# Lab No. 02

# Working with Navigation, and File & Directory Handing Commands

## Objective:

This lab will introduce the Directory and File related commands to you. We will start with the some basic but important commands used to navigate through the Linux file system. Then we will discuss Directory and File related commands. Finally, we will introduce the I/O redirection in Linux

## Activity Outcomes:

On completion of this lab students will be able to:

- Navigation through Linux file system using CLI
- Working with directories in Linux using CLI
- Handling Files in Linux using CLI
- Using I/O redirection in Linux.

## Instructor Notes

As pre-lab activity, read Chapter 1 to 6 from the book "The Linux Command Line", William E. Shotts, Jr.

## 1) Useful Concepts

Linux organizes its files in a hierarchical directory structure. The first directory in the file system is called the root directory. The root directory contains files and subdirectories, which contain more files and subdirectories and so on and so on. If we map out the files and directories in Linux, it would look like an upside-down tree. At the top is the root directory, which is represented by a single slash (/). Below that is a set of common directories in the Linux system, such as bin, dev, home, lib , and tmp , to name a few. Each of those directories, as well as directories added to the root, can contain subdirectories.

## Navigation

The first thing we need to learn is how to navigate the file system on our Linux system. In this section we will introduce the commands used for navigation in Linux system.

## Print Working Directory

The directory we are standing in is called the current working directory. To display the current working directory, we use the pwd (print working directory) command. When we first log in to our system our current working directory is set to our home directory. Suppose, a user is created with name ***me*** on machine Ubuntu; we display its current working directory as given below:

```
[me@ubuntu ~] $ pwd
/home/me
```

## Listing The Contents Of A Directory

To list the files and directories in the current working directory, we use the **ls** command. Suppose, a user ***me*** is in its home directory; to display the contents of current working directory can be displayed as follows:

```
[me@ubuntu ~] $ ls
Desktop Documents Music Pictures Pulic Templates Videso
```

Besides the current working directory, we can specify the directory to list, like so:

```
[me@ubuntu ~] $ ls /usr
bin  games  kerberos  libexec  sbin  src
etc  include  lib  local  share  tmp
```

Or even specify multiple directories. In this example we will list both the user's home directory (symbolized by the "~" character) and the /usr directory:

```
[me@ubuntu ~] $ ls ~ /usr

/home/me:
Desktop Documents Music Pictures Pulic Templates Videso
/usr:
bin  games  kerberos  libexec  sbin  src
etc  include  lib  local  share  tmp
```

The following options can also be used with ls command

| Options | Long-options | Description |
|---|---|---|
| -a | - - all | List all files, even those with names |
| -d | - - directory | Ordinarily, if a directory is specified, ls will list the contents of the directory, not the directory itself. Use this option in conjunction with the -l option to see details about the directory rather than its contents. |
| -h | - - human-readable | In long format listings, display file sizes in human readable format rather than in bytes. |
| -r | - - reeverse | Display the results in reverse order. Normally,ls displays its results in ascending alphabetical order. |
| -S | - | Sort results by file size. |
| -t | | Sort by modification time |
| -l | | Display results in long format. |

## Changing the Current Working Directory

To change your working directory, we use the **cd** command. To do this, type cd followed by the pathname of the desired working directory. A pathname is the route we take along the branches of the tree to get to the directory we want. Pathnames can be specified in one of two different ways; as **absolute pathnames** or as **relative pathnames**. An absolute pathname begins with the root directory

and follows the tree branch by branch until the path to the desired directory or file is completed. On the other hand a relative pathname starts from the working directory.
Suppose, a user ***me*** is in its home directory and we want to go into the Desktop directory, then it can be done as follows:

```
[me@ubuntu ~] $ cd /home/me/Desktop (absolute path)
or
[me@ubuntu ~] $ cd Desktop (relative path)
```

The ".." operator is used to go to the parent directory of the current working directory. In continuation of the above example, suppose we are in the Desktop directory and we have to go to the Documents directory. To this task, first we will go the parent directory of Desktop (i.e. me, home directory of the user) that contains the Documents directory then we will go into the Documents directory as given below.

```
[me@ubuntu Desktop ] $ cd /home/me/Documents (absolute path)
or
[me@ubuntu Desktop ] $ cd ../Documents (relative path)
```

## Working With Directories

In this Section, we introduce the most commonly used commands related to Directories.

## Creating a Directory

In Linux, ***mkdir*** command is used to create a directory. We pass the directory name as the argument to the mkdir command. Suppose, the user me is in its home directory and we want to create a new directory named **mydir** in the Desktop directory. To do this, first we will change the current directory to Desktop and then we will create the new directory. It is shown below:

```
[me@ubuntu ~] $ cd /home/me/Desktop
[me@ubuntu Desktop ] $ mkdir mydir
```

Multiple directories can also be created using single mkdir command as given below:

```
[me@ubuntu Desktop ] $ mkdir mydir mydir1 mydir2
```

## Copying Files and Directories

**cp** command is used to copy files and directories. The syntax to use cp command is givenbelow:

```
cp item1 item2
```

Here, item1 and item2 may be files or directories. Similarly, multiple files can also be copiedusing

```
cp item .... directory
```

single cp command.

The common options that can be used with cp commands are:

| Option | Long Option | Explanation |
|---|---|---|

| | | |
|---|---|---|
| -a | --archive | Copy the files and directories and all of their attributes |
| -i | --interactive | Before overwriting an existing file, prompt the user for confirmation |
| -r | --recursive | Recursively copy directories and their contents |
| -u | -update | When copying files from one directory to another, only copy files that either don't exist, or are newer |

## Moving and Renaming Files and Directories

**mv** command is used to move files and directories. This command can also be used to rename files and folder. To rename files and directories, we just perform the move operation with old name and new name. As a result, the files or directory is created again with a new name. The syntax to use mv command is given below:

```
mv item1 item2
```

```
mv items ..... directory
```

Similarly, multiple files can be moved to a directory as given below

Common options, used with mv command are:

| Option | Long Option | Explanation |
|---|---|---|
| -i | --interactive | Before overwriting an existing file, prompt the user for confirmation |
| -u | -update | When moving files from one directory to another, only copy files that either don't exist, or are newer |

## Removing and Files and Directories

To remove or delete a files and directories, **rm** command is used. Empty directories can also be deleted using rmdir command but rm can be used for both empty and non-emptydirectories as well as for files. The syntax is given below:

```
rm item1 item2 .....
```

The common options, used with rm command are:

| Option | Long Option | Explanation |
|---|---|---|
| -i | --interactive | Before deleting an existing file, prompt the user for confirmation. |
| -r | --recursive | Recursively delete directories. |

## Working with Files

In this section, we will introduce the file related commands.

## Creating and Empty Text File

In Linux, there are several ways to create an empty text file. Most commonly the **touch** command is used to create a file. We can create a file with name myfile.txt using touch command as given below:

```
touch  myfile.txt
```

```
cat  > myfile.txt
```

Another, way to create a file in Linux is the cat command.

Similarly, a file can be created using some editors. For example, to create a file using gedit editor

```
gedit myfile.txt
```

## Reading the File Contents

**cat** command can also be used to read the contents of a file.

```
cat  myfile.txt
```

```
less   myfile.txt
```

Another option to view the contents of a text file is the use of less command.

Similarly, an editor can also be used to view the contents of a file.

```
gedit   myfile.txt
```

## Appending text files

**cat** command is also used to append a text file. Suppose we want to add some text at theend of

```
cat >>  myfile.txt
```

**myfile.txt**

Now, type the text and enter ctrl+d to copy the text to myfile.txt.

## Combining multiple text files

Using cat command, we can view the contents of multiple files. Suppose, we want to viewthe

```
cat  file1  file2   file3
```

contents of file1, file2 and file3, we can use the cat command as follows:

Similarly, we can redirect the output of multiple files to file instead of screen using cat command. Suppose, in the above example we want to write the contents of file1, file2 and file3 into another file file4 we can do this as shown below:

```
cat file1 file2 file3 > file4
```

## Determining File Type

To determine the type of a file we can use the **file** command. The syntax is given below:

```
file filename
```

## Redirecting I/O

Many of the programs that we have used so far produce output of some kind. This output often consists of two types. First, we have the program's results; that is, the data the program is designed to produce, and second, we have status and error messages that tell us how the program is getting along. If we look at a command like ls, we can see that it displaysits results and its error messages on the screen.
Keeping with the Unix theme of "everything is a file," programs such as ls actually send their results to a special file called standard output (often expressed as stdout) and their status messages to another file called standard error (stderr). By default, both standard output andstandard error are linked to the screen and not saved into a disk file. In addition, many programs take input from a facility called standard input (stdin) which is, by default, attached to the keyboard. I/O redirection allows us to change where output goes and whereinput comes from. Normally, output goes to the screen and input comes from the keyboard, but with I/O redirection, we can change that.

## Redirecting Standard Output

I/O redirection allows us to write the output on another file instead of standard output i.e. screen. To do this, we use the redirection operator i.e. <. For example, we want to write the output of ls command in a text file myfile.txt instead of screen. This can be done as given below:

If we write the output of some other program to myfile.txt using > operator, its previous contents will

```
ls -l > myfile.txt
```

be overwritten. Now, if want to append the file instead of over-writing we can use the << operator.

## Redirecting Standard input

Redirecting input enables us to take input from another file instead of standard input i.e. keyboard. We have already discussed this in previous section while discussing cat command where we used the text file as input instead of keyboard and wrote it to another file.

## Pipelines

The ability of commands to read data from standard input and send to standard output is utilized by a shell feature called pipelines. Using the pipe operator "|" (vertical bar), the standard output of one command can be piped into the standard input of another

```
[me@ubuntu ~] $ ls -l | less
```

## 2) Solved Lab Activities

| *Sr.No* | *Allocated Time* | *Level of Complexity* | *CLO Mapping* |
|---|---|---|---|
| *1* | *15* | *Medium* | *CLO-5* |
| *2* | *15* | *Medium* | *CLO-5* |
| *3* | *15* | *Medium* | *CLO-5* |

## Activity 1:

***In this activity, you are required to perform tasks given below:***

- ***Display your current directory.***
- ***Change to the /etc directory.***
- ***Go to the parent directory of the current directory.***
- ***Go to the root directory.***
- ***List the contents of the root directory.***
- ***List a long listing of the root directory.***
- ***Stay where you are, and list the contents of /etc.***
- ***Stay where you are, and list the contents of /bin and /sbin.***
- ***Stay where you are, and list the contents of ~.***
- ***List all the files (including hidden files) in your home directory.***
- ***List the files in /boot in a human readable format.***

## Solution:

- pwd
- cd /etc
- cd ..
- cd /
- ls
- ls -l
- ls /etc
- ls /bin /sbin
- ls ~
- ls -al ~
- ls -lh /boot

## Activity 2:

***Perform the following tasks using Linux CLI***

- ***Create a directory "mydir1" in Desktop Directory. Inside mydir1 create another directory "mydir2".***
- ***Change your current directory to "mydir2" using absolute path***
- ***Now, change you current directory to Documents using relative path***
- ***Create mydir3 directory in Documents directory and go into it***
- ***Now, change your current directory to mydir2 using relative path***

**Solution:**

- cd /home/ubuntu/Desktop
  mkdir mydir1
  cd mydir1
  mkdir mydir2
- cd /home/ubuntu/Desktop/mydir1/mydir2
- cd ../../../Documents
- mkdir mydir3
  cd mydir3
- cd ../../Desktop/mydir1/mydir2

## Activity 3:

***Considering the directories created in Activity 2, perform the following tasks***

- ***Go to mydir3 and create an empty file myfile using cat command***
- ***Add text the text "Hello World" to myfile***
- ***Append myfile with text "Hello World again"***
- ***View the contents of myfile***

**Solution:**

- cd /home/Documents/mydir3 (suppose the user name is ubuntu)
  cat >myfile
- cat > myfile
  type: Hello World
  type: ctrl + d
- cat >> myfile
  type: Hello World again
  type: ctrl + d
- cat myfile

# 3) Graded Lab Tasks

***Note: The instructor can design graded lab activities according to the level of difficult and complexity of the solved lab activities. The lab tasks assigned by the instructor should be evaluated in the same lab.***

## Task 1:

*Considering the above activities (given solved actvites section), perform the following tasks*

- *move myfile to mydir1*
- *copy myfile to mydir2*
- *copy mydir2 on Desktop*
- *delete mydir1 (get confirmation before deleting)*
- *Rename myfile to mynewfile*

## Task 2:

*This activity is related to I/O redirection*

- *Go to Desktop directory*
- *Write the long-listing of contents of Desktop on an empty file out-put-file*
- *View contents of out-put-file*

## Task 3:

*Considering the lab activities, perform the following tasks*

- *Go to Desktop directory*
- *write the contents of mynewfile to newfile*
- *view the output of both mynewfile and newfile on screen*
- *write the combined output of mynewfile and newfile to a third file out-put-file*

## Task 4:

*Long list all files and directories in your system and write out-put on a text-file.*

# Lab Exercise and Summary

**Summary should cover Introduction, Procedure, Data Analysis and Evaluation**

**LABORATORY SKILLS ASSESSMENT (Psychomotor)**
**Total Marks: 100**

| Criteria (Max Marks) | Level 1 0% ≤ S < 50% | Level 2 50% ≤ S< 70% | Level 3 70% ≤ S< G0% | Level 4 G0%≤ S ≤100% | Score (S) |
|---|---|---|---|---|---|
| **Procedural Awareness (20)** | Selects inappropriate Linux commands, shell scripting techniques, or process management methods. | Selects and applies partially appropriate Linux commands and techniques | Selects and applies considerably appropriate Linux commands and techniques. | Selects and applies completely appropriate Linux commands and techniques | |
| **Practical Implementation (20)** | Makes major critical errors in executing Linux commands, scripting, and system processes. | Makes numerous critical errors in executing commands and process management. | Makes minor non-critical errors in executing Linux commands and system operations. | Executes Linux commands and manages processes correctly with no errors. | |
| **Process Management and Shell Scripting (20)** | Program logic contains major errors with incorrect or contradictory script flow. | Program logic has some errors with occasional contradictions in process execution. | Program logic is mostly correct but may contain occasional redundancy or minor errors. | Program logic is completely correct with no contradictions or redundant processes. | |
| **Syntax Correctness and Results (20)** | Program does not follow proper syntax for Linux commands and shell scripting, leading to incorrect outputs | Program partially follows proper syntax, producing correct results for few inputs. | Program adequately follows proper syntax, producing correct results for most inputs. | Program fully follows proper syntax, producing accurate results for all inputs. | |
| **Use of OS Tools (10)** | Uses OS tools (like terminal, process manager) with limited competence. | Uses OS tools with some competence. | Uses OS tools with considerable competence. | Uses OS tools proficiently with a high degree of competence. | |
| **Safety (10)** | Requires constant reminders to follow system safety procedures (e.g., file permissions, process handling). | Requires some reminders to follow system safety procedures. | Follows system safety procedures with minimal reminders. | Routinely follows system safety procedures. | |
| | | | | **Marks Obtained** | |

## LABORATORY SKILLS ASSESSMENT (Affective)

**Total Marks: 40**

| Criteria (Max. Marks) | Level 1<br>0% ≤ S < 50% | Level 2<br>50% ≤ S < 70% | Level 3<br>70% ≤ S < G0% | Level 4<br>G0% ≤ S ≤ 100% | Score (S) |
|---|---|---|---|---|---|
| **Attitude s Engagement (5)** | Shows little interest in lab activities; does not participate actively. | Participates occasionally but lacks enthusiasm and consistency. | Engages actively in most lab activities with interest. | Highly motivated, participates enthusiastically, and shows a proactive approach | |
| **Responsibility s Punctuality (5)** | Frequently misses deadlines and is often late to lab sessions. | Occasionally late or misses deadlines but tries to catch up. | Submits work on time and attends lab sessions regularly. | Always punctual, meets deadlines, and takes full responsibility for assigned tasks. | |
| **Collaboration s Teamwork (10)** | Rarely collaborates, struggles to work in a team, and does not contribute effectively. | Works with team members occasionally but struggles with communication. | Cooperates well, contributes effectively, and maintains professional interactions. | Actively engages in teamwork, supports peers, and demonstrates excellent collaboration. | |
| **Communicatio n s Presentation Skills (10)** | Struggles to explain concepts, unclear verbal/written communication. | Communicates ideas with some clarity but lacks confidence or coherence. | Presents ideas effectively with minor issues in clarity or structure. | Communicates clearly, confidently, and effectively in all aspects of lab work. | |
| **Report Quality (10)** | Report contains many errors. | Report is somewhat organized with some spelling or grammatical errors. | Report is well organized and cohesive but contains some grammatical errors. | Report is well organized and cohesive and contains no grammatical errors. Presentation seems polished. | |
| | | | | **Marks Obtained** | |