contents of the directory because you do not have read permission. You

will be able to change to the directory using `cd`; you have execute permission. `tree orchid` will also not allow you to view the contents of the directory.

**Exercise 7. Use of a directory with only write and execute permissions.** Think of two real life examples of having a directory with read permission removed. Write your answer in the space below.

**Exercise 8. Minimum permission to delete a file from a directory.** Create a directory, call it `orchid`, create a file within this directory, call it `vanda`, use `touch vanda`. Confirm the file's existence using `tree orchid`. Change the permission of `orchid` to 355, i.e. remove the read permission, but leave the write and execute permissions for the owner. Use `tree orchid`, you cannot see the contents of the directory, the read permission has been removed.
Delete the file `vanda` from the directory using `rm orchid/vanda`. Put the read permission back using `chmod 755 orchid`, confirm that the file `vanda` has been removed.
Repeat this exercise by removing either the write or execute permission, and observe the error messages.

**Exercise 9. Copying files to a directory and restricting access.** To copy a file from one directory to another the minimum permission required are execute permission to the source directory, write and execute permission in the target directory. With this *minimum* permission in the source directory, a user must know the filename in the source directory.

Create two directories `PrjMgr` for project manager and `archive/Mgt/Q01` for archiving the files. `PrjMgr` will be source directory and `archive/Mgt/Q01` will be the target directory. Create a file in the directory `PrjMgr`, call it `Plan.Q01`, use touch.
Remove read and write permission from `PrjMgr` directory, and remove read permission from the target directory `archive/Mgt/Q01`.
Copy the file Plan.Q01 from the source to target. To verify the operation you will need to restore the read permission to the target directory. Did the operating succeed?
Repeat the exercise by removing the execute permission from the source directory. You may copy this file to the current directory, since you are testing the permission of the source directory.
Are you able to `cd` to the directory without the execute permission?
Restore the permission of the source directory to `165`. Change the target directory's permission to read and write only.
Are you able to perform the copy operation?
Write your observations in the space provided.

**Exercise 10. Minimum Permissions.** Circle the minimum permission requried to complete the actions.

1. To copy a file a user requires:
   (a) for source directory: `r w x`
   (b) for target directory: `r w x`
   (c) for the file: `r w x`
2. To move a file a user requires:
   (a) for source directory: `r w x`
   (b) for target directory: `r w x`
   (c) for the file: `r w x`
3. To delete a file a user requires:
   (a) for directory: `r w x`
   (b) for the file: `r w x`