Lab 1 - Basic Linux Commands

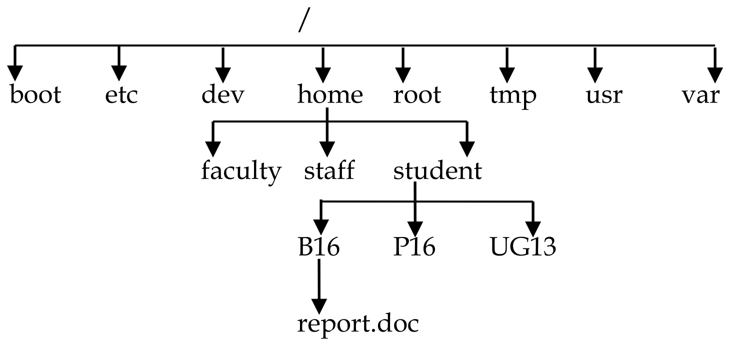
Using Applications

Launch a terminal from your desktop's application menu. The shell opens in the home directory and you get a prompt. Short cut to launch the terminal window *Alt* + *Ctl* + *T*

bash-4.3$

Linux Directory Structure

All the files are grouped together in the directory structure. The file---system is arranged in a hierarchical structure, like an inverted tree. The top of the hierarchy is traditionally called root (written as a slash /)



The full path to the file report.doc is "/home/student/B16/report.doc"

Note: By default the shell opens in your home directory.

File Administration Commands

ls [option(s)] [file_name(s)]

If you run **ls** without any additional parameters, the program will list the contents of the current directory in short form.

-i	detailed list
-a	displays hidden files

Example: ls

Example: ls –l

Example: ls –a

Example: ls –la

pwd

Sometimes you really wonder where you are in the system. PWD is the solution for that problem. PWD stands for Print Working Directory.

Example: pwd

cat [option(s)] [file_name(s)]

The **cat** command displays the contents of a file, printing the entire contents to the screen without interruption.

-n	display the line numbers on left margin

Example: cat test.txt	(display the contents of the file)
Example: cat > test.txt	(over wright the contents if file exist otherwise create new file)
Example: cat >> test.txt	(append contents to existing file)

Example: cat file1 file2 > newfile (merge contents of two files into single file)

cp [option(s)] source_file target_file

Copies source_file to target_file.

-i	Waits for confirmation, if necessary, before an existing target_file is overwritten
-r	Copies recursively (includes subdirectories)

Example: cp test.txt copy.txt

mv [option(s)] source_file target_file

Copies source_file to target_file then deletes the original source_file.

-b	Creates a backup copy of the source_file before moving
-i	Waits for confirmation, if necessary, before an existing target_file is overwritten

Example: mv text.txt new.txt

rm [option(s)] file_name(s)

Removes the specified files from the file system. Directories are not removed by **rm** unless the option -r is used.

-r	Deletes any existing subdirectories
-i	Waits for confirmation before deleting each file.

Example: rm test.txt

cd [options(s)] [directory]

Changes the current directory. **cd** without any parameters changes to the user's home directory.

Example: cd Downloads	(move to Downloads directory)
Example: cd /etc/	(move to etc directory)
Example: cd ..	(Directory up one level)

mkdir [option(s)] directory_name

Creates a new directory.

Example: mkdir dir1

rmdir [option(s)] directory_name

Deletes the specified directory, provided it is already empty.

Example: rmdir dir1

touch [file_name(s)]

Now we know how to make directories and deleting them, i now want to use the touch command. The touch command is used to make files.

Example: touch sample.txt

chown [option(s)] username.group file_name(s)

Transfers the ownership of a file to the user with the specified user name.

-R	Changes files and directories in all subdirectories.

chgrp [option(s)] groupname file_name(s)

Transfers the group ownership of a given file to the group with the specified group name. The file owner can only change group ownership if a member of both the existing and the new group.

chmod [options] mode file_name(s)

Changes the access permissions.

The mode parameter has three parts: group, access, and access type. group accepts the following characters:

u	user
g	group
o	others

For access, access is granted by the + symbol and denied by the - symbol.

The access type is controlled by the following options:

r	read
w	write
x	eXecute — executing files or changing to the directory.
s	Set uid bit — the application or program is started as if it were started by the owner of the file.

We have the following bits available:

7	full
6	read and write
5	read and execute
4	read only
3	write and execute
2	write only
1	execute only
0	none

locate [pattern(s)]

The locate command can find in which directory a specified file is located. If desired, use *wild cards* to specify file names. The program is very speedy, as it uses a database specifically created for the purpose (rather than searching through the entire file system). This very fact, however, also results in a major drawback: locate is unable to find any files created after the latest update of its database.

find [option(s)]

The **find** command allows you to search for a file in a given directory. The first argument specifies the directory in which to start the search. The option **-name** must be followed by a search string, which may also include wild cards. Unlike **locate**, which uses a database, **find** scans the actual directory.

Commands to Access File Contents

less [option(s)] file_name(s)

This command can be used to browse the contents of the specified file. Scroll half a screen page up or down with **PgUp** and **PgDn** or a full screen page down with **Space**. Jump to the beginning or end of a file using **Home** and **End**. Press **Q** to exit the program.

grep [option(s)] pattern filename(s)

The grep command finds a specific search pattern in the specified file_name(s). If the search string is found, the command displays the line in which the search pattern was found along with the file name.

-i	Ignores case
-l	Only displays the names of the respective files, but not the text lines
-n	Additionally displays the numbers of the lines in which it found a hit
-l	Only lists the files in which search pattern does not occur

diff [option(s)] file1 file2

The **diff** command compares the contents of any two files. The output produced by the program lists all lines that do not match. This is frequently used by programmers who need only send their program alterations and not the entire source code.

-q	Only reports *whether* the two given files differ

Processes

top [options(s)]

top provides a quick overview of the currently running processes. Press **H** to access a page that briefly explains the main options to customize the program.

ps [option(s)] [process ID]

If run without any options, this command displays a table of all *your own* programs or processes — those you started. The options for this command are not preceded by hyphen.

aux	Displays a detailed list of all processes, independent of the owner.

kill [option(s)] [process ID]

Unfortunately, sometimes a program cannot be terminated in the normal way. However, in most cases, you should still be able to stop such a runaway program by executing the **kill** command, specifying the respective process ID (see **top** and **ps**).

kill sends a *TERM* signal that instructs the program to shut itself down. If this does not help, the following parameter can be used:

-9 Sends a *KILL* signal instead of a *TERM* signal, with which the process really is *annihilated* by the operating system. This brings the specific processes to an end in almost all cases.

killall [option(s)] [process_name]

This command is similar to **kill**, but uses the process name (instead of the process ID) as an argument, causing all processes with that name to be killed.

Miscellaneous

man [command]

The man command shows the users the "manual" of the command. In some situation you might need to get more information about the command you are using. The man command shows you this information about the command.

Example: man cp

This will open up the "cp" manual document for us in the shell. The manual shows us the parameters available for the commands.

Note: To close the manual simply press "Q".

date [option(s)]

This simple program displays the current system time. If run as root, it can also be used to change the system time. Details about the program are available in date.

cal

This simple program displays the calendar of system date.

Example: cal	(display calendar of current month)
Example: cal 2016	(display calendar of 2016)
Example: cal mar 2016	(display calendar of month March 2016)

clear

This command cleans up the visible area of the console. It has no options.

who

This command is used to display the users information who are currently logged in.

whoami

This command is used to display the user name.

passwd [option(s)] [username]

Users may change their own passwords at any time using this command. Furthermore, the administrator root can use the command to change the password of any user on the system.

su [option(s)] [username]

The **su** command makes it possible to log in under a different user name from a running session. When using the command without specifying a user name, you will be prompted for the root password. Specify a user name and the corresponding password to use the environment of the respective user. The password is not required from *root*, as root is authorized to assume the identity of any user.

halt [option(s)]

To avoid loss of data, you should always use this program to shut down your system.

reboot [option(s)]

Does the same as **halt** with the difference that the system performs an immediate reboot.

File Editors

vi / vim [file_name]

This command are used to create the file and edit any existing file. vi has 2 mode 'input mode' and 'command mode'

1. **input mode**

   ESC to switch to command mode

2. **command mode**

   Switch to input mode
   - ***a*** (append), ***i*** (insert), ***o*** (insert line after), ***O*** (insert line before)

   Cursor movement
   - ***h*** (left), ***j*** (down), ***k*** (up), ***l*** (right), ***G*** (bottom page), ***:1*** (go to first line)

   Delete
   - ***dd*** (delete a line), ***d10d*** (delete 10 lines), ***d$*** (delete till end of line), ***dG*** (delete till end of file), ***x*** (current char)

   Paste
   - ***p*** (paste after), ***P*** (paste before)

   Undo
   - ***u***

   Search
   - ***/***

   Save/Quit
   - ***:w*** (write), ***:q*** (quit), ***:wq*** (write and quit), ***:q!*** (give up changes)

gedit [file_name]

This command are used to create the file and edit any existing file in graphical mode.