Viewing Files:-

1. How do you view a text file with the ability to scroll through the pages of the file.

Less –M xyz.txt

Less –b xyz.txt

2. How do you view the first 10 lines of the file?

head -10 /etc/xyz.txt

**sed** -n 1,10p **/**etc**/**group

3. How do you view the last 10 lines of the file?

tail -10 /etc/xyz.txt

4. How do you see the last few lines of file as new data written to it in realtime.

$ tail -v state.txt

Analyzing Text:-

1. Write the first few lines of output of "ls -la" to the file "data.txt"

sed -i '1i new\_text' file\_name

2. Write the last few line of output of "la -la" to an existing file "data.txt" without overwriting the existing content in it.

sed -e "\$aTEXTTOEND"

3. From the data.txt file, extract last 3 columns

awk 'NF{print $(NF-2),$(NF-1),$NF}' OFS="\t" Input\_file

$ cat file | tr -s '\n' | rev | cut -f 1-3 | rev

4. Sort the content of the file alphabetically based on first character in the file

sort -k 1.4 file

sort -b -k 2.2 file

second character of second field, ignoring blanks.

5. Sort the content of the file numerically

sort -n abc.txt

6. Print number of lines in the file.

$ wc -l /dir/file.txt

7. Print number of words in the file.

wc /dir/file.txt

8. Print number of character in the file.

wc -l /dir/file.txt

Pipes and Regular Expressions:-

1. Check the manual page for "grep" command

NOTE: Find below regex to create Pattern:-

^ = Search the beginning of the line

$ = Search the end of a line

. = Stands in for a single character

[abc] = Search for specified characters

[^abc] = Search for other characters except for these

\* = Match Zero or more number of preceding characters or expression

NOTE:-

| = This is a pipe symbol, used to send the output of one command as a input to other command.

Example:- command1 | command2

NOTE:-I often find people try and write their pipes all out in one go and make a mistake somewhere along the line. They then think it is in one point but in fact it is another point. They waste a lot of time trying to fix a problem that is not there while not seeing the problem that is there. If you build your pipes up incrementally then you won't fall into this trap. Run the first program and make sure it provides the output you were expecting. Then add the second program and check again before adding the third and so on. This will save you a lot of frustration.

$ ls | head -3 | tail -1 > myoutput

2. return lines that matches a pattern of your choice.

grep "Linux" input.txt

# grep -i "LINUX" input.txt Ignore case

# grep -r "Linux" \*

input.txt:Welcome to Linux.

new\_dir/new.txt:Linux vs Windows

output.txt:I hope you enjoyed working on Linux.

Binary file test\_strace matches

test\_strace.c: if(NULL == fopen("Linux","rw"))

3. return lines that does not matches the pattern of your choice.

grep -r [!Linux] \*

4. Perform a recursive search on a pattern your choice.

# grep -r "Linux" \*

5. Perform a case-insensitive search on a pattern of your choice.

grep -i "LINUX" input.txt Ignore case

6. Find the lines that matches a pattern of your choice, but display only the first 3 lines out of all the lines mathed.

Ls –l3 | grep -i "LINUX" input.txt Ignore case

7. Find out how many times a particular word appears in the file.

**$grep -c "unix" geekfile.txt**

8. Use plato\_republic.txt file, and display all the line that starts with "Republic".

$ grep -w " **Republic** " **plato\_republic.txt**

9. Use Plato\_republic.txt file, and display all the lines that ends with "Republic".

$ grep "$**Republic**" **plato\_republic.txt**

10. Use plato\_republic.txt file, and display the number of lines that has word "republic".

$**grep** -c " **Republic" plato\_republic.txt**

11. Use plato\_republic.txt file, and display all the lines that begins with either a or A or b or B

$ grep –h [^A|^a]+[^b|^B] **plato\_republic.txt**

12. Use plato\_republic.txt file, and display all the lines that does not begins with either a or A or b or B

$ grep –h ! [^A|^a]+[^b|^B] **plato\_republic.txt**

13. Use plato\_republic.txt file, and display all the lines that has "h" or "H" as the second character.

$ grep –h .2 [^H|^h] **plato\_republic.txt**

14. Use plato\_republic.txt file, and display all the lines that has internet web address

**grep –h ^([a-zA-Z0-9\_\-\.]+)@([a-zA-Z0-9\_\-\.]+)\.([a-zA-Z]{2,5})$ plato\_republic.txt**

15. Sort the listing of a directory "/etc" so that all the directories are listed first.

**$ tree -a ./etc**

16. Long List the content of "/etc" directory, but make sure you view it comfortably

**$ tree –L ./etc**

17. Identify all files in your home directory which the group has write permission for.

**getfacl /home**

18. Create a listing of every user which owns a file in a given directory as well as how many files and directories they own.

**getfacl –h ./**

19. Get a count of how many files and directories you have the execute permission for in your home directory.

**getfacl -m u:mandeep:rwx ./home**

---Activities

Let's play with some patterns.

1. A good directory to play with is /etc which is a directory containing config files for the system. As a normal user you may view the files but you can't make any changes so we can't do any harm. Do a listing of that directory to see what's there. Then pick various subsets of files and see if you can create a pattern to select only those files.

2. Do a listing of /etc with only files that contain an extension.

**$ ls -l | find ./etc -type f -name "\*.config"**

3. What about only a 3 letter extension?

**$ ls -l | find ./etc -type f -name "\*.con"**

4. How about files whose name contains an uppercase letter? (hint: [[:upper:]] may be useful here)

**$ ls -l | find ./etc -type f -name [[:upper:]\*]**

5. Can you list files whose name is 4 characters long?

**$ ls -l | find ./etc -type f -name "????"**

----Activities

Let's mangle some data. You can use the plato\_republic.txt file

1. Now play with each of the programs below. Make sure you use both relative and absolute paths.

head

tail

sort

nl

wc

cut

sed

uniq

tac

Have a look at the man page for each of the programs and try at least 2 of the command line options for them.

---Activities

Let's identify some information. You can use the plato\_republic.txt file

1. Now play with some of the examples of grep command with regular expressions.

2. Have a look at the man page for "egrep" and try atleast 2 of the command line options for them.

-----VI/VIM Editor

Operation Modes

While working with the vi editor, we usually come across the following two modes -

Command mode - This mode enables you to perform administrative tasks such as saving the files, executing the commands, moving the cursor, cutting (yanking) and pasting the lines or words, as well as finding and replacing. In this mode, whatever you type is interpreted as a command.

Insert mode - This mode enables you to insert text into the file. Everything that's typed in this mode is interpreted as input and placed in the file.

vi always starts in the command mode. To enter text, you must be in the insert mode for which simply type "i". To come out of the insert mode, press the "Esc" key, which will take you back to the command mode.

When we run vi we normally issue it with a single command line argument which is the file you would like to edit.

$ vi <file>

If you forget to specify a file then there is a way to open a file within vi but it is easiest to just quit vi and have another go. Also remember that when we specify the file it can be with either an absolute or relative path.

Let's dive in and get started. It's going to be hard for me to demonstrate a lot of this so instead I'll list what I want you to type and you'll have to give it a go and see what happens.

First off let's move into your directory you created in the section on file manipulation. We're going to create a few files and this will keep them out of the way of your normal stuff.

Now we'll edit our first file.

$ vi firstfile

When you run this command it opens up the file. If the file does not exist then it will create it for you then open it up. (no need to touch files before editing them)

You always start off in edit mode so the first thing we are going to do is switch to insert mode by pressing i. You can tell when you are in insert mode as the bottom left corner will tell you.

Now type in a few lines of text and press Esc which will take you back to edit mode.

---Saving and Exiting File

There are a few ways to go about doing this. They all do essentially the same thing so pick whichever way you prefer. For all of these, make sure you are in edit mode first.

NOTE:- If you are unsure if you are in edit mode or not you can look at the bottom left corner. As long as it doesn't say INSERT you are fine. Alternatively you can just press Esc to be sure. If you are already in edit mode, pressing Esc does nothing so you won't do any harm.

ZZ (Note: capitals) - Save and exit

:q! - discard all changes, since the last save, and exit

:w - save file but don't exit

:wq - again, save and exit

Most commands within vi are executed as soon as you press a sequence of keys. Any command beginning with a colon ( : ) requires you to hit <enter> to complete the command.

Save and exit the file you currently have open

---Other ways to view files

vi allows us to edit files. If we wanted, we could use it to view files as well, but there are two other commands which are a bit more convenient for that purpose. The first one is cat which actually stands for concatenate. It's main purpose is to join files together but in it's most basic form it is useful for just viewing files.

$ cat <file>

If you run the command cat, giving it a single command line argument which is the file we just created, you will see it's contents displayed on the screen, followed by the prompt.

NOTE:- If you accidentally run cat without giving it a command line argument you will notice that the cursor moves to the next line and then nothing happens. Because we didn't specify a file, cat instead reads from something called STDIN (which we'll learn about in the section 'Piping and redirection' which defaults to the keyboard. If you type something then hit <enter> you will see cat mirror your input to the screen. To get out of here you may press <Ctrl> + c which is the universal signal for Cancel in Linux.

In fact, whenever you get in trouble you can generally press <Ctrl> + c to get yourself out of trouble.

This command is nice when we have a small file to view but if the file is large then most of the content will fly across the screen and we'll only see the last page of content. For larger files there is a better suited command which is "less".

$ less <file>

less allows you to move up and down within a file using the arrow keys. You may go forward a whole page using the SpaceBar or back a page by pressing b. When you are done you can press q for quit.

Have a look at the file you just created now using both these commands.

---Navigating a file in Vi

Now let's go back into the file we just created and enter some more content. In insert mode you may use the arrow keys to move the cursor around. Enter two more paragraphs of content then hit Esc to go back to edit mode.

Below are some of the many commands you may enter to move around the file. Have a play with them and see how they work.

Arrow keys - move the cursor around

j, k, h, l - move the cursor down, up, left and right (similar to the arrow keys)

^ (caret) - move cursor to beginning of current line

$ - move cursor to end of the current line

nG - move to the nth line (eg 5G moves to 5th line)

G - move to the last line

w - move to the beginning of the next word

nw - move forward n word (eg 2w moves two words forwards)

b - move to the beginning of the previous word

nb - move back n word

{ - move backward one paragraph

} - move forward one paragraph

NOTE:- If you type :set nu in edit mode within vi it will enable line numbers. I find that turning line numbers on makes working with files a lot easier.

---Deleting content

We just saw that if we want to move around in vi there are quite a few options available to us. Several of them also allow us to precede them with a number to move that many times. Deleting works similar to movement, in fact several delete commands allow us to incorporate a movement command to define what is going to be deleted.

Below are some of the many ways in which we may delete content within vi. Have a play with them now. (also check out the section below on undoing so that you can undo your deletes.)

x - delete a single character

nx - delete n characters (eg 5x deletes five characters)

dd - delete the current line

dn - d followed by a movement command. Delete to where the movement command would have taken you. (eg d5w means delete 5 words)

---Undoing

Undoing changes in vi is fairly easy. It is the character u.

u - Undo the last action (you may keep pressing u to keep undoing)

U (Note: capital) - Undo all changes to the current line

---Activities

Let's play with some content.

1. Start by creating a file and putting some content into it.

2. Save the file and view it in both cat and less

3. Go back into the file in vi and enter some more content.

4. Move around the content using at least 6 different movement commands.

5. Play about with several of the delete commands, especially the ones that incorporate a movement command. Remember you may undo your changes so you don't have to keep putting new content in.

---Permissions

Linux permissions dictate 3 things you may do with a file, read, write and execute. They are referred to in Linux by a single letter each.

r read - you may view the contents of the file.

w write - you may change the contents of the file.

x execute - you may execute or run the file if it is a program or script.

s -> Assignment

For every file we define 3 sets of people for whom we may specify permissions.

owner - a single person who owns the file. (typically the person who created the file but ownership may be granted to some one else by certain users)

group - every file belongs to a single group.

others - everyone else who is not in the group or the owner.

Three persmissions and three groups of people. That's about all there is to permissions really. Now let's see how we can view and change them.

--View Permissions

To view permissions for a file we use the long listing option for the command ls.

$ ls -l [path]

In the above output the first 10 characters of the output are what we look at to identify permissions.

The first character identifies the file type. If it is a dash ( - ) then it is a normal file. If it is a d then it is a directory.

The following 3 characters represent the permissions for the owner. A letter represents the presence of a permission and a dash ( - ) represents the absence of a permission. In this example the owner has all permissions (read, write and execute).

The following 3 characters represent the permissions for the group. In this example the group has the ability to read but not write or execute. Note that the order of permissions is always read, then write then execute.

Finally the last 3 characters represent the permissions for others (or everyone else). In this example they have the execute permission and nothing else.

--Change Permissions

To change permissions on a file or directory we use a command called chmod It stands for change file mode bits which is a bit of a mouthfull but think of the mode bits as the permission indicators.

chmod [permissions] [path]

chmod has permission arguments that are made up of 3 components

Who are we changing the permission for? [ugoa] - user (or owner), group, others, all

Are we granting or revoking the permission - indicated with either a plus ( + ) or minus ( - )

Which permission are we setting? - read ( r ), write ( w ) or execute ( x )

The following examples will make their usage clearer.

Grant the execute permission to the group. Then remove the write permission for the owner.

$ ls -l frog.png

$ chmod g+x frog.png

$ ls -l frog.png

$ chmod u-w frog.png

$ ls -l frog.png

Don't want to assign permissions individually? We can assign multiple permissions at once.

$ ls -l frog.png

$ chmod g+wx frog.png

$ ls -l frog.png

$ chmod go-x frog.png

$ ls -l frog.png

It may seem odd that as the owner of a file we can remove our ability to read, write and execute that file but there are valid reasons we may wish to do this. Maybe we have a file with data in it we wish not to accidentally change for instance. While we may remove these permissions, we may not remove our ability to set those permissions and as such we always have control over every file under our ownership.

---Setting Permissions Shorthand

The method outlined above is not too hard for setting permissions but it can be a little tedious if we have a specific set of permissions we sould like to apply regularly to certain files (scripts for instance that we'll see in section 13). Luckily, there is a shorthand way to specify permissions that makes this easy.

To understand how this shorthand method works we first need a little background in number systems. Our typical number system is decimal. It is a base 10 number system and as such has 10 symbols (0 - 9) used. Another number system is octal which is base 8 (0-7). Now it just so happens that with 3 permissions and each being on or off, we have 8 possible combinations (2^3). Now we can also represent our numbers using binary which only has 2 symbols (0 and 1). The mapping of octal to binary is in the table below.

Octal Binary

0 0 0 0

1 0 0 1

2 0 1 0

3 0 1 1

4 1 0 0

5 1 0 1

6 1 1 0

7 1 1 1

Now the interesting point to note is that we may represent all 8 octal values with 3 binary bits and that every possible combination of 1 and 0 is included in it. So we have 3 bits and we also have 3 permissions. If you think of 1 as representing on and 0 as off then a single octal number may be used to represent a set of permissions for a set of people. Three numbers and we can specify permissions for the user, group and others. Let's see some examples. (refer to the table above to see how they match)

$ ls -l frog.png

$ chmod 751 frog.png

$ ls -l frog.png

$ chmod 240 frog.png

$ ls -l frog.png

People often remember commonly used number sequences for different types of files and find this method quite convenient. For example 755 or 750 are commonly used for scripts.

---Permissions for Directories

The same series of permissions may be used for directories but they have a slightly different behaviour.

r - you have the ability to read the contents of the directory (ie do an ls)

w - you have the ability to write into the directory (ie create files and directories)

x - you have the ability to enter that directory (ie cd)

Let's see some of these in action

$ ls testdir

$ chmod 400 testdir

$ ls -ld testdir

$ cd testdir

$ ls testdir

$ chmod 100 testdir

$ ls -ld testdir

$ ls testdir

$ cd testdir

$ pwd

Note, on command 3rd and 7th above when we ran ls l included the -d option which stands for directory. Normally if we give ls an argument which is a directory it will list the contents of that directory. In this case however we are interested in the permissions of the directory directly and the -d option allows us to obtain that.

These permissions can seem a little confusing at first. What we need to remember is that these permissions are for the directory itself, not the files within. So, for example, you may have a directory which you don't have the read permission for. It may have files within it which you do have the read permission for. As long as you know the file exists and it's name you can still read the file.

$ ls -ld testdir

$ cd testdir

$ ls

ls: cannot open directory .: Permission denied

$ cat samplefile.txt

---The root user

On a Linux system there are only 2 people usually who may change the permissions of a file or directory. The owner of the file or directory and the root user. The root user is a superuser who is allowed to do anything and everything on the system. Typically the administrators of a system would be the only ones who have access to the root account and would use it to maintain the system. Typically normal users would mostly only have access to files and directories in their home directory and maybe a few others for the purposes of sharing and collaborating on work and this helps to maintain the security and stability of the system.

---Basic Security

Your home directory is your own personal space on the system. You should make sure that it stays that way.

Most users would give themselves full read, write and execute permissions for their home directory and no permissions for the group or others however some people for various reasons may have a slighly different set up.

Normally, for optimal security, you should not give either the group or others write access to your home directory, but execute without read can come in handy sometimes. This allows people to get into your home directory but not allow them to see what is there. An example of when this is used is for personal web pages.

It is typical for a system to run a webserver and allow users to each have their own web space. A common set up is that if you place a directory in your home directory called public\_html then the webserver will read and display the contents of it. The webserver runs as a different user to you however so by default will not have access to get in and read those files. This is a situation where it is necessary to grant execute on your home directory so that the webserver user may access the required resources.

-----Activities

Let's play with some permissions.

1. First off, take a look at the permissions of your home directory, then have a look at the permissions of various files in there.

2. Now let's go into your any directory of your choice and change the permissions of some of the files in there. Make sure you use both the shorthand and longhand form for setting permissions and that you also use a variety of absolute and relative paths. Try removing the read permission from a file then reading it. Or removing the write permission and then opening it in vi.

3. Let's play with directories now. Create a directory and put some files into it. Now play about with removing various permissions from yourself on that directory and see what you can and can't do.

4. Finally, have an explore around the system and see what the general permissions are for files in other system directories such as /etc and /bin