**HPCShiksha**



**Weak 1: 24th April - 30th April (Assignment)**

By

Mahendra Kumar Pankaj (HPC-TECH)

Emp Id: 346697

**Section 1: Linux commands:**

**01: Can you explain the 'cat' command in Linux and how to use it to concatenate and display files?**

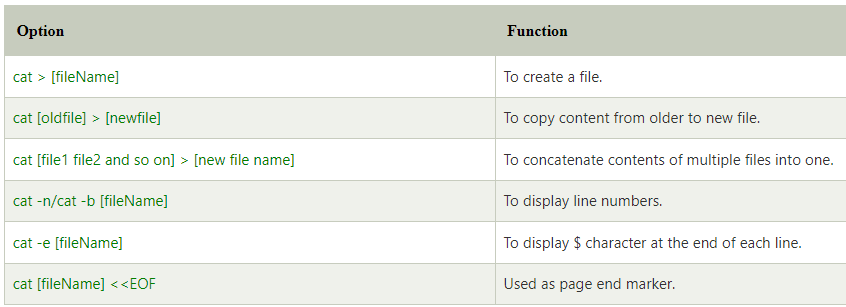
The 'cat' command is the most universal and powerful tool. It is considered to be one of the most frequently used commands. It can be used to display the content of a file, copy content from one file to another, concatenate the contents of multiple files, display the line number, display $ at the end of the line, etc.

Cat can be described as a standard Unix utility that sequentially reads files, writing them on the standard output. The name is derived from the function to catenate files. It has been shipped to several operating systems. This command is used to concatenate files which include multiple files into one file.

**Options in the cat Command:**

* **--show-all, -A:** It is the same as -vET.
* **--number-nonblank, -b:** It shows the total non-empty output lines. Also, it overrides -n.
* **-e:** It is the same as -vE.
* **--show-ends, -E:** It shows the $ symbol at the completion of all lines.
* **--number, -n:** It gives the total of every output line.
* **--squeeze-blank, -s:** It suppresses redundant empty output lines.
* **-t:** It is the same as -vT.
* **--show-tabs, -T:** It shows TAB characters as ^|.
* **-u:** ignored.
* **--show-nonprinting, -v:** It uses M- and ^ notation, except TAB and LFD.
* **--version:** It displays the information of the output version and exit.
* **--help:** It shows the help menu and exit.

**Usage of cat Command:**

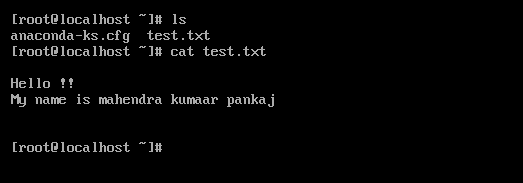


## Linux cat command: to display file content:

## The 'cat' command can be used to display the content of a file.

**Syntax**: cat <fileName>

#cat test.txt



## Linux cat command (to create a file):

## The 'cat' command can be used to create a new file with greater than sign ****(>)****.

## Press Cntl+D to save.

## Syntax: cat > <file\_name>

## #cat > test1.txt

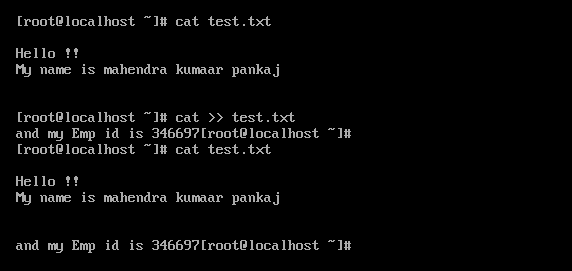
## 

## To Append the Content of A File:

## The 'cat' command with double greater than sign ****(>>)**** append (add something in the last of a file) something in your already existing file.

## Syntax: cat >> <file\_name>

## #cat >> test1.txt



**02: What is the 'grep' command in Linux and how do you use it to search for specific patterns in files and directories?**

The 'grep' command stands for **"global regular expression print"**. grep command filters the content of a file which makes our search easy.

It is a command-line utility to search plain-text data groups for lines that are the same as a regular expression. The name **"grep"** comes from the command, i.e., ed, which contains the same effect. Originally, grep was designed for the Unix operating system, but it became available for every Unix-like system later and a few others like OS 9.

## How to install grep in Linux?

Grep comes pre-installed in almost every distribution of Linux. However, in case, we can install it with the below command in the terminal window if it is missing from our system:

$ sudo apt-get install grep

**grep with pipe**

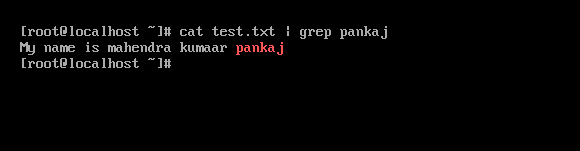
The 'grep' command is generally used with pipe **(|)**.

**Syntax:**

command | grep **<searchWord>**

**Example:**

#cat text.txt | grep pankaj



## grep without pipe:

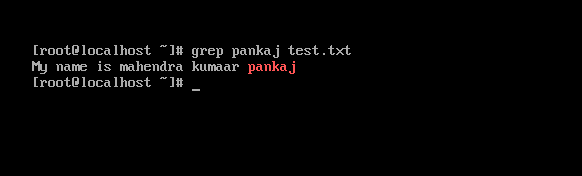
It can be used without pipe also.

**Syntax:**

grep **<searchWord>** **<file** name**>**

**Example:**

#grep pankaj test.txt



## grep options:

## 

**03: How do you use the 'sed' command in Linux to replace text in a file or a stream?**

Linux 'sed' command stands for **stream editor**. It is used to edit streams (files) using regular expressions. But this editing is not permanent. It remains only in display, but in actual, file content remains the same.

Primarily, it is used for text substitution; additionally, it can be used for other text manipulation operations like insert, delete, search, and more. The sed command allows us to edit files without opening them. Regular expression support makes it a more powerful text manipulation tool.

## Usage of sed Command

**Substitution command**

The below example represents the most command and typical use of the sed command, i.e., substitution. The usage was the actual motivation for the sed command:  
sed 's/regexp/replacement/g' inputFileName > outputFileName

**Other commands of sed**

Other ways are also possible for simple processing with some 25 sed commands. For instance, below uses the d option for filtering out lines that only include spaces and the line character end:  
sed '/^ \*$/d' inputFileName  
The above example uses a few of the below regular expression metacharacters:

* The caret (^) is the same as the starting of the line.
* The dollar symbol ($) is the same as the completion of the line.
* The asterisk (\*) is the same as the more or zero previous character occurrence.
* The plus symbol (+) is the same as the one or multiple previous character occurrences.
* The question mark (?) is the same as the more or zero previous character occurrence.
* The dot symbol (.) is exactly the same as one character.

**Sed as a filter**

Often, the sed command is used as the filter inside a pipeline under Unix:  
$ generateData | sed 's/x/y/g'  
The generateData program generates data, and the sed command makes a small change by substituting x in place of y.

**File-based sed scripts**

Often, it's helpful to put many sed commands, a single command per each line, inside a script file and use the -f flag to execute the commands from the file:  
sed -f subst.sed inputFileName > outputFileName

### **Syntax:**

sed [OPTION]... {script-only-if-no-other-script} [input-file]...

### **Options:**

The following are some command line options of the sed command:

**-n, --quiet, --silent:** It forcefully allows us to print of pattern space.

**-e script, --expression=script:** It is used to add the script to the commands to be executed.

**-f script-file, --file=script-file:** It is used to add the contents of script-file to the commands to be executed.

**--follow-symlinks:** it is used to follow symlinks when processing in place.

**-i[SUFFIX], --in-place[=SUFFIX]:** it is used to edit files in place (creates backup if SUFFIX option is supplied).

**-l N, --line-length=N:** It is used to specify the desired line-wrap length for the `l' command.

**--posix:** it is used to disable all GNU extensions.

**-E, -r, --regexp-extended:** It allows us to use the extended regular expressions in the script (for portability use POSIX -E).

**-s, --separate:** it is used for considering files as separate rather than as a single and continues the long stream.

**--sandbox:** It is used to operate in sandbox mode.

**-u, --unbuffered:** It is used for loading the minimal amounts of data from the input files and flushes the output buffers more often.

**-z, --null-data:** It is used to separate lines by NUL characters.

**--help:** it is used to display the help manual.

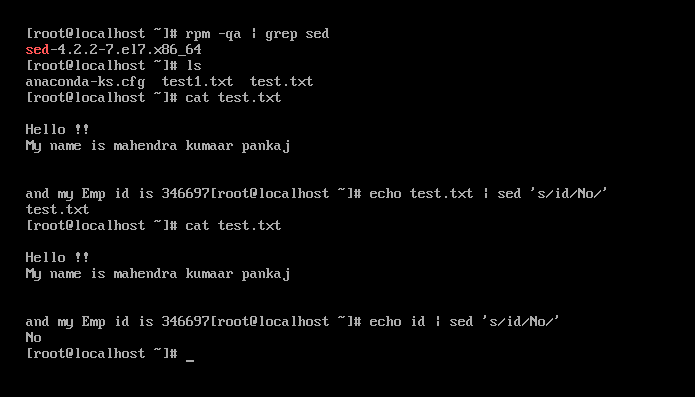
**--version:** It is used to display version information.

### **Applying to the STDIN directory:**

The sed command is not just limited to manipulate files; also, we can apply it to the STDIN directory.

# echo test.txt | sed ‘s/id/No/’

The above commands will replace the first text with the second text pattern.

****

### **Global Replacement:**

In the earlier example, all 'learn' words were not edited into 'study'. To edit every word, we have to use a global replacement 'g'. It will edit all the specified words in a file or string.

**Syntax:**

command | sed 's/**<oldWord>**/**<newWord>**/g'

### **Removing a Line:**

The 'd' option will let us remove a complete line from a file. We only need to specify a word from that line with 'd' option, and that line will be deleted. But, note that all the lines having that same word will be deleted. It will be executed as:

**Syntax:**

cat **<fileName>** | sed '/**<Word>**/d'

### **Replacing Characters:**

We can use the exclamation mark (!) as a string delimiter. For example, we want to replace bash shell and replace it with csh shell in the "/etc/passwd". To do so, execute the below command:

sed 's/\/bin\/bash/\/bin\/csh/' /etc/passwd

We can achieve the same result by executing the below command:

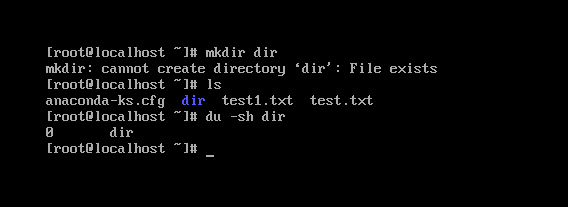
sed 's!/bin/bash!/bin/csh!' /etc/passwd

**04.Can you explain the difference between 'du' and 'df' commands in Linux and how to use them to check disk usage?**

The **du** and **df** commands are both used to check disk usage in Linux, but they provide different types of information.

The du command is used to estimate file space usage in a directory or a file hierarchy. It displays the size of the directory or file, as well as the sizes of its subdirectories and files, recursively. The du command is commonly used to determine which files and directories are taking up the most space on a file system.

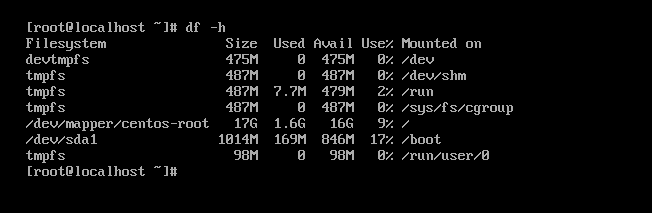
Here's an example of how to use the ‘**du**’ command to check disk usage:



In this example, **-s** tells **du** to display only a summary of the total disk usage, and -h tells it to display the sizes in a human-readable format (i.e., using units like "KB" or "MB"). /path/to/directory is the path to the directory you want to check.

The df command, on the other hand, is used to display the amount of disk space available on the file system. It shows the total size of the file system, the amount of space used, and the amount of space available.

Here's an example of how to use the **df** command to check disk usage:



NOTE: du is used to estimate file space usage within a directory or file hierarchy, while df is used to show the amount of disk space available on the file system.

**05: How do you use the 'mv' command to move multiple files and directories at once in Linux?**

To move multiple files and directories at once using the mv command in Linux, you can simply list all the files and directories you want to move, separated by spaces, followed by the destination directory.

Here's the basic syntax of the mv command:

**mv [OPTION]... [-T] SOURCE... DIRECTORY**

Here are the steps to move multiple files and directories using the mv command:

1. Open the terminal and navigate to the directory containing the files and directories you want to move.

2. Use the mv command followed by the list of files and directories you want to move, separated by spaces, and the destination directory. For example, if you want to move file1.txt, file2.txt, and directory1 to a new directory called myfiles, you can run the following command:

**mv file1.txt file2.txt directory1 myfiles/**

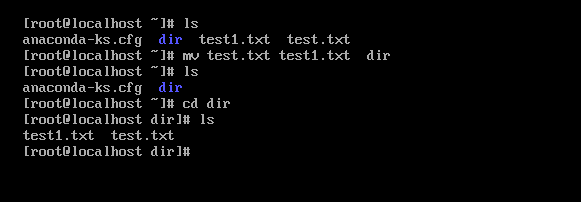
In this example, the mv command moves file1.txt, file2.txt, and directory1 to the myfiles directory.

3. Press Enter. The files and directories will be moved to the destination directory.

Note that you can also use wildcards to move multiple files at once. For example, if you want to move all files with a .txt extension to a new directory, you can run the following command:

**mv \*.txt myfiles/**

In this example, the mv command moves all files with a .txt extension to the myfiles directory.



**06.What command do you use to untar a file in Linux? Also, write a command to create a tar archive of an already present directory.**

To untar a file in Linux, you can use the tar command with the x option, which stands for "extract". Here's the basic syntax of the tar command to untar a file:

**#tar -xvf filename.tar**

In this syntax, filename.tar is the name of the tar archive file that you want to extract. The v option stands for "verbose" and will display the progress of the extraction process.

Here's an example command to untar a file named myarchive.tar:

**#tar -xvf myarchive.tar**

This command will extract the contents of myarchive.tar to the current directory.

To create a tar archive of an already present directory, you can use the tar command with the c option, which stands for "create". Here's the basic syntax of the tar command to create a tar archive:

**#tar -cvf filename.tar directory/**

In this syntax, filename.tar is the name you want to give to the tar archive, and directory is the name of the directory you want to archive. The v option stands for "verbose" and will display the progress of the archiving process.

Here's an example command to create a tar archive of a directory named mydir:

**#tar -cvf myarchive.tar mydir/**

This command will create a tar archive named myarchive.tar that contains the contents of the mydir directory.

**07. How do you use the 'head' and 'tail' commands in Linux to display the beginning and end of a file respectively?**

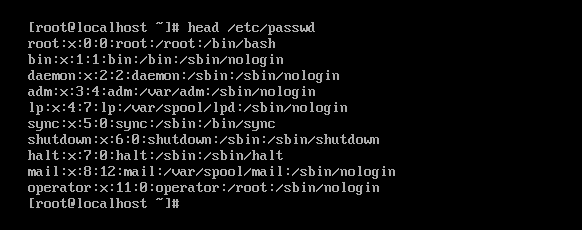
The **'head'** command displays the starting content of a file. By default, it displays starting 10 lines of any file.

**Syntax:**

head **<file** name**>**

**Example:**

#head /etc/passwd



If we'll write two file names then it will display first ten lines (in this case file has five lines only) of each file separated by a heading.

**Syntax:**

head **<file** name**>** **<file** name**>**

The 'head -n' option displays specified number of lines.

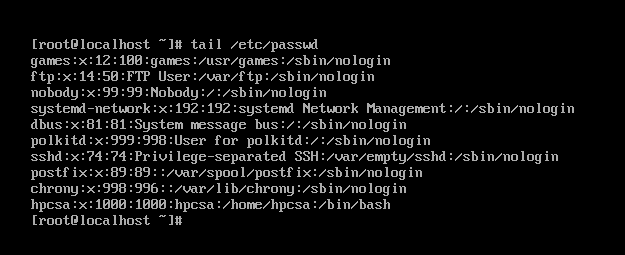
**Syntax:**

head -n **<file** name**>**

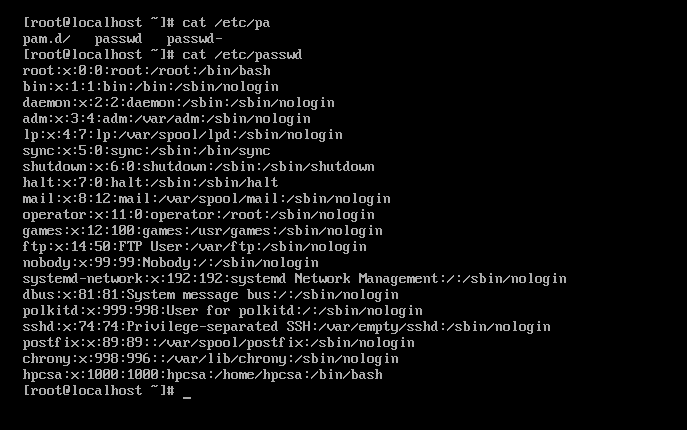
Linux tail command is used to display the last ten lines of one or more files. Its main purpose is to read the error message. By default, it displays the last ten lines of a file. Additionally, it is used to monitor the file changes in real-time. It is a complementary command of the [head command](https://www.javatpoint.com/linux-head).

### **Syntax:**

tail **<file** name**>**



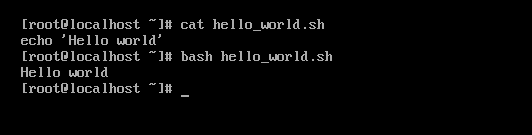
#cat /etc/passwd



**Section 2: Shell scripting:**

**01.Write a Linux shell script to print hello world.**

#vi hello\_world.sh



**02.** **Write a Linux shell script to show the usage of if, if else, and else if.**

Decision making is the most important aspect of almost all the programming languages. As the name implies, decision making allows us to run a particular block of code for a particular decision. Here, the decisions are made on the validity of the particular conditions. Condition checking is the backbone of decision making.

This statement is used when there is a need to check only conditions. If the condition founds to be true then the statement was written inside the if block will get executed.

**Syntax:**

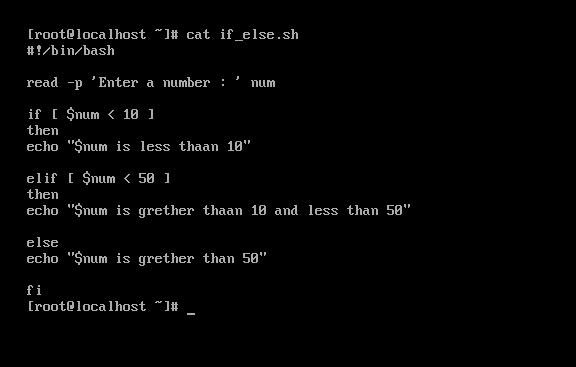
if (condition)

then

statement

fi

Code:



**03.Write a shell script program to demonstrate the use of a case statement.**

a is greater than 10

You can use multiple **if...elif** statements to perform a multiway branch. However, this is not always the best solution, especially when all of the branches depend on the value of a single variable.

Shell supports **case...esac** statement which handles exactly this situation, and it does so more efficiently than repeated if...elif statements.

## Syntax:

The basic syntax of the **case...esac** statement is to give an expression to evaluate and to execute several different statements based on the value of the expression.

The interpreter checks each case against the value of the expression until a match is found. If nothing matches, a default condition will be used.

case word in

pattern1)

Statement(s) to be executed if pattern1 matches

;;

pattern2)

Statement(s) to be executed if pattern2 matches

;;

pattern3)

Statement(s) to be executed if pattern3 matches

;;

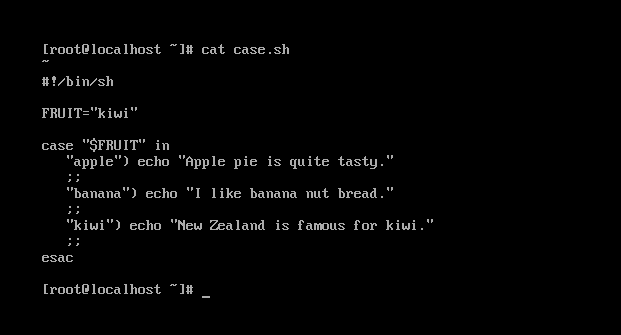
\*)

Default condition to be executed

;;

esac

**example:**

****

**04.Write a shell script program to demonstrate the usage of while and do-while loop.**

The **while** loop enables you to execute a set of commands repeatedly until some condition occurs. It is usually used when you need to manipulate the value of a variable repeatedly.

## Syntax

while command

do

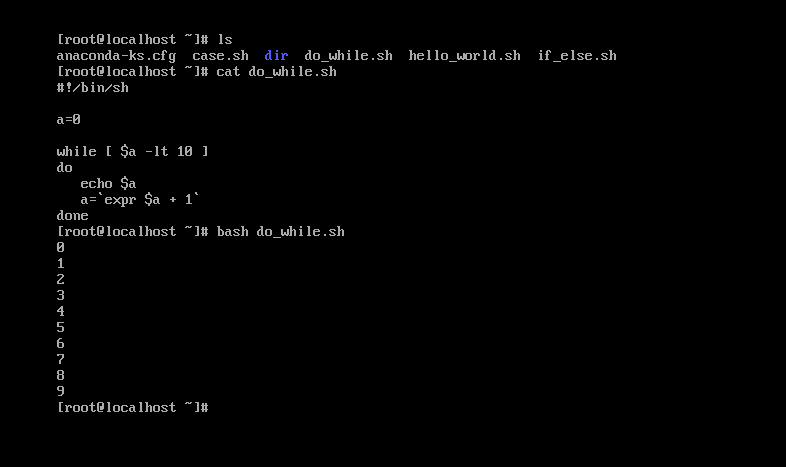
Statement(s) to be executed if command is true

done

Here the Shell *command* is evaluated. If the resulting value is *true*, given *statement(s)* are executed. If *command* is *false* then no statement will be executed and the program will jump to the next line after the done statement.

## Example

Here is a simple example that uses the **while** loop to display the numbers zero to nine –

****

**~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~**