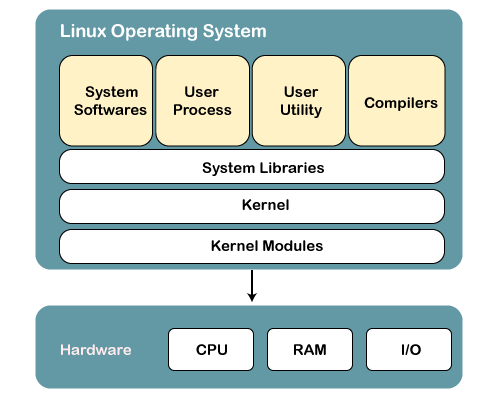
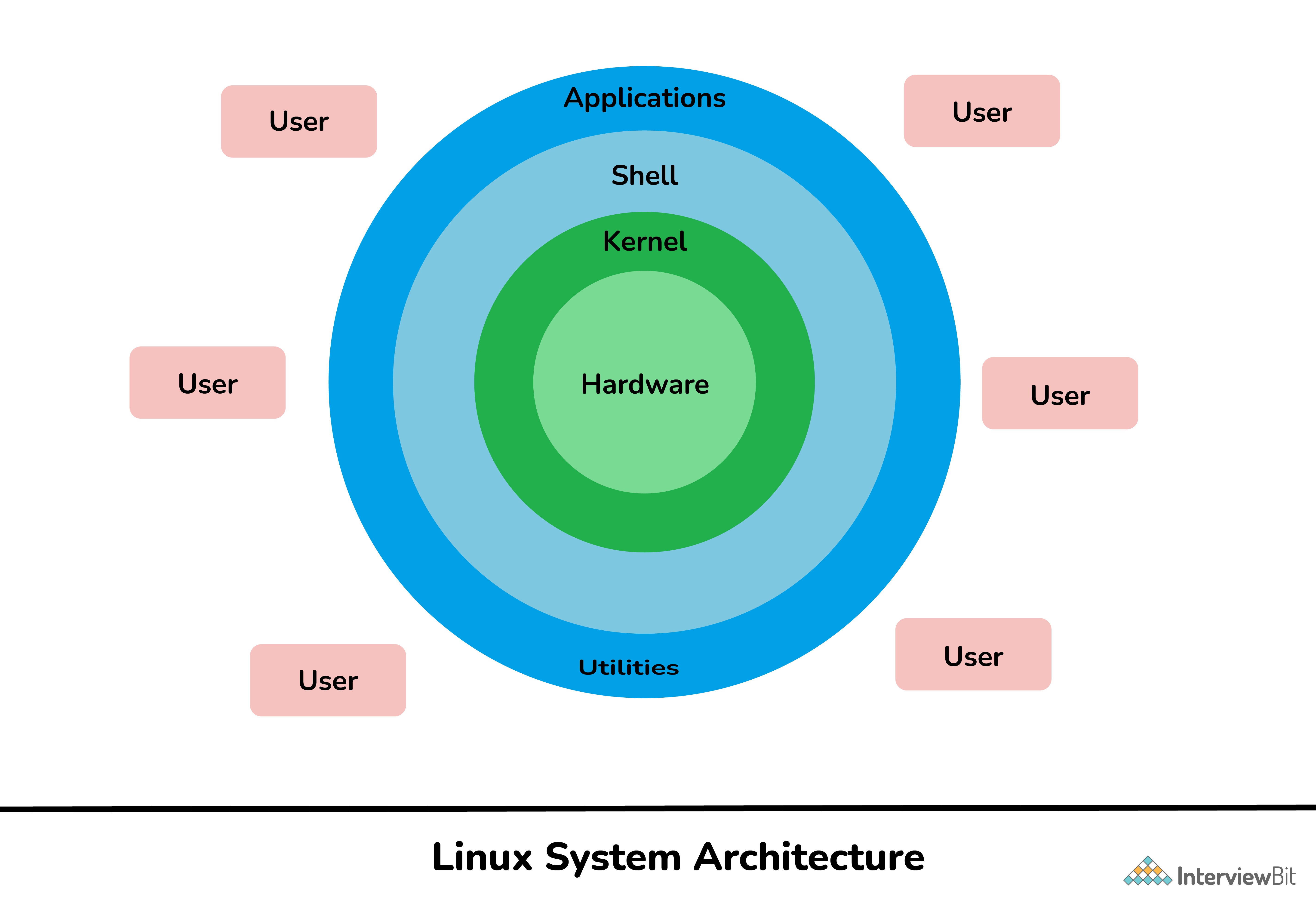
**LINUX**

Linux is an open-source operating system. It is like Windows, Mac, Android, etc. It is an commercial OS.

It consists of three parts: Kernal, Shell and Programs. n operating system is a collection of software, each designed for a specific function.

Linux OS has following components:



**Components of linux**

**Kernel:** It is considered a core or main part of Linux and is generally responsible for all major activities of OS such as process management, device management, etc.

**System Library:** These are special functions or programs with the help of which application programs or system utilities can access features of the kernel without any requirement of code. It is simply used to implement the functionality of the OS.

* **System Utility:** These are utility programs that are responsible to perform specialized and individual-level tasks. They are considered more liable and allow users to manage the computer.
* **Hardware:** It is physical hardware that includes items such as a mouse, keyboard, display, CPU, etc.
* **Shell:** It is an environment in which we can run our commands, shell scripts, and programs. It is an interface between user and kernel that hides all complexities of functions of the kernel from the user. It is used to execure commands

The terms “**terminal,” “shell,” and “command line interface”** are often used interchangeably, but there are subtle differences between them:

* A *terminal* is an input and output environment that presents a text-only window running a shell.
* A *shell* is a program that exposes the computer’s operating system to a user or program. In Linux systems, the shell presented in a terminal is a command line interpreter.
* A *command line interface* is a user interface (managed by a command line interpreter program) which processes commands to a computer program and outputs the results.

When someone refers to one of these three terms in the context of Linux, they generally mean a terminal environment where you can run commands and see the results printed out to the terminal, such as this:

**Basic commands:**

* 1. **cd command – for change directory, ( $**cd *filename or dir.name*) as o/p we get enter to given file or directory in command.
  2. **Pwd command – present working directory, it will show the location and path as well of current working or logged in directory.**
  3. **ls command – for listing the stuff ($** ls {*option} {directoryname*})

|  |  |
| --- | --- |
| **ls option** | **Description** |
| [ls -a](https://www.javatpoint.com/linux-ls#linux-ls-a) | The (ls -a) command will enlist the whole list of the current directory including the hidden files. |
| [ls -l](https://www.javatpoint.com/linux-ls#linux-ls-l) | It will show the list in a long list format. |
| ls -lh | This command will show you the file sizes in human readable format. Size of the file is very difficult to read when displayed in terms of byte. The (ls -lh)command will give you the data in terms of Mb, Gb, Tb, etc. |
| ls -lhS | If you want to display your files in descending order (highest at the top) according to their size, then you can use (ls -lhS) command. |
| [ls -l - -block-size=[SIZE]](https://www.javatpoint.com/linux-ls#linux-ls-l-block-size) | It is used to display the files in a specific size format. Here, in [SIZE] you can assign size according to your requirement. |
| [ls -d \*/](https://www.javatpoint.com/linux-ls#linux-ls-d-asterisk-slash) | It is used to display only subdirectories. |
| [ls -g or ls -lG](https://www.javatpoint.com/linux-ls#linux-ls-g) | With this you can exclude column of group information and owner. |
| ls -n | It is used to print group ID and owner ID instead of their names. |
| [ls --color=[VALUE]](https://www.javatpoint.com/linux-ls#linux-ls-color) | This command is used to print list as colored or discolored. |
| ls -li | This command prints the index number if file is in the first column. |
| ls -p | It is used to identify the directory easily by marking the directories with a slash (/) line sign. |
| ls -r | It is used to print the list in reverse order. |
| ls -R | It will display the content of the sub-directories also. |
| ls -lX | It will group the files with same extensions together in the list. |
| ls -lt | It will sort the list by displaying recently modified filed at top. |
| [ls ~](https://www.javatpoint.com/linux-ls#linux-ls-tilde) | It gives the contents of home directory. |
| [ls ../](https://www.javatpoint.com/linux-ls#linux-ls-dot-dot-slash) | It give the contents of parent directory. |
| ls --version | It checks the version of ls command. |

* 1. **mkdir : for create new directory (** mkdir *{name1} {name2}....*)
  2. **rm :** to remove objects such as files, directories. ( rm {optionS} {file/directory name} )

**now remove subdirectory also so** ( rmdir **-R** {file/directory name}

**File commands**

**1.cat command: to read file**

|  |  |
| --- | --- |
| **Command** | **Description** |
| [file](https://www.javatpoint.com/linux-file) {filename1} | Determines file type of filename1 |
| [touch](https://www.javatpoint.com/linux-touch) {filename1} | Used to create a filename1 |
| [rm](https://www.javatpoint.com/linux-rm) {filename1} | To remove a file., powerful option **–R to remove subdirectories also** |
| [cp](https://www.javatpoint.com/linux-cp) {filename1 filename2} | To copy a file from filename1 to filename2 |
| [mv](https://www.javatpoint.com/linux-mv) {filename1 filename2} | To rename or to move a file from filename1 to filename2 |
| less filename1 | view the contents of a long file |
| cat filename1 | To view content of file |

**Chmod command :** the **chmod** command is used to change the access mode of a file.

u owner file's owner

g group users who are members of

the file's group

o others users who are neither the

file's owner nor members of

the file's group

a all All three of the above, same as ugo

**r=4, w=2, x=1, none(-)=0**

**chown** **command is used to change the file Owner**

**chgrp command is used to change the file group**

**CuT command**

cutting out the sections from each line of files and writing the result to standard output. It can be used to cut parts of a line by **byte position, character and field**. Basically the cut command slices a line and extracts the text. It

**Syntax:**

**cut OPTION... [FILE]...**

.**$ cut state.txt**

Try 'cut --help' for more information.

**Options and their Description with examples:**

**1. -b(byte):** To extract the specific bytes,

**List without ranges**

**$ cut -b 1,2,3 state.txt**

And

Aru

Ass

Bih

**List with ranges**

**$ cut -b 1-3,5-7 state.txt**

Andra

Aruach

Assm

Bihr

It uses a special form for selecting bytes from beginning upto the end of the line:

**$ cut -b 1- state.txt #** In this, 1- indicate from 1st byte to end byte of a line

Andhra Pradesh

Arunachal Pradesh

Assam

Bihar

Chhattisgarh

**$ cut -b -3 state.txt #** In this, -3 indicate from 1st byte to 3rd byte of a line

And

Aru

Ass

Bih

Chh

**2. -c (column):** To cut by character use the -c option. **Syntax:**

**$ cut -c 2,5,7 state.txt**

nr

rah

sm

ir

hti

Above cut command prints second, fifth and seventh character from each line of the file.

**$ cut -c 1-7 state.txt**.

Cut uses a special form for selecting characters from beginning upto the end of the line:

**$ cut -c 1- state.txt** Cut uses a special form for selecting characters from beginning upto the end of the line:

Andhra Pradesh

Arunachal Pradesh

Assam

**$ cut -c -5 state.txt #** Above command prints starting position to the fifth character.

Andhr

Aruna

Assam

**3. -f (field):**  To extract the useful information you need to cut by fields rather than columns. Ranges are not described with -f option. **cut**uses **tab**as a default field delimiter but can also work with other delimiter by using **-d** option.  
**Note:** Space is not considered as delimiter in UNIX.

**Syntax:**

**$cut -d "delimiter" -f (field number) file.txt**

**$ cut -f 1 state.txt**

Andhra Pradesh

Arunachal Pradesh

Assam

Bihar

Chhattisgarh

If -d option is used then it considered space as a field separator or delimiter:

**$ cut -d " " -f 1 state.txt**

Andhra

Arunachal

Assam

Bihar

Chhattisgarh

Command prints field from first to fourth of each line from the file.

**Command:**

$ cut -d " " -f 1-4 state.txt

**Output:**

Andhra Pradesh

Arunachal Pradesh

Assam

Bihar

Chhattisgarh

**4. –complement:** As the name suggests it complement the output. This option can be used in the combination with other options either with **-f** or with **-c**.

**$ cut --complement -d " " -f 1 state.txt**

Pradesh

Pradesh

Assam

**$ cut --complement -c 5 state.txt**

Andha Pradesh

Arunchal Pradesh

Assa**5. –output-delimiter:** By default the output delimiter is same as input delimiter that we specify in the cut with **-d** option. To change the output delimiter use the option **–output-delimiter=”delimiter”**.

**$ cut -d " " -f 1,2 state.txt --output-delimiter='%'**

Andhra%Pradesh

Arunachal%Pradesh

Assam

Bihar

Chhattisgarh

Here cut command changes delimiter(%) in the standard output between the fields which is specified by using -f option ..

**Applications of cut Command**

**$ cat state.txt | head -n 3 | cut -d ' ' -f 1 > list.txt**

**$ cat list.txt**

Andhra

Arunachal

**SYSTEM**

1. **du** command, short for disk usage, is used to estimate file space usage.  
   The du command can be used to track the files and directories which are consuming excessive amount of space on hard disk drive.

**du *Path # path k ander jitni file hogi sbki space usage bta degi***

Examples :

du /home/mandeep/test

Output:

44 /home/mandeep/test/data

2012 /home/mandeep/test/system design

24 /home/mandeep/test/table/sample\_table/tree

28 /home/mandeep/test/table/sample\_table

32 /home/mandeep/test/table

100104 /home/mandeep/test

***-0, –null :****end each output line with NULL****-a, –all :****write count of all files, not just directories****–apparent-size :****print apparent sizes, rather than disk usage.****-B, –block-size=SIZE :****scale sizes to SIZE before printing on console****-c, –total :****produce grand total****-d, –max-depth=N :****print total for directory only if it is N or fewer levels below command line argument****-h, –human-readable :****print sizes in human readable format****-S, -separate-dirs :****for directories, don’t include size of subdirectories****-s, –summarize :****display only total for each directory****–time :****show time of last modification of any file or directory.****–exclude=PATTERN :****exclude files that match PATTERN*

1. The **df** command (short for disk free), is used to display information related to file systems about total space and available space.

**df :** df [OPTION]... [FILE]...or path

Output :

Filesystem 1K-blocks Used Available Use% Mounted on

udev 3996816 0 3996816 0% /dev

tmpfs 804624 10020 794604 2% /ru/dev/sda9 68117056 18036160 46597712 28% /

tmpfs 4023116 29848 3993268 1% /dev/shm

tmpfs 5120 4 5116 1% /run/lock

tmpfs 4023116 0 4023116 0% /sys/fs/cgroup

***-a, –all :* includes pseudo, duplicate and inaccessible file systems.  
*-B, –block-size=SIZE :*scales sizes by SIZE before printing them.  
*-h, –human-readable :* print sizes in power of 1024  
*-H, –si:*print sizes in power of 1000  
*-i, –inodes :* list inode information instead of block usage  
*-l, –local :* limit listing to local file systems  
*-P, –portability :* use POSIX output format  
*–sync :* invoke sync before getting usage info  
*–total :* elide all entries insignificant to available space, and produce grand total  
*-t, –type=TYPE :* limit listing to file systems of type TYPE  
*-T, –print-type :* print file system type**

**Fdisk**

**fdisk** also known as format disk is a dialog-driven command in Linux used for creating and manipulating disk partition table. It is used for the view, create, delete, change, resize, copy and move partitions on a hard drive using the dialog-driven interface.

fdisk -l [device...]

**PROCESS**

1. **PS command - “Process Status”.** ps command is used to list the currently running processes and their PIDs along with some other information depends on different options. It reads the process information from the virtual files in **/proc** file-system. /proc contains virtual files, this is the reason it’s referred as a virtual file system.

**Simple process selection :** Shows the processes for the current shell – 

[root@rhel7 ~]# ps

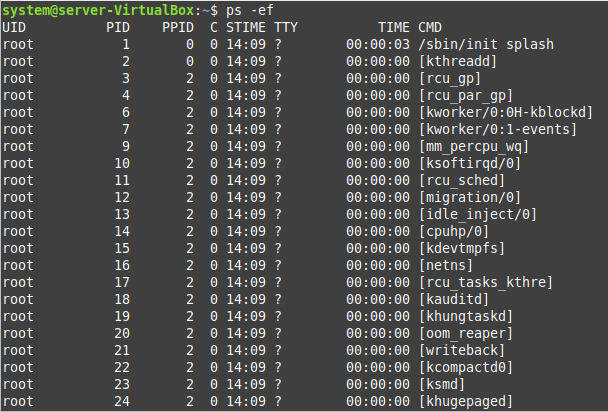
PID TTY TIME CMD

12330 pts/0 00:00:00 bash

21621 pts/0 00:00:00 ps

[root@rhel7 ~]# ps –ef \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

This command displayed all the processes running on the system in the standard format as shown in the image below:



KILL command – k**ill command sends a signal to a process which terminates the process**

**1. kill -l :**To display all the available signals you can use below command option:

**Syntax:**

$kill –l

In which -9 for sig kill and -15 for terminate

**2. kill pid :** To show how to use a PID with the kill command.

**Syntax:**

$kill pid

**3. kill -s :** To show how to send signal to processes.

**Syntax:**

kill {-signal | -s signal} pid

**4. kill -L :**This command is used to list available signals in a table format.

**Syntax:**

kill {-l | --list[=signal] | -L | --table}

**ZOMBIE PROCESS**

A zombie process is a process whose execution is completed but it still has an entry in the process table. Zombie processes usually occur for child processes, as the parent process still needs to read its child’s exit status. Once this is done using the wait system call, the zombie process is eliminated from the process table. This is known as reaping the zombie process.

A diagram that demonstrates the creation and termination of a zombie process is given as follows −

Salient points of Zombie Processes

Some of the salient points related to zombie processes are as follows −

* All the memory and resources allocated to a process are deallocated when the process terminates using the exit() system call. But the process’s entry in the process table is still available. This process is now a zombie process.
* The exit status of the zombie process can be read by the parent process using the wait() system call. After that, the zombie process is removed from the system. Then the process ID and the process table entry of the zombie process can be reused.
* If the parent process does not use the wait() system call, the zombie process is left in the process table. This creates a resource leak.
* If the parent process is not running anymore, then the presence of a zombie process indicates an operating system bug. This may not be a serious problem if there are a few zombie processes but under heavier loads, this can create issues for the system such as running out of process table entries.
* The zombie processes can be removed from the system by sending the SIGCHLD signal to the parent, using the kill command. If the zombie process is still not eliminated from the process table by the parent process, then the parent process is terminated if that is acceptable.

Dangers of Zombie Processes

Zombie processes don't use any system resources but they do retain their process ID. If there are a lot of zombie processes, then all the available process ID’s are monopolized by them. This prevents other processes from running as there are no process ID’s available.

The presence of zombie processes also indicates an operating system bug if their parent processes are not running anymore. This is not a serious problem if there are a few zombie processes but under heavier loads, this can create issues for the system.

Killing Zombie Processes

Zombie processes can be killed by sending the SIGCHLD signal to the parent, using the kill command. This signal informs the parent process to clean up the zombie process using the wait() system call. This signal is sent with the kill command.

kill -s SIGCHLD *pid*

grep stands for global search for regular expression and print out).**Pattern search**

**grep [options] pattern [files]**

**Options Description**

**-c** : This prints only a count of the lines that match a pattern

**-h :** Display the matched lines, but do not display the filenames.

**-i :** Ignores, case for matching

**-l :** Displays list of a filenames only.

**-n :** Display the matched lines and their line numbers.

**-v :** This prints out all the lines that do not matches the pattern

**-e exp :** Specifies expression with this option. Can use multiple times.

**-f file :** Takes patterns from file, one per line.

**-E :** Treats pattern as an extended regular expression (ERE)

**-w :** Match whole word

**SED command – ONLY APPLICABLE FILES DATA—**

SED command in UNIX stands for stream editor and it can perform lots of functions on file like searching, find and replace, insertion or deletion. Though most common use of SED command in UNIX is for substitution or for find and replace.

**Replacing or substituting string :** Sed command is mostly used to replace the text in a file. The below simple sed command replaces the word “unix” with “linux” in the file.

**$sed 's/unix/linux/' geekfile.txt- replace only a first occurance in each line**

**Output :**

linux is great os. unix is opensource. unix is free os.

learn operating system.

linux linux which one you choose.

linux is easy to learn.unix is a multiuser os.Learn unix .unix is a pow

**Replacing the nth occurrence of a pattern in a line :** Use the /1, /2 etc flags to replace the first, second occurrence of a pattern in a line. The below command replaces the second occurrence of the word “unix” with “linux” in a line.

**$sed 's/unix/linux/2' geekfile.txt**

flag /g (global replacement) replace all words to linux

**$sed 's/unix/linux/g' geekfile.txt**

**$sed 's/unix/linux/p' geekfile.txt #repalce 1st occureance in a line & create duplicate also**

**$sed '1,3 s/unix/linux/' geekfile.txt -** replace only 1 occurance word unix to linux **1st to 3rd**lines

**$sed '2,$ s/unix/linux/' geekfile.txt-** replace only 1 occurance word unix to linux **1st to last ($)**lines

$ sed 'nd' filename.txt - To Delete a particular line say n in this example

$ sed '/pattern/d' filename.txt - To Delete pattern matching line

**Shell script** <https://www.javatpoint.com/for-loop-shell-scripting>

**SHELL**

This **command** passes to the shell which reads the command and execute it. Shell is a method for the user to interact with the system. Default shell in Linux is called bash (Bourne-Again Shell).

There are two types of shell commandsC++ vs Java

* **Built-in shell commands:** They are part of a shell. Each shell has some built in commands.
* **External/Linux commands:** Each external command is a separate executable program written in C or other programming languages.

**#!/bin/bash (USED initially to prepare new script) script should be .sh extension**

1. **Read argument**

**2.for loop:**

**AIM to print 1 to 5 natural no.**

#!/bin/usr/env bash

for n in {1..5}; #in for loop curly brackets are used

do

echo $n

done

**If else loop**

1. #!/bin/sh
2. a=10
3. b=20
4. if [ $a == $b ]
5. then
6. echo "a is equal to b"
7. else
8. echo "a is not equal to b"
9. fi

**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 −

#!/bin/sh

a=0

while [ $a -lt 10 ]

do

echo $a

a=`expr $a + 1`

done

Upon execution, you will receive the following result −

0

1

2

3

4

5

6

7

8

9

Arithmatic operation in linux shell

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| + (Addition) | Adds values on either side of the operator | `expr $a + $b` will give 30 |
| - (Subtraction) | Subtracts right hand operand from left hand operand | `expr $a - $b` will give -10 |
| \* (Multiplication) | Multiplies values on either side of the operator | `expr $a \\* $b` will give 200 |
| / (Division) | Divides left hand operand by right hand operand | `expr $b / $a` will give 2 |
| % (Modulus) | Divides left hand operand by right hand operand and returns remainder | `expr $b % $a` will give 0 |
| = (Assignment) | Assigns right operand in left operand | a = $b would assign value of b into a |
| == (Equality) | Compares two numbers, if both are same then returns true. | [ $a == $b ] would return false. |
| != (Not Equality) | Compares two numbers, if both are different then returns true. | [ $a != $b ] would return true. |

# Crontab- Schedules commands execution at Specified time or time interval.

MIN HOUR DOM MON DOW CMD

Crontab Options

To Install or update job in crontab, use -e option:

$ crontab -e

To List Crontab entries, use -l option:

$ crontab -l

To Deinstall job from crontab, use -r option:

$ crontab -r

To Confirm Deinstall of job from crontab, use -i option:

$ crontab -i -r

To add SELINUX security to crontab file, use -s option:

$ crontab -s

To edit other user crontab, user -u option and specify username:

$ crontab -u username -e

To List other user crontab entries:

$ crontab -u username –l

To run sample.sh every Sunday and Monday.

0 4,17 \* \* sun,mon sample.sh

To run sample.sh every 30 Seconds.

\* \* \* \* \* sample.sh

\* \* \* \* \* sleep 30; sample.sh

To run multiple jobs using single cron.

\* \* \* \* \* sample1.sh; sample2.sh

s

To run sample.sh on yearly ( @yearly ).

@yearly sample.sh

To run sample.sh on monthly ( @monthly ).

@monthly sample.sh

To run sample.sh on Weekly ( @weekly ).

@weekly sample.sh

To run sample.sh on daily ( @daily ).

@daily sample.sh

To run sample.sh on hourly ( @hourly ).

@hourly sample.sh

To run sample.sh on system reboot ( @reboot ).

@reboot sample.sh

**Options :**

* **-exec CMD:** The file being searched which meets the above criteria and returns 0 for as its exit status for successful command execution.
* **-ok CMD :** It works same as -exec except the user is prompted first.
* **-inum N :** Search for files with inode number ‘N’.
* **-links N :** Search for files with ‘N’ links.
* **-name demo :** Search for files that are specified by ‘demo’.
* **-newer file :** Search for files that were modified/created after ‘file’.
* **-perm octal :** Search for the file if permission is ‘octal’.
* **-print :** Display the path name of the files found by using the rest of the criteria.
* **-empty :** Search for empty files and directories.
* **-size +N/-N :** Search for files of ‘N’ blocks; ‘N’ followed by ‘c’can be used to measure size in characters; ‘+N’ means size > ‘N’ blocks and ‘-N’ means size < ‘N’ blocks.
* **-user name :** Search for files owned by user name or ID ‘name’.
* **\(expr \) :** True if ‘expr’ is true; used for grouping criteria combined with OR or AND.
* **! expr :** True if ‘expr’ is false.

**1. Search a file with specific name.**

$ find ./GFG -name sample.txt

It will search for sample.txt in GFG directory.

**Output :**

**2. Search a file with pattern.**

$ find ./GFG -name \*.txt

It will give all files which have ‘.txt’ at the end.

**Output :**

**3. How to find and delete a file with confirmation.**

$ find ./GFG -name sample.txt -exec rm -i {} \;

When this command is entered, a prompt will come for confirmation, if you want to delete sample.txt or not. if you enter ‘Y/y’ it will delete the file. **Output :**

**4. Search for empty files and directories.**

$ find ./GFG -empty

This command find all empty folders and files in the entered directory or sub-directories.

**Output :**

**5. Search for file with entered permissions.**

$ find ./GFG -perm 664

This command find all the files in the GFG directory or sub-directory with the given permissions.

**Output :**

**6. Search text within multiple files.**

$ find ./ -type f -name "\*.txt" -exec grep 'Geek' {} \;

This command print lines which have ‘Geek’ in them and ‘-type f’ specifies the input type is a file.

**Output :**