University of Central Punjab

(Incorporated by Ordinance No. XXIV of 2002 promulgated by Government of the Punjab)

FACULTY OF INFORMATION TECHNOLOGY

Operating System

Lab No 2

Overview of Ubuntu Directories and Linux Basic Shell Commands

Faculty of Information Technology

UCP Lahore Pakistan

Powered By M.waqar Mughal & Sherjeel Ahmed

Topic to be covered

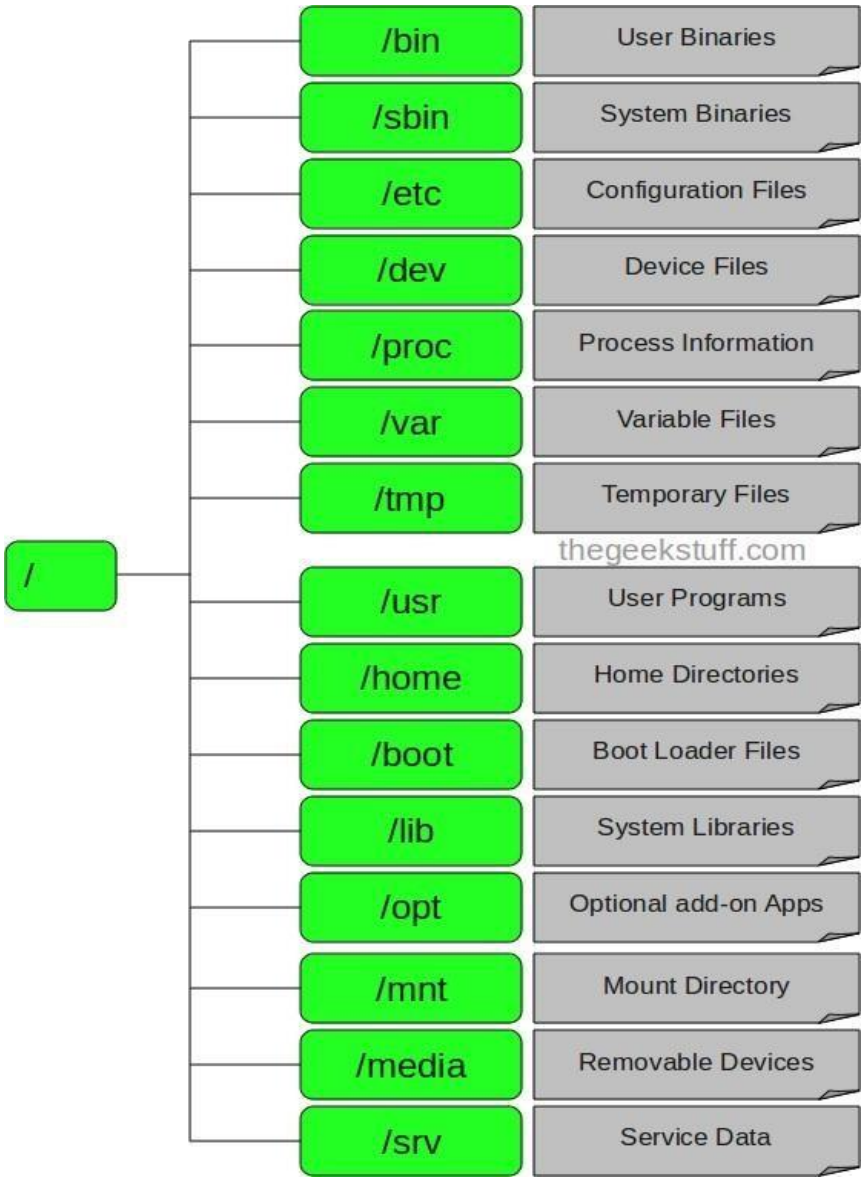
- Overview of Ubuntu Directories
- Basic Command on Linux for working on Terminal
    a. Helping Manual in Ubuntu using command
    b. Listing Files
    c. Listing Hidden Files
    d. Creating & Viewing Files
    e. Combining files
    f. Page wise content view
    g. Deleting Files
    h. Moving & Re-naming Files
    i. Copying Files
    j. Searching Contents of a file
        i. Simple Search using less
        ii. Grep
    i. Appending to a file
- Directory Manipulations
    a. Creating Directories
    b. Change directory
    c. Getting Directory path
    d. Removing Directories
    e. Renaming Directories
    f. Coping Directories
- Other Useful Command
    a. Clear
    b. Redirections
    c. Pipes
    d. Wild Cards
    e. History
    f. Sudo

Objectives

- Students are able to understand Linux directories
- Students are able to understand and use different Linux commands.

Overview of Ubuntu Directories

We will learn the Linux file system structures and understand the meaning of individual high-level directories.



1. **/ – Root**

Every single file and directory starts from the root directory. Only root user has write privilege under this directory. Please note that **/root** is root user's home directory, which is not same as **/.**

Further details: - Root

2. **/bin – User Binaries**

Contains binary executables. Common Linux commands you need to use in single-user modes are located under this directory. Commands used by all the users of the system are located here.

For example: ps, ls, ping, grep, and cp.
Further details- User Binaries

3. **/sbin – System Binaries**

Just like **/bin, /sbin** also contain binary executables. But, the Linux commands located under this directory are used typically by system administrator, for system maintenance purpose.

For example: iptables, reboot, fdisk, ifconfig, swapon
Further details- User Binaries

4. **/etc – Configuration Files**

Contains configuration files required by all programs. This also contains startup and shutdown shell scripts used to start/stop individual programs.

For example: /etc/resolv.conf, /etc/logrotate.conf
Further details: - Configuration Files

5. **/dev – Device Files**

Contains device files. These include terminal devices, usb, or any device attached to the system.

For example: /dev/tty1, /dev/usbmon0
Further details: - Device Files
Further details: - Device Details 2

6. **/proc – Process Information**

Contains information about system process. This is a pseudo filesystem contains information about running process. For example: **/proc/** {PID} directory contains information about the process with that particular PID. This is a virtual filesystem with text information about system resources.

For example: /proc/uptime
Further details: - Process Detail

7. **/var – Variable Files** var stands for variable files.

Content of the files that are expected to grow can be found under this directory

For example: log files **(/var/log);** packages and database files **(/var/lib);** emails **(/var/mail);** print queues **(/var/spool)**; **lock files (/var/lock)**; temp files needed across reboots **(/var/tmp)**;

Further details: - Variable Files

8. **/tmp – Temporary Files**

Directory that contains temporary files created by system and users. Files under this directory are deleted when system is rebooted.

Further details: - Temp Files

9. **/usr – User Programs**

Contains binaries, libraries, documentation, and source-code for second level programs. **/usr/bin** contains binary files for user programs. If you can't find a user binary under **/bin**, look under **/usr/bin**.

For example: at, awk, cc, less, scp
/usr/sbin contains binary files for system administrators. If you can't find a system binary under /sbin, look under /usr/sbin. For example: atd, cron, sshd, useradd, userdel /usr/lib contains libraries for /usr/bin and /usr/sbin
/usr/local contains users programs that you install from source. For example, when you install apache from source, it goes under /usr/local/apache2

10. **/home – Home Directories**

Home directories for all users to store their personal files.

For example: /home/Sherjeel, /home/Waqar

11. **/boot – Boot Loader Files**

Contains boot loader related files. Kernel initrd, vmlinux, grub files are located under /boot

For example: initrd.img-2.6.32-24-generic, vmlinuz-2.6.32-24-generic

12. /lib – System Libraries

Contains library files that supports the binaries located under /bin and /sbin. Library filenames are either ld* or lib*.so.*

For example: ld-2.11.1.so, libncurses.so.5.7

13. **/opt – Optional add-on Applications** opt stands for optional.

Contains add-on applications from individual vendors. Add-on applications should be installed under either /opt/ or /opt/ subdirectory.

14. /mnt – Mount Directory

Temporary mount directory where sysadmins can mount filesystems.

15. /media – Removable Media Devices

Temporary mount directory for removable devices.

For examples, /media/cdrom for CD-ROM; /media/floppy for floppy drives; /media/cdrecorder for CD writer.

16. **/srv – Service Data** srv stands for service.

Contains server specific services related data.

For example, /srv/cvs contains CVS (Concurrent version system) related data.

Source: http://www.thegeekstuff.com/2010/09/linux-file-system-structure/

Basic Command on Linux for working on Terminal

Launch a terminal by pressing alt + ctrl + t or press windows and type terminal

a. Helping Manual in Ubuntu using command

The 'Man' command

Man stands for manual which is a reference book of a Linux operating system. It is similar to HELP file found in popular software.

To get help on any command that you do not understand, you can type

man

The terminal would open the manual page for that command.

For an example, if we type *man* **ls** and hit enter; terminal would give us information on **ls** command.

Figure 1: man command for ls

When you type this command and hit Enter you will get the detailed manual for ls command as show in figure below.

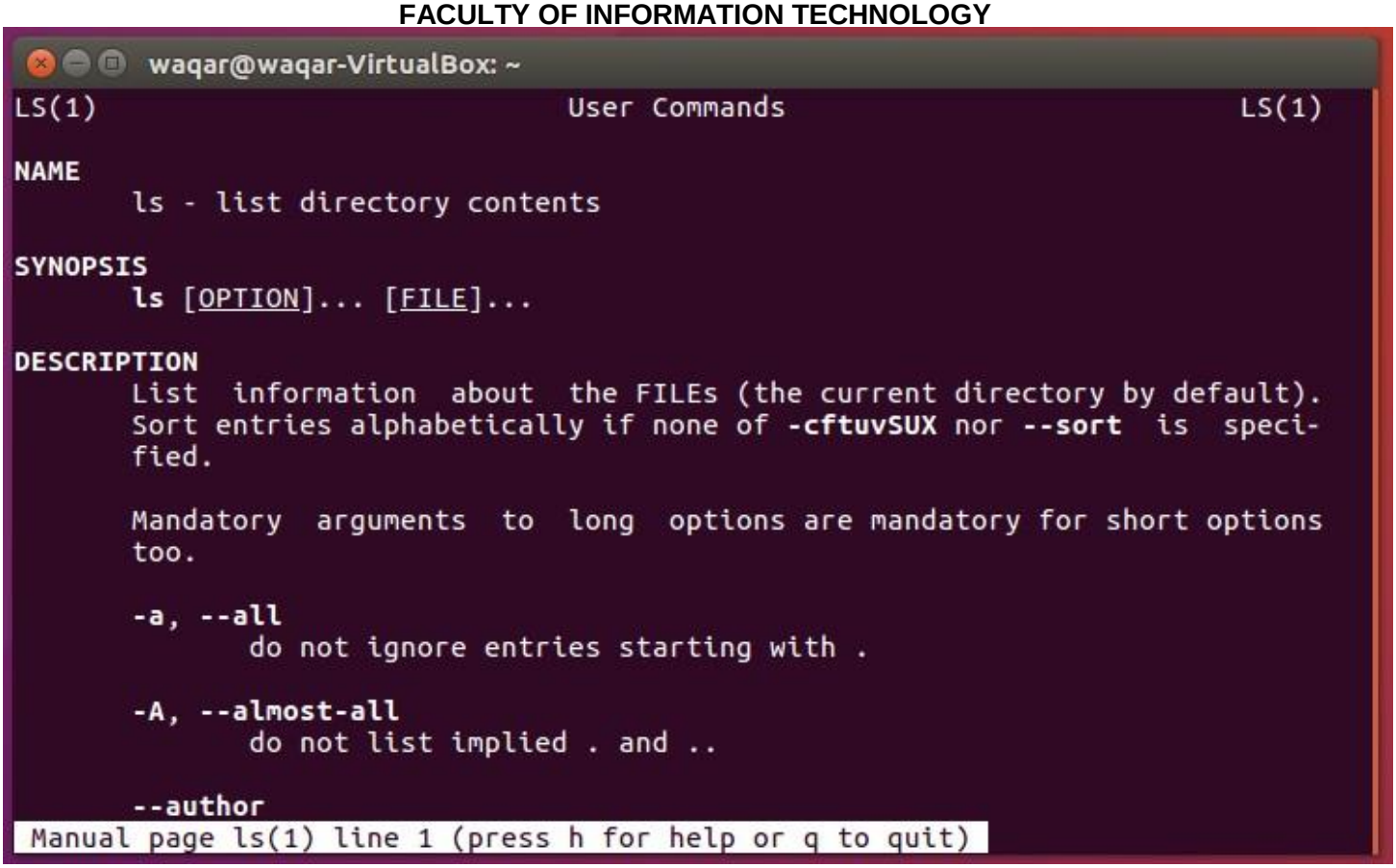


Figure 2: Detail manual for ls command

b. Listing Files

When you first login, your current working directory is your home directory. Your home directory has the same name as your user-name, for example, **Waqar**, and it is where your personal files and subdirectories are saved.

To find out what is in your home directory, type

ls

The **ls** command (lowercase L and lowercase S) lists the contents of your current working directory.

Figure 3: ls Command

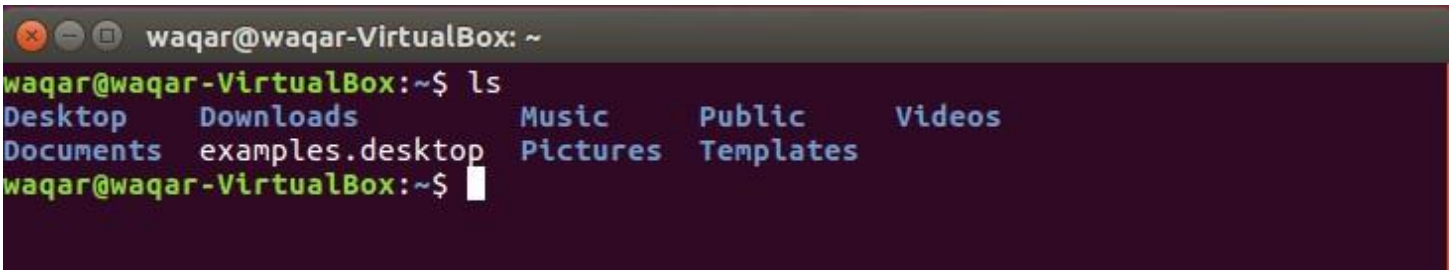


Figure 4: ls command execution

There may be no files visible in your home directory, in which case, the LINUX prompt will be returned. Alternatively, there may already be some files inserted by the System Administrator when your account was created.

ls does not, in fact, cause all the files in your home directory to be listed, but only those ones whose name does not begin with a dot (.) Files beginning with a dot (.) are known as hidden files and usually contain important program configuration information. They are hidden because you should not change them unless you are very familiar with LINUX!!!

Note:

- Directories are denoted in blue color.
- Files are denoted in white.
- You will find similar color schemes in different flavors of Linux.

Suppose, in your "Music" folder has following sub-directories and files

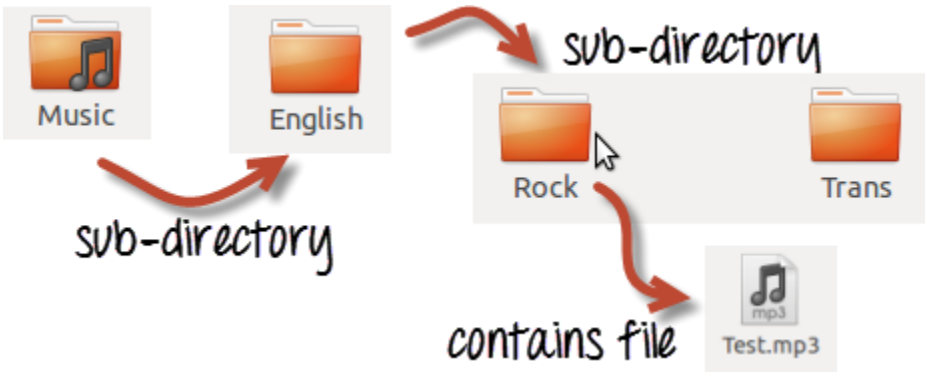


Figure 5: Sample Example

You can use **'ls -R'** to shows all the files not only in directories but also subdirectories.

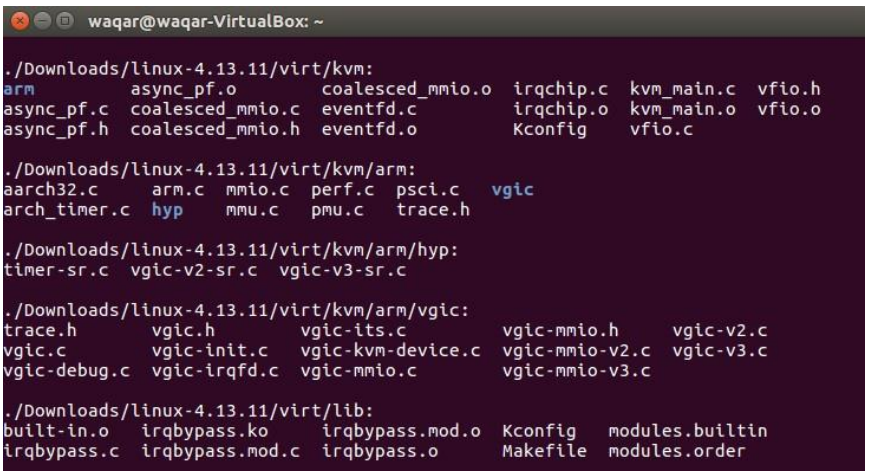


Figure 6: ls -r execution

c. Listing Hidden Files

To list all files in your home directory including those whose names begin with a dot type following command.

ls -a

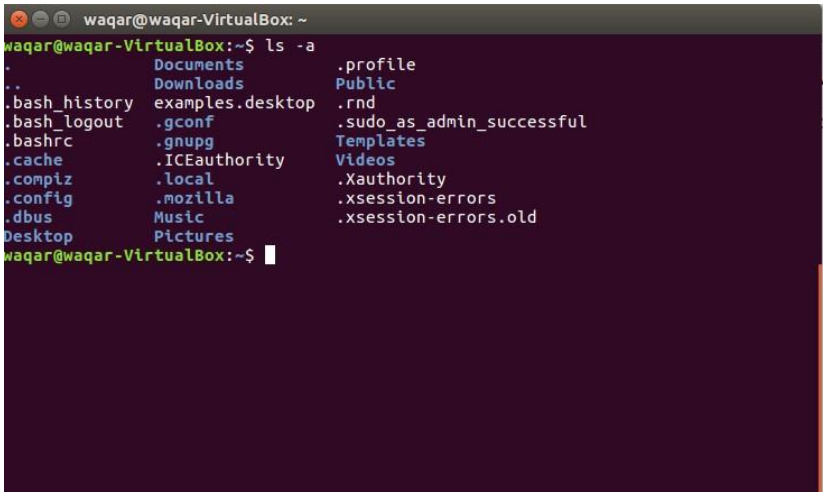


Figure 7: ls -R command execution

As you can see, **ls -a** lists files that are normally hidden.
ls is an example of a command which can take options: "**-a"** is an example of an option. The options change the behavior of the command. There are online manual pages that tell you which options a particular command can take, and how each option modifies the behavior of the command. (See later in this tutorial).

d. Creating & Viewing Files

You can use gedit, cat and touch command for creating a file.

1. **Gedit**

   If you want to create a file and open in text editor then type following command

   gedit yourfilename.extension

   gedit OS.txt

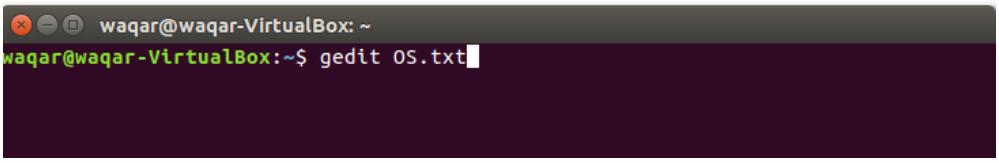


Figure 8: File creating using gedit

2. Cat

If you want to create file and input in it using command line then you can use cat command.

cat > yourfilename.extension
cat > os.txt

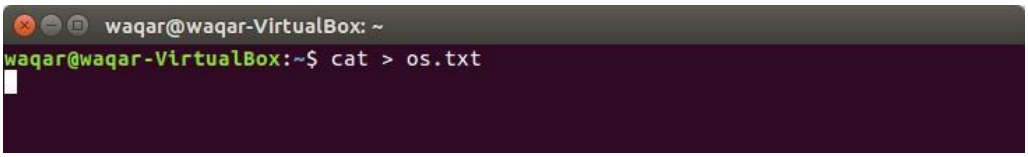


Figure 9: File creation using cat

After this what text you enter will be written in file and you need to press Ctrl+d to end the input in file.

3. Touch

If you want to create an empty file and don't want it to open in any editor or terminal then you can use touch command.

touch yourfilename.extension
touch Os.txt

Figure 10: File creation using touch

To see the content of your file you can use **cat** command as well

cat yourfilename. extension
cat OS.txt

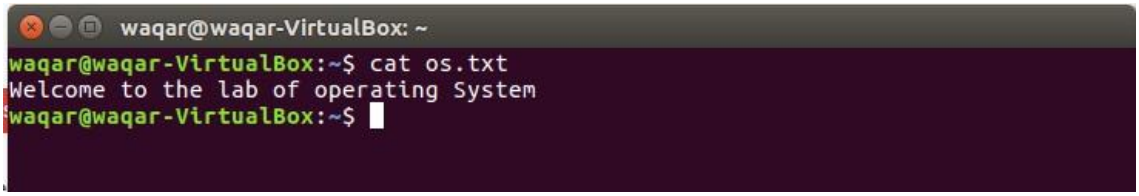


Figure 11: Data display of file using cat

e. Combining Files

if you want to combine different files then you can also use cat command

cat yourfilename.extension yourfilename.extension > newfilename.extension
cat OS.txt os.txt > waqar.txt

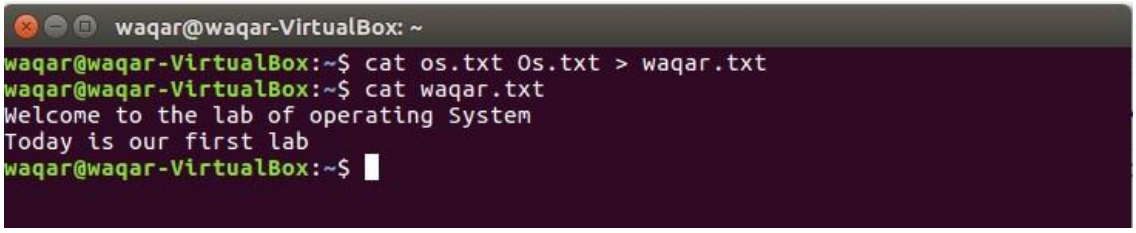


Figure 12: Combining a file

f. Page wise content view

Let us assume that there is a big file or having data up to 30 Mb in the form of text. Than the size of terminal is small to display the data of file. So, we can display data page wise by using **less** command.

less yourfilename.extension

less OS.txt

University of Central Punjab

(Incorporated by Ordinance No. XXIV of 2002 promulgated by Government of the Punjab)

FACULTY OF INFORMATION TECHNOLOGY

In Lab Task1

You have to create 2 different text files. One with your name and the second one with your registration no. Copy and paste some dummy data from google in both file. You can even use following link for sample data

http://www.gutenberg.org/wiki/Category:Bookshelf

after that do the following task

1) Display the data of both files
2) Display data page wise of both files
3) Combine both files in new file you have to submit

screenshots of all tasks in which results and command clearly visible.

g. Deleting Files

The 'rm' command removes files from the system without confirmation.

rm yourfilename.extension

rm os.txt

to check if file is delete or not type ls and see there will be no file with name os.txt

h. Moving & Renaming Files

mv file1 new_file_location is the command which moves a file to new destination.

What we are going to do now, is to take a file stored in an open access area of the file system and use the **mv** command to move it to your Desktop.

mv filename newfilename will rename the file.

i. Copying Files

cp *file1 file2* is the command which makes a copy of file1 in the current working directory and calls it file2

What we are going to do now, is to take a file stored in an open access area of the file system and use the **cp** command to copy it to your Desktop.

j. Searching Contents of a file

i. Simple search using less

Using **less**, you can search though a text file for a keyword (pattern). For example, to search through os.txt for the word **'science'**, type

less os.txt

then, still in **less**, type a forward slash [**/**] followed by the word to search

/science

As you can see, **less** finds and highlights the keyword. Type [**n**] to search for the next occurrence of the word.

ii. Grep

grep is one of many standard LINUX utilities. It searches files for specified words or patterns. First clear the screen using clear command, then type

grep science os.txt

As you can see, **grep** has printed out each line containing the word **science**.

Or has it ????
Try typing

grep Science os.txt

The **grep** command is case sensitive; it distinguishes between Science and science.

To ignore upper/lower case distinctions, use the -i option, i.e. type

grep -i science os.txt

To search for a phrase or pattern, you must enclose it in single quotes (the apostrophe symbol). For example, to search for spinning top, type

grep -i 'spinning top' os.txt

Some of the other options of grep are:

-v display those lines that do NOT match

-n precede each matching line with the line number

-c print only the total count of matched lines

Try some of them and see the different results. Don't forget, you can use more than one option at a time. For example, the number of lines without the words science or Science is

grep -ivc science os.txt

k. wc (word count) of File

A handy little utility is the **wc** command, short for word count. To do a word count on os.txt, type

wc -w os.txt

To find out how many lines the file has, type

wc -l os.txt

l. Appending to a file

The form **>>** appends standard output to a file. So, to add more items to the file list1, type

cat >> list1

Then type in the names of more fruit

peach

grape

orange

^D (Control D to stop)

To read the contents of the file, type

cat list1

You should now have two files. One contains six fruits, the other contains four fruits.

We will now use the **cat** command to join (concatenate) list1 and list2 into a new file called biglist. Type

cat list1 list2 > biglist

What this is doing is reading the contents of list1 and list2 in turn, then outputting the text to the file biglist

To read the contents of the new file, type

cat biglist

We use the **<** symbol to redirect the input of a command.

The command sort alphabetically or numerically sorts a list. Type

sort

Then type in the names of some animals. Press [Return] after each one.

dog

cat

bird

ape

^D (control d to stop)

The output will be

ape

bird

cat

dog

Using **<** you can redirect the input to come from a file rather than the keyboard. For example, to sort the list of fruit, type

sort < biglist

and the sorted list will be output to the screen.

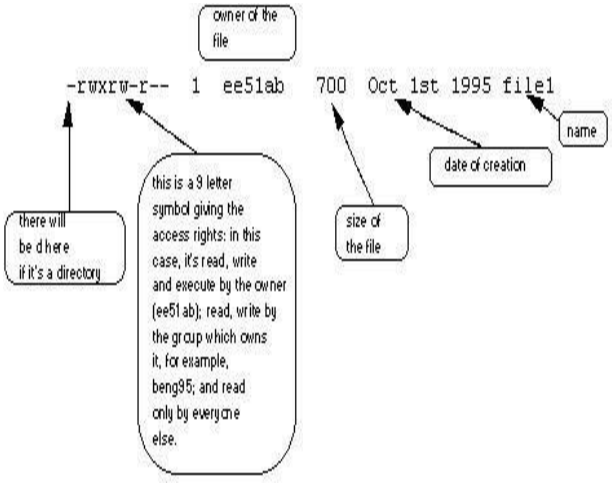
To output the sorted list to a file, type,

sort < biglist > slist

Use cat to read the contents of the file slist

m.l s -l (l for long listing!)

You will see that you now get lots of details about the contents of your directory, similar to the example below.



Each file (and directory) has associated access rights, which may be found by typing **ls -l**. Also, **ls -lg** gives additional information as to which group owns the file (beng95 in the following example):

-rwxrw-r-- 1 ee51ab beng95 2450 Sept29 11:52 file1

In the left-hand column is a 10 symbol string consisting of the symbols d, r, w, x, -, and, occasionally, s or S. If d is present, it will be at the left hand end of the string, and indicates a directory: otherwise - will be the starting symbol of the string.

The 9 remaining symbols indicate the permissions, or access rights, and are taken as three groups of 3.

- The left group of 3 gives the file permissions for the user that owns the file (or directory) (ee51ab in the above example);
- the middle group gives the permissions for the group of people to whom the file (or directory) belongs (eebeng95 in the above example);
- the rightmost group gives the permissions for all others.

The symbols r, w, etc., have slightly different meanings depending on whether they refer to a simple file or to a directory.

Access rights on files.

- r (or -), indicates read permission (or otherwise), that is, the presence or absence of permission to read and copy the file
- w (or -), indicates write permission (or otherwise), that is, the permission (or otherwise) to change a file
- x (or -), indicates execution permission (or otherwise), that is, the permission to execute a file, where appropriate

Access rights on directories.

- r allows users to list files in the directory;
- w means that users may delete files from the directory or move files into it;
- x means the right to access files in the directory. This implies that you may read files in the directory provided you have read permission on the individual files.

So, in order to read a file, you must have execute permission on the directory containing that file, and hence on any directory containing that directory as a subdirectory, and so on, up the tree.

Some examples

-rwxrwxrwx: a file that everyone can read, write and execute (and delete). a file that only the owner can read and write - no-one else

-rw-------: can read or write and no-one has execution rights (e.g. your mailbox file).

m. Changing access rights

chmod (changing a file mode)

Only the owner of a file can use chmod to change the permissions of a file. The options of chmod are as follows

Symbol	Meaning
u	user
g	group
o	other
a	all
r	read

w	write (and delete)
x	execute (and access directory)
+	add permission
-	take away permission

For example, to remove read write and execute permissions on the file os.txt for the group and others, type

chmod go-rwx os.txt

This will leave the other permissions unaffected.

To give read and write permissions on the fileos.txt to all,

chmod a+rw os.txt

In Lab Task2

Try changing access permissions on the file os.txt

Use **ls -l** to check that the permissions have changed.

Here you have practice some useful commands regarding File Manipulation. Now move to Directories manipulation.

- **Directory Manipulation**

a) Creating Directories

Directories can be created on a Linux operating system using the following command

mkdir directoryname
mkdir OS

If you want to create a directory in a different location other than 'Home direct ory', you could use the following command.

mkdir /tmp/MUSIC

will create a directory 'Music' under '/tmp' directory

b) Change directory

The command **cd** means change the current working directory to desire 'directory'. The current working directory may be thought of as the directory you are in, i.e. your current position in the file-system tree.

To change to the directory you have just made, type

cd OS

The current directory (.)

In LINUX, (.) means the current directory, so typing

cd .

NOTE: There is a space between cd and dot

Means stay where you are (the OS directory).

This may not seem very useful at first but using (.) as the name of the current directory will save a lot of typing, as we shall see later in the tutorial.

The parent directory (..)

(..) means the parent of the current directory, so typing

cd ..

will take you one directory up the hierarchy (back to your home directory). Try it now.

Note: typing **cd** with no argument always returns you to your home directory. This is very useful if you are lost in the file system.

~ (your home directory)

Home directories can also be referred to by the tilde ~ character. It can be used to specify paths starting at your home directory. So typing

ls ~/OS

Will list the contents of your OS directory, no matter where you currently are in the file system.

What do you think **ls ~** would list? What do you think **ls ~/..** Would list?

c) Getting Directory path

Pathnames enable you to work out where you are in relation to the whole filesystem. For example, to find out the absolute pathname of your home directory, type cd to get back to your home-directory and then type

pwd

The full pathname will look something like this -

/home/OS

which means that OS (your home directory) is in the sub-directory home, which is in the home sub-directory, which is in the top-level root directory called "/ ".

d) Removing Directories

To remove a directory, use the command

rmdir yourdirectoryname

rmdir OS

Note:- Ensure that there is no file / sub-directory under the directory that you want to delete. Delete the files/sub-directory first before deleting the parent directory.

e) Renaming Directories:

Same command as used in Files System.

f) Coping Directories

Same command as used in Files System.

- **Other useful LINUX commands**

a. Clear

Whenever you want to clear the terminal use clear command

b. Redirections

Most processes initiated by LINUX commands write to the standard output (that is, they write to the terminal screen), and many take their input from the standard input (that is, they read it from the keyboard). There is also the standard error, where processes write their error messages, by default, to the terminal screen.

We have already seen one use of the **cat** command to write the contents of a file to the screen.

c. Pipes

To see who is on the system with you, type

who

One method to get a sorted list of names is to type,

who > names.txt

sort < names.txt

This is a bit slow and you have to remember to remove the temporary file called names when you have finished. What you really want to do is connect the output of the who command directly to the input of the sort command. This is exactly what pipes do. The symbol for a pipe is the vertical bar |

For example, typing **who | sort**

will give the same result as above, but quicker and cleaner.

To find out how many users are logged on, type

who | wc -l

d. Wild Cards

The * wildcard

The character * is called a wildcard and will match against none or more character(s) in a file (or directory) name. For example, in your OS directory, type

ls list*

This will list all files in the current directory starting with list....

Try typing

ls *list

This will list all files in the current directory ending with.....list

The ? wildcard

The character ? will match exactly one character. So ?ouse will match files like house and mouse, but not grouse.

Try typing

ls ?list

e. History

To see the history of commands you have used so far

history

Additional Links:

https://www.guru99.com/must-know-linux-commands.html

http://www.mediacollege.com/linux/command/linux-command.html

https://fossbytes.com/a-z-list-linux-command-line-reference/

https://www-uxsup.csx.cam.ac.uk/pub/doc/suse/suse9.0/userguide-9.0/ch24s04.html