Unit 14 : Useful Command in Linux

Lesson 1 : Directory Commands in Linux

1.1. Learning Objectives

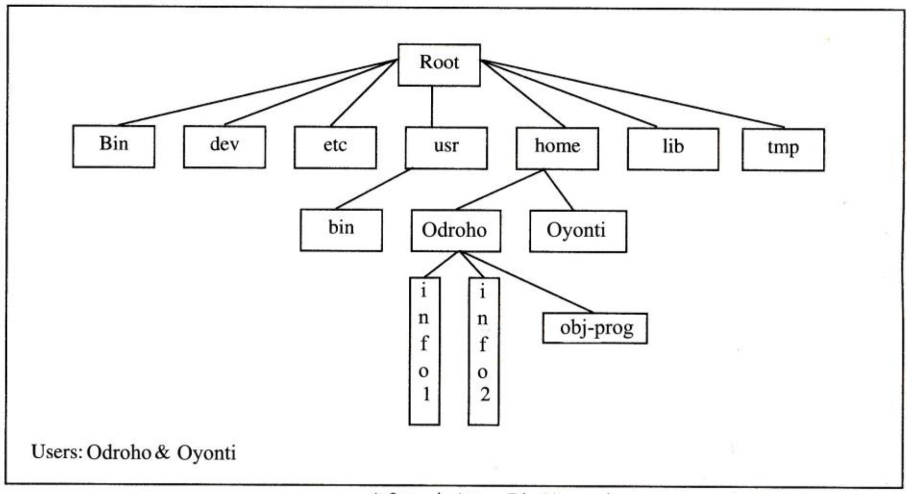
On completion of this lesson you will be able to know:

- ❖ About the important directory commands in Linux

1.2. Linux file hierarchy related to the directory commands

Linux commands are entered after the Linux prompt is displayed. All Linux commands should be entered in lower-case characters.

Linux commands are entered after the Linux prompt is displayed. All Linux commands should be entered in lower-case characters.
For each of the commands that follow, the file hierarchy as shown in the following diagram has been used. Each directory is referred to using its path name beginning from the `'/'` (root) directory. For example, the directory named *home* is referred to as */home* (*/ or* root is its parent directory). The directory named *Odroho* is referred to as `/home/Odroho`, and *home* is the parent directory of the `Odroho` sub-directory



A Sample Linux File Hierarchy

1.3. Identification of the current directory

The **pwd** (print working directory) command is used to display the full path name of the current directory.
Example

```
[Odroho@localhost Odroho] $ pwd /home/ Odroho
/home/ Odroho
[Odroho@localhost Odroho]$
```

Here, `/home/Odroho` is the directory in which the user is currently working.

1.4. Change of current directory

The **cd** (change directory) command changes the current directory to the directory specified.
Assume that Odroho has logged in and has given the following command:
Example

```
      [Odroho@localhost Odroho] $ pwd
      /home/Odroho
      [Odroho @localhost Odroho]$ cd /usr/bin
      [Odroho @localhost bin]$ pwd
      /usr/bin
      [Odroho@localhost bin]$
```

Note that the complete path name has been specified with the cd command. Linux also allows the use of relative path names with commands. Let's look at an example. You can also use the `..` (double dot) option with the **cd** command to move to the parent directory of your current directory. For example, Odroho can enter the following command after logging in, to change to the parent directory of his HOME directory**.**
Example

```
[Odroho@localhost Odroho] $ pwd
/home/Odroho
[Odroho@localhost Odroho] $ cd . .
[Odroho@localhost /home]$ pwd
 /home
[Odroho@localhost /home]$ cd . .
[Odroho@localhost /]$ pwd
/
[Odroho@localhost /]$
```

The two dots refer to the parent directory of the current directory. Note that there has to be a space between **cd** and the two dots, but not between the dots.
The **cd** command without any path name always takes a user back to the HOME directory.

1.5. Creation of a directory

The **`mkdir`** **(**make directory) command is used to create directories.
Example

```
[Odroho@localhost Odroho] $ mkdir program-files
[Odroho@localhost Odroho]$
```

The sub-directory, program-files, is created under the current directory. However, the new directory does not become the current directory. Complete path names can be specified with **mkdir.**
Example

```
[Odroho@localhost Odroho] $ mkdir /tmp/program-files
[Odroho@localhost Odroho] $
```

In the above example, the directory, program-files, is created under the /tmp directory.

1.6. Remove of directory

The **rmdir** (remove directory) command removes the directory specified.

Example

```
[Odroho@localhost Odroho] $ rmdir program-files
[Odroho@localhost Odroho] $
```

Here, the program-files directory is deleted

1.7. List of the contents of directory

The **ls** command is used to display the names of the files and sub-directories in a directory.
Example

```
[Odroho@localhost Odroho]$ ls /home/Odroho
DCSA        X    Cricket    comm       badminton
Desktop    a.out  football  program.cc
[Odroho@localhost Odroho]$
```

In the above example, all the files and directories under the directory named Odroho are listed. If the files and directories under the current directory are to be listed, it is optional to specify the directory name with Is.

You are shown the file names but not the types of files. The -1 option, when used with Is displays a detailed list of files and directories.

Example

```
[Ishrak@localhost Ishrak] $ ls -l
Total   22
--rw-rw-r--      1  Odroho    Odroho        134  nov  23  00:18  DCSA
drwxr-xr-x       5  Odroho    Odroho       1024  nov  22  13: 00  Desktop
-rw-rw-r--       1  Odroho    Odroho          5  nov  23  11:47  X
-rwxrwxr-x       1  Odroho    Odroho      12901  nov  22  16: 13  a.out
drwxrwxr-x       2  Odroho    Odroho       1024  nov  23  00:11  Cricket
drwxrwxr-x       2  Odroho    Odroho       1024  nov  23  00:12  football
drwxrwxr-x       2  Odroho    Odroho       1024  nov  23  00:12  corrun
-rw-rw-r--       1  Odroho    Odroho         10  nov  23  14:45  program.cc
drwxrwxr-x       2  Odroho    Odroho       1024  nov  23  12:26  badminton

[Ishrak@localhost Ishrak]$
```

The output of the **ls – l** command is explained in the following table

Column#	**Description**
1	File type and File Access Permission (FAPs)
2	Number of links
3	File owner
4	Group owner (group name)
5	File size (in bytes)
6,7, and	Day and time of last modification to the file
9	Name of file

1.8. Exercises

1.8.1. Multiple choice questions

a. The **pwd** command

(i) Is used to display the full path name of the current directory
(ii) To change the current directory to the directory specified.
(iii) Is used to create directories.
(iv) Is used to display the names of the files and sub-directories in a directory.

b. To **remove** the directory Linux uses the command

(i) mkdir
(ii) rmdir
(iii) pwd
(iv) cd –r.

c. The fifth column of **ls – l** command implies

(i) Number of links
(ii) File owner
(iii) Group owner
(iv) File size.

1.8.2. Questions for short answers

a. How do you identify the current directory path?
b. Why do you use two dots (**. .**) following the **cd** command?
c. Mention the command to create a directory.
d. What is **'ls – l'** command displayed?

1.8.3. Analytical questions

a. List the command to create and remove directory with examples.
b. Describe columns of the result obtained by **'ls – 1'** command.

Lesson 2 : Common file commands in Linux

2.1. Learning Objectives

On completion of this lesson you will be able to know:

- ❖ About important file commands

2.2. Displaying the contents

The **cat** (concatenate) command displays the contents of the specified file. The **cat** command can be used to vertically concatenate the contents of more than one file.
Example

```
      [Odroho@localhost Odroho]$ cat info1
       A sample file
      [Odroho@localhost Odroho]$ _
```

The command assumes that the file, info1, is in the current directory. Complete path names can also be specified to display a file in another directory.
The **cat** command can also display more than on file as shown in the following command:
Example

```
   [Odroho@localhost Odroho] $ cat infor1 info2
   A sample file
   Another sample file
   [Odroho@localhost Odroho]$ _
```

2.3. Copying files

Copying files

The **cp** (copy) command duplicates the contents of the source file into a target file.
Syntax

```
cp [options] <sourcefile/s> <destination directory/file>
```

Example

```
      [Odroho@localhost Odroho]$ cp infol info2
```

In the above example. the contents of info1 are copied to a new file, info2. If info2 already exists, its contents will be overwritten by the contents of **`info1`**.
Complete path names can be specified with the **`cp`** command to copy files across directories.
You can also copy a directory recursively using **`cp`** command with the –**r** option.
Example

```
[Odroho@localhost Odroho]$ cp -r temp temp1
```

The above command copies the temp directory and all its files and sub-directories to the temp1 directory. If the directory, temp1, exists, all the contents are put inside that directory, otherwise temp1 is created in the current working directory.
The other common options and their functions are given below:

Option	Function
-i	Prompts before overwriting
-l	Links a file instead of copying it
-s	Creates a symbolic link
-v	Verbose-explains what is being done, in details

Options with the cp Command.

2.4. Removing files

The **rm** (remove) command is used to delete files or directories.

Syntax:

rm[options] file/s

Example

```
[Odroho@localhost Odroho]$ rm info1 info2
```

The above command will remove the files, info1 and info2, from your current directory.
If the file to be deleted is not located in the current directory, the complete path name has to be given.

Example

```
[Odroho@localhost Odroho]$ rm /home/Odroho/info1
```

The **–r** option is used with the **rm** command to remove a directory along with its sub-directories. This option is sometimes preferred over the **rmdir** command since in the case of **rmdir,** you can only delete an empty directory.

Example

```
[Odroho@localhost Odroho] rm –r temp1
```

The above command removes the temp1 directory along with all its sub-directories.
The other commonly used options with the **rm** command are shown in the table below.

Option	Function
-i	Prompts before removing
-f	Removes a file by force. It ignores the non-existence of a file, that is, if the file does not exist, the command will not flag an error
-r or –R	Deletes recursively, that is, deletes a directory along with its sub-directories
-v	Verbose-explains what is being done, in detail

Options of the **rm** Command.

2.5. Moving and renaming files

The **mv** (move) command is used to move a file or directory from one location to another or to change the name of a file or directory. Note that moving a file from one location to another is different from copying a file. While moving a file, no new file is created.

Syntax

```
mv [option] source destination
```

Example

```
[Odroho@localhost Odroho] $ mv comm comm1
```

In the above example, the comm directory is renamed to comm1.
A file can be moved to another directory as shown below.

Example

```
[Odroho@localhost Odroho] $ mv info3  /home/Odroho/programs/info3
```

In the above example, the info, data3, is moved from the current directory to the /home/Odroho/programs directory.

Example

```
[Odroho@localhost Odroho] $ mv comm1 temp
```

In the above example, the directory, temp, exists in the current directory; therefore, the common directory is moved from the current directory to the temp directory.

Option	Function
-f	Removes the existing destination by force
-i	Interactive, prompts before overwriting at the destination location
-v	Verbose-explains what is being done, in detail

Options of the mv command.

2.6. Displaying the contents page-wise

The **cat** command is used to display the contents of a file on the screen, however, if the file being displayed is large, then the entire contents, will scroll up the screen. To view the file on screen-full at a time, you can use the more or the less command.
The **more** command is used to display data one screen-full at a time. While viewing a file using the more command, once you have scrolled down, you cannot move up.

Syntax

```
more [options] <filename>
```

Example

```
[Odroho@lsrv01 HOWTO]$ more Xwindow-user-HOWTO
```

The above command will display a page-wise listing of the contents of the file, Xwindow-User-HOWTO.
The **less** command is similar to the more command except that you scroll upwards also while viewing the contents of a file. The **less** command is also a little faster than the **more** command.
To move up and down the screen, you can use the arrow keys. You can also specify

a number to move down the screen by that number of lines. To quit the display, you have to type ‘q’.

Syntax

```
Less[options]<filename>
```

Example

```
[Odroho@lsrv01 HOWTO]$ less XWindow-User-HOWTO
```

The above command will display a page-wise listing of the contents of the file, XWindow-User-HOWTO.

2.7. Wildcard characters

The shell offers the facility to perform an operation on a set of files without having to specify all the names of the files on which the operation is to be performed. This is made possible by the use of certain special characters in the command in place of the actual file names. The shell interprets these special characters as a specific pattern of characters. The shell then compares all the file names under the directory specified in the command to find the file names that match the pattern. The command is executed on the files with name that match the pattern.

Option	Function
*	Matches none or one character or a string of more one character
?	Matches exactly one character
[]	Matches exactly one of a specified set of characters

Windcard Characters.

The * Wildcard

The * wildcard is interpreted as string of none, one, or more characters.

Example

```
[Odroho@Localhost Odroho]$ cat c*
```

Here, c* will match the files whose names start with ‘c’.

The * wildcard can also be repeated in the command line.

Example

```
[Odroho@Localhost Odroho]$ cat chap*.*
```

The above command displays all the files starting with chap and containing any sequence of characters (or no character), followed by a dot, and then followed by any sequence of characters (or no character). Note that. Is not a special character for *.

The ? Wildcard

The ? wildcard matches exactly on occurrence of any character.

Example

```
[Odroho @Localhost Odroho]$ ls *. ?
```

The above command displays all the files having any character(s) before a dot, followed by a single character after the dot.

The [ ] Wildcard

The [ ] wildcard can be used to restrict the characters to be matched.

Example

```
    [Odroho@Localhost Odroho]$ cat a[123]
```

This displays the contents of the files with two character file names starting with a and the next character as either 1, 2 or 3 for example, a1, a2 and a3.

2.8. Exercises

2.8.1. Multiple choice questions

a. The cp command

(i) Duplicates the contents of the source file into a target file
(ii) Is used to delete files of directories
(iii) To change the name of file or directory
(iv) Is used to display the content of file.

b. '– s' option with the cp command implies

(i) Prompts before overwriting
(ii) links a file instead of copying it
(iii) Creates a symbolic link
(iv) explains what is being done.

c. The '*' Wildcard

(i) Is interpreted as string of none, one, or more characters.
(ii) Matches exactly on occurrence of any character.
(iii) Can be used to restrict the characters to be matched.
(iv) Display a page-wise listing of the contents of the file.

2.8.2. Questions for short answers

a. What is the command used to display the content of a file?
b. What does **'-**l' option of '**cp'** (copy) command do?
c. Write the syntax of moving and renaming files with examples.
d. Why is the **'?**' Wildcard used?

2.8.3. Analytical questions

a. List the common file commands. Give example of each.
b. Write down the options of the '**rm'** commands with their functions.
c. List the wildcard characters with their purpose.

Lesson 3 : Other Linux Commands

3.1. Learning Objectives

On completion of this lesson you will be able to know:

- ❖ About some other important commands in linux

3.2. Viewing the System Date and Time

All Linux systems display the current date and time. Users can display the current date and time using the date command.

Viewing the System Date and Time

Example

```
[Odroho@localhost Odroho]$ date
Tue nov 23 21:36:42 BDT 2013
[Odroho@localhost Odroho]$
```

The options of the **date** command can be used to format the date and time before displaying them.

3.3. Manipulating the Screen

Clear
The **clear** command is used to clear the terminal screen.
Linux allows you some measure of screen manipulation with the **tput** commands. Some of them are given below.
tput clear
The command **tput clear** clears the standard output device, the screen, and positions the cursor at the top left corner of the screen.
Example

```
tput clear
```

The **tput cup** command, followed by the screen coordinates, positions the cursor at the specified row and column.
Example

```
tput cup 15 20
```

This will position the cursor at row 15, column 20.
tput smso
This sets the screen to reverse video.
tput rmso

This sets the screen back to normal.
tput blink
This command displays a blinking output. Note that this option may not work on the Telnet session.
tput reset
This resets the screen back to the default settings.

3.4. Identifying the Current users working on the System

who
The **who** command is used to display the names of all the users who are currently logged in.

Example

```
[Odroho@localhost Odroho]$ who
root   tty1 nov   23    13:30
root   tty2 nov   23    15:55
Odroho tty3 nov   23    15:55
nazrul      pts/0 nov   23    12:37 (203.16.70.78)
ruhul pts/1 nov   23    12:16 (203.16.70.130)
```

The output of the '**who'** command also consists of the terminal file name and the date and time the user logged in. In the above example, note that for the first three users the terminal type is **ttyN**, where N is a number from 1 to 12. The tty terminal type is given for users who have logged in from the server. It is possible for multiple users to simultaneously log on from the server. These terminals are known as virtual consoles. Linux allows up to twelve users to log on to the operating system from the server computer. This is possible by using **<Alt>** and any of the function keys. Note that a virtual console is possible only from the server. In the output, the **pts** denotes a remote terminal, which is a machine connected from a machine other than the server.
The '**who am I'** command displays the name of the current user logged in.
Example

```
[Odroho@localhost     Odroho]$ who am i
Localhost.localdomain!Odroho pts/0          nov 23  19:09 (203.16.70.168)
```

3.5. Displaying the Manual Pages

The **man** command displays pages from the Linux reference manual that is installed along with the Linux OS.
For example, to get detailed information about the ls command, you can use the following command:
[Odroho@localhost   Odroho]$ **man ls**

3.6. Checking a Background Process

Checking a Background Process

Sometimes a user will need to check a background process to determine its current status. Is it still running? Has it completed or did it run into problems?
The **ps** (process status) command generates a one-line entry for each of the processes that is currently active.
Example

```
[Odroho@localhost     Odroho]$ ps
```

```
PID     TTY   TIME   CMD
1186    pts/1  0:00    bash
1280    pts/1  0:04    ps
```

The **ps** entries display the process identifier (process number assigned by the kernel), the user's terminal number, the length of time each process has been active - in minutes and seconds, and the name of the command. If the background process is not listed in the output of the **ps** command, it may be already completed.

3.7. Terminating a Background Process.

There may be a need to stop a process from executing any further after a point of time for the following reasons:

- An urgent job needs to be completed quickly, therefore all the unimportant processes need to be stopped immediately.
- A command/shell script does not terminate as desired, and the only way to stop it from executing is to terminate it.
- The terminal goes into a hang because the program being executed does not function as desired.

The **kill** command can be used shown below:

```
$ kill 278
```

Instead of the process ID, you can also specify the job ID with the % option. Here's the syntax:

```
kill %job_id
```

Note that the process identifier (278 in this case), which is first displayed when a background process is started, is given as an argument with the **kill** command.

3.8. Exercises

3.8.1. Multiple choice questions

a. To display the names of all the users who are currently logged in

(i) command 'who' is used
(ii) command 'whom' is used
(iii) command 'who am I' is used
(iv) command 'ls' is used.

b. '**man ls'** implies

(i) To get detailed information about the ls command
(ii) To get a few information about the ls command
(iii) To get a manual how Linux works
(iv) To get detailed user information.

3.8.2. Questions for short answers

a. What does 'clear' command do?
b. Why do you need to stop a process from executing any further after a point of time?
c. Write the syntax of kill command.

3.8.3. Analytical questions

a. Discuss the different command of manipulating screen with examples.
b. Narrate columns of the obtained result of 'who' command.