UNIX Tutorial

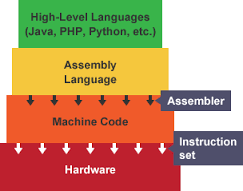
[Machine code](https://en.wikipedia.org/wiki/Machine_code): is the only language a computer can process directly without a previous transformation. Currently, programmers almost never write programs directly in machine code, because it requires attention to numerous details that a high-level language handles automatically.

Machine code is completely in 1’s and 0’s.

Assembly Languages: Assembly language is one level above machine language. It uses short mnemonic codes for instructions and allows the programmer to introduce names for blocks of memory that hold data.

Code is written with the help of Assemblers and mnemonic’s.

High Level Language: A programming language with strong [abstraction](https://en.wikipedia.org/wiki/Abstraction_(computer_science)) from details of the computer.



Programming: is basically divided into two types.

1. System Programming: To build operating system and its utilities and device drivers.
2. Application Programming: To write application software like desktop applications, banking applications, web applications using Java and .net.

Unix: is a [multitasking](https://en.wikipedia.org/wiki/Computer_multitasking), [multiuser](https://en.wikipedia.org/wiki/Multiuser) computer [operating systems](https://en.wikipedia.org/wiki/Operating_system) that was written in C-Language. Kenneth Thompson, Dennis Ritchie developed at [AT&T](https://en.wikipedia.org/wiki/AT%26T_Corporation) and Bell Labs.

UNIX Operating System: are meant for commercial usage across the enterprises and the license is chargeable.

Below are the commercial distributors who developed their own UNIX Flavor from the original Unix edition developed by AT&T and Bell Labs in 1970’s. Enterprises have to pay for original Unix source code to develop their own editions.

Enterprise Editions:

1. HP-UX
2. IBM AIX
3. Sun Solaris
4. Mac OS X
5. IRIX

Linux / Linux Kernel: is a Unix Like operating system which is completely written from scratch by Linus Torvalds. Linux doesn’t use the original Unix edition code but we can say it is exact clone of Unix.

Linux was written primarily in “C” and “Assembly languages”.

As Unix Versions are Licensed and works on specific hardware hence Linux was developed for freeware and can run on most of the hardware’s.

There are many Linux distributers who have developed different Linux Flavors by taking the Linux kernel source code for free of cost. This distributers have both free OS and enterprise editions with minimal license cost.

Kernel Source code download: <https://www.kernel.org/>

Bash is the default shell for Linux OS.

Linux Kernel Editions:

1. Redhat Enterprise Linux (Enterprise Edition from Redhat)
2. Ubuntu (Free OS & Enterprise Editions)
3. Cent OS (Free OS from Redhat)
4. Fedora Linux
5. Debian Linux

Difference between Unix and Linux:

|  |  |
| --- | --- |
| Linux Kernel OS | Unix OS |
| The Source Code of Linux is freely available to its Users. | The Source Code of Unix is not available for the general public. |
| Linux primarily uses Graphical User Interface with an optional Command Line Interface. | Unix primarily uses Command Line Interface. |
| Written Primarily in C and Assembly Language | Written in C Language. |
| Linux OS is portable and can be executed in different Hard Drives. | Unix is not portable. |
| Linux is very flexible and can be installed on most of the Home Based Pcs. | Unix has a rigid requirement of the Hardware. Hence, cannot be installed on every other machine. |
| Linux is mainly used in Home Based PC, Mobile Phones, Desktops, etc. | Unix is mainly used in Server Systems, Mainframes and High End Computers. |
| Different Versions of Linux are: Ubuntu, Debian, OpenSuse, Redhat, Solaris, etc. | Different Versions of Unix are: AIS, HP-UX, BSD, Iris, etc. |
| Linux Installation is economical and doesn’t require much specific and high end hardware. | Unix Installation is comparatively costlier as it requires more specific hardware circuitry. |
| The Filesystems supported by Linux are as follows: xfs, ramfs, nfs, vfat, cramfsm ext3, ext4, ext2, ext1, ufs, autofs, devpts, ntfs | The Filesystems supported by Unix are as follows: zfs, js, hfx, gps, xfs, gps, xfs, vxfs. |
| Linux is development by an active Linux Community worldwide. | Unix is developed by AT&T Developers. |

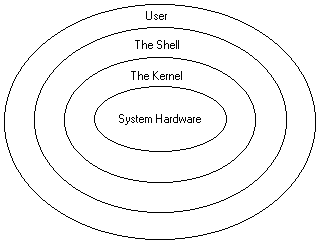
Shell:

Shell is an interface between user and the kernel. Even though there can be only one kernel; A system can have many shell running simultaneously. Whenever a user enters a command through keyboard the shell communicates with the kernel to execute it and then display the output to the user.

Kernel:

The kernel is the heart of the operating system. It interacts with the hardware and most of the tasks like memory management, task scheduling and file management.

Kernel acts as an intermediary between System hardware and Shell/Applications/Programs.



Kernel Tasks:

1. I/O management
2. Process management
3. Device management
4. File management
5. Memory management

Different Types of Shells to interact with Kernel:

csh, ksh, bash, Bourne .

The most commonly used and advanced shell used today is “Bash”.

1. The Bourne Shell (sh)
2. The C Shell (csh or tsch)
3. The Bourne Against Shell (bash)
4. The Korn Shell (ksh)

Default Shell:

To check the default shell type the below command

View all the Available Shells: **cat /etc/shells**

echo $SHELL

Whenever a user login into the Unix server the default shell initialization script will executed.

Below are the default initialization scripts with their respective shells:

1. **.profile** − The Bourne shell ( sh) initialization script
2. .**bashrc** - The Bash shell ( bash) initialization script
3. **.kshrc** − The Korn shell ( ksh) initialization script
4. **.cshrc** − The C shell ( csh) initialization script
5. **.rhosts** − The remote shell configuration file

ctrl+z or ctrl+c : Come out of the running process in unix.

users: will show the list of existing users in the unix server.

For every user created in the UNIX server will have an id and group id.

Id- will display the user id and group id’s that the user belongs.



set –x :

Is used to debug the script and to check the execution of the script. Enabling debugging with "set -x" also shows each line of execution preceded by a plus sign (+).

Ref: Refer question 9 for the example.

<http://www.livefirelabs.com/unix_tip_trick_shell_script/unix_shell_scripting/20-unix-shell-scripting-interview-questions-and-answers-part-1.htm>

/ (Root path of the UNIX)

Creating user defined command. (Creating shell script filename.sh is nothing but user defined command)

inode: inode is a numeric value assigned to every file/dir which is created in the Unix/Linux file system.

Basic Commands

uname: Displays what kind of OS(Linux,AIX,Solaris…) the host is using.

*-r: Displays kernel version*

*-a: All the info about the machine*

uptime: Tells, how long the server has been running.

logname: Displays the current username.

exit: To logout from the current user.

clear: Clear the screen

tty: Display the current terminal name (multiple users connected to host, tty will display ur terminal)

su username(switch user): to switch from one user account to other user account.(Provide the password to switch to other account)

finger username: it gives complete information about user.

which cmnd\_name: Displays the location of given command.

whereis cmnd\_name: Display the path of the command with its binaries

history : displays the history of the shell prompt.

Home Directory

cd ~: move to home dir of the logged user.

cd ~username : Will switch to other users home directory.

cd .. : Move to parent dir.

cd .. / .. : move to grandparent dir.

cd - : will go back to the last accessed directory

Working with directories and Path

Path: The position of any file within the hierarchy is described by its pathname.

Absolute Path: A pathname is absolute, if it is described in relation to root, thus absolute pathnames always begin with a **/**.

Example:

/etc/passwd

/users/sjones/chem/notes

/dev/rdsk/Os3

pwd: It will display the current path of the directory(Present working directory)

mkdir dirname: creates single dir or multiple dir’s.

mkdir –p dirname: Will create the required parent or child directories where ever required.

EX: mkdir -p /home/hirwcourseuser0720/kirandir/test1/test2

mkdir xyz abc kiran : (creating multiple directories)

rmdir works only with empty directories only. To delete non empty directories we have to use below command.

rm –R dirname: Will delete the empty and non-empty directories recursively.

du -k -s directory : Get directory size

soft link(ln -s): Can be used to create a link which points to actual file. Soft links preceded with ‘l’

Syntax: ln -s {/path/to/file-name} {link-name}

ln –s /home/dir1/file1.txt link1 # Execute this command in required path.

ls –il # -i will display the inodes of all the files, for soft link inodes will be same.

rm link1 # Deleting the soft link.

1. Soft Links have different inodes numbers.  
2. ls -il command shows all links with second column value 1 and the link points to original file.  
3. Soft Link contains the path for original file and not the contents.  
4. Removing soft link doesn't affect anything but when the original file is removed, the link becomes a 'dangling' link that points to nonexistent file.  
5. A Soft Link can link to a directory.

hard link (ln): When a hard link is created for a file the link is exact copy for the actual file and having inodes same for hard link and actual file.

Syntax: ln {/path/to/file-name} {link-name}

Hard links are not preceded with any think they look exactly like files.

**1.** Hard Links have same inodes number.  
**2.** ls -il command shows all the links with the link column showing the number of links.  
**3.** Links have actual file contents  
**4.** Removing any link, just reduces the link count but doesn't affect the other links.  
**5.** You cannot create a Hard Link for a directory.  
**6.** Even if the original file is removed, the link will still show you the contents of the file.

Hard Link vs copy file:

copy:   
    - you have two different versions of the file.   
   - if you edit one, the other one stays the same.  
   - if you delete one, the other one stays there, but it may not be identical if it was edited  
   - twice as much disk space used (two different files)  
hard link:   
    - you have one file with two different filenames.   
    - If you edit one, it gets edited in all filename locations  
    - if you delete one, it still exists in other places

readlink:  prints the value of a [symbolic link](http://www.computerhope.com/jargon/s/symblink.htm) or absolute path of file name.

-f : is canonical file name i.e with absolute path.

ln -s file1.txt link1.txt (Creating soft link for the file file1.txt)

echo $(readlink -f link1.txt) # /home/akiran1234/file1.txt

echo $(readlink -f file1.txt) # /home/akiran1234/file1.txt

IP Address Commands

hostname: Display the hostname(server name)

hostname -i : Will display the i/p address.

hostname | nslookup: will display the i/p address of the host.

Ifconfig : will display the ip address of the machine.

Login Details

who: displays the list of users connected to the server.

who am i:(with space) will display only current user information.

whoami :(without space) will display the current child user name or switched username, If not switched will show the current user name.

date: Displays the current date and time.

cal: Displays current month.

cal 2011: Displays the whole year calendar.

File System commands

file filename # Displays what kind of file it is and record level delimiter.

Ex: ascii file or empty file and record level delimiter (CRLF, LF)

CRLF: Carriage Return Line Feed.

LF: Line Feed.

########## Cat is derived from concatenation ##################

cat>file\_name # Used to create a file.( contrl-d) will end writing into file.

cat filename: #Used to open the file that is existing. (vi filename: will open if already existing , else it will open by creating new one)

cat>>filename: Used to append the data to existing file.

cat>abc.txt #creates new file

cntr-d # will end writing into file

cat>>abc.txt #appends the data to existing file abc.txt

cntrl-d(will end writing into file)

cat a b c d # will open multiple files at a time on the screen on by one

cat –n filename # Display the file with line numbers on the screen.

cat>.kiran # prefix dot before file will create hidden file

more/less: "More is less, and less is more." -- UNIX hear say  
more and less is programs used to break up long text files, into screen size bits, for display. Especially useful when piped (\*4) from other commands, to break their output down.

view filename # open the file in vi mode read only.

pg filename # opens file in page wise.

fold filename # will displays the long lines by breaking into small on the same screen.

touch filename # Creates the file instantly with zero bytes. But can’t write data at the same time. But we can append the data once it’s created.

touch a b c : (Creates 3 files a,b,c at a stretch). No data in the files a,b,c.

cat>>a

Hi I am Kiran (writing text to file a)

cntr-d (Will end writing text to file)

Note: Cat is able to create only one file at a time, but touch creates multiple files at a time.

File Name with Space or Blank File: File name should be encoded with quotes while creating and listing the file.

touch ‘first file’: This command will create with file name (first file).

ls ‘first file’: While listing also it should be quoted.

ls (Listing of files)

ls –a : List all files including hidden.

ls –l: list files with long listing format.

ls –r : list files in reverse order (alphabetical)

ls –t: list files with the time of creation.

ls –R : list files in recursive order( Directory wise)

ls –i: list the inode value of the files/dir.

ls \* or rm \*-- display all or remove all

ls k\* : Display starting with “k”

ls \*k: Displays ending with “k”

ls kira?: Displays only with 5 chars.

ls ?a: Displays ending with a and with 2 chars only. 1 char can be anything.

ls ki[ranx]: displays total 3 chars and 3 char will be any char in the brackets.

ls abc[a-k]:Displays total 3 chars and 3rd char will be any char within the range a-k.

Removing Files (rm): (removes single file or multiple files at a time)

-R(Recursively option is must for dir’s, sub dir’s exists)

rm xyz: Removes file xyz

rm \* : removes all files in the directory.

rm kiran kumar achanta (removes multiple files separated by space)

rm –f : removes single or multi files forcibly

rm –i : will prompt before deleting the file.(Yes or NO)

rm –r: removes directories recursively(Current dir and sub dir’s).

cp existing file new\_file: copying a file.

cp kiran\* /xyz/abc/q123/ : copies all files with starting kiran to specified dir.

cp a b: a is existing b is existing. Now B will be over ridded with new data A.

-i: Prompt for override of existing file.

-R: copy directories recursively.

cp kriandir to /abc/xyz/: copying dir recursively.

mv (move command which is used to rename the files or move file)

mv abc xyz: abc file is renamed to xyz file. If xyz is not available it will create a new file with that name.

Hiding file or directory

mv xyz .xyz: hiding dir xyz.

mv abc .abc : Hiding file abc.

Check the above whether it works or not without –R

wc (word count with in a file): Display’s no of lines, words and characters.

wc abc:

20(L) 80(Words) 150(Total chars)

options: -l (display only lines)

-w(display only words)

-c (display only total chars in a file)

Comparison of files:

cmp file1 file2: if both are same it doesn’t display any msg on screen.

if different

file1 file2 differ: byte 1, Line 4 (Differ at line 4)

diff file1 file2: prints the different lines of two files on to the screen.

sdiff file1 file2:

cut : Used for flat files.

-d: To extract fields in a flat file

-c: To extract chars in all lines of a file.

cut –d “,” -f 1,5 filename: Displays the delimited file(,) from fields 1 to 5.

-d: Delimiter

-f: field

cut –d “|” –f 2 filename: displays 2 nd file only .

cut –c 5 filename: displays 5th char of each line of supplied file.

cut –c 5-10 filename: Displays 5 to 10 chars of each line of supplied file.

paste: To join 2 files horizontally with delimiter paste command used.

paste –d “|” file1 file2

sort (sort data as per ascii values)

sort filename: By default sorts in ascending.

-r : sort in descending

-u: sorts unique lines by eliminating duplicate lines.

Uniq (which displays only unique lines, prerequisite is sorted file should be provided) =sort –u filename.

uniq sortedfile: Displays only unique lines.

-d : displays only duplicate lines.

-c: counts how times lines have been repeated.

-u: displays only non-duplicate lines.

head – 10 filename: Displays first 10 rows of a file.

tail -20 filename: Displays last 20 records of a file.

piping(|) which is used to redirect the result to next command.

File Permissions:

chmod: Used to change the file permissions.

rwx rwx rwx

user group others

Total (rwx=7) ------ r=4, w=2, x=1.

Default permissions when a file or dir is created.

file: rw-rw-r (662)

dir: rwx-r\_x-r\_x (755)

chmod 7 4 2 filename: this file can be accessed .

chmod 000 dirname: Removing all the permissions (rwx) to directory.

Metacharacter: is a character that has a special meaning to a computer program -such as a [shell interpreter](https://en.wikipedia.org/wiki/Operating_system_shell) or a [regular expression](https://en.wikipedia.org/wiki/Regular_expression) engine.

 There are 14 meta characters that must be preceded by a backslash "\" in order to drop their special meaning and be treated literally inside an expression: the open/close square brackets, "[" and "]"; the backslash "\"; the caret "^"; the dollar sign "$"; the period or dot "."; the vertical bar or pipe symbol "|"; the question mark "?"; the asterisk "\*"; the plus-sign "+"; open/close curly braces, "{" and "}"; and open/close parenthesis, "(" and ")".

Meta characters= Wild card characters + Regular Expressions.

Note: Wild card characters have a different meaning and when used in conjunction with Regular expressions (i.e modifiers) the same characters will have different meaning.

Wildcard Characters: are **file-matching patterns** used to search for a files/directories. This wild cards has special meaning when used in regular expression programs like grep, sed and awk.

*Wild card characters (Used in search of files/Directories)*

(\*) ---Matches 0 or more chars

(?) – Matches only 1 char only.

[]—Matches any single char specified in the square brackets called SET.

[-] – Matches any char within the given range.

ls a\*; ls \* ;

ls ? # Search for a single character files.

ls ab?.txt; ls a??cd.txt;

ls abc[kgf]abc.txt; # Search for exactly one char with the set of char’s.

ls abc[0-9]abc.txt; # Search for range of numbers from 0-9.

Wild card Characters Vs Regular expressions

Wild card character deals with files and directory searches (**file-matching patterns**).

Regular expressions deals with matching a **pattern** in a specific series of characters in a single File/Multiple Files or Stream of input data.

Regular Expressions:

A regular expression is a concept of matching a **pattern** in a specific series of characters in a single File/Multiple Files or Stream of data.

Regular expressions search for patterns on a single line, and not for patterns that start on one line and end on another.

List of Languages/Programs that support R.E🡪 vi, grep, sed, awk, perl, python etc.

Now we will see usage of regular expression with grep program.

**Regular Expressions** are a feature of [UNIX](http://www.robelle.com/smugbook/unix.html). They describe a pattern to match, a sequence of characters, not words, within a line of text.

In regular expressions reading an expression is very important.

Basically the regular expressions are combination of the below 3 categories –

1. Anchor characters (Anchor a character sequence to beginning or end of **line** or beginning or end of **word**)
2. Character search Meta characters
3. Modifiers (Which will identify the no of preceding characters)

|  |  |  |
| --- | --- | --- |
| Line Anchor | ^ A(Caret) | Match expression at the start of a line, as in ^A. |
| Line Anchor | A$ (Question) | Match expression at the end of a line, as in A$. |
| Word Anchor | \< | Match for the beginning of the word. |
| Word Anchor | \> | Match for the ending of the word. |
| Character Search | . (Period) | Match a single character of any value, except end of line. |
| Character Search | [ ] (Set) | Match any one of the enclosed characters, as in [aeiou]. Use Hyphen "-" for a range, as in [0-9]. |
| Character Search | [^ ] | Match any one character except those enclosed in [ ], as in [^0-9]. |
| Modifier | \* (Asterisk) | Match zero or more of the preceding character or expression. |
| Modifier | \+ | Match one or more preceding character or expression. |
| Modifier | a\{x\} | Match exactly x occurrences of the preceding. |
| Modifier | a\{x,\} | Match x or more occurrences of the preceding. |
| Modifier | a\{x,y\} | Match x to y occurrences of the preceding. |
| Modifier | a\? | Matches zero or one occurrence at max of the previous character |
| Escape Operator | \(Back slash) | Turn off the special meaning of the next character |

grep (Globally search Regular Expression Print)

Options:

*-v: which doesn’t have supplied pattern*

*-n: print the displayed lines with numbers*

*-c: count no of lines Matching pattern and return digit.*

*-- color: prints with pattern highlighted.*

*-i: ignore case sensitive.*

grep + Regular expressions:

grep “hello” filename: It will check for character h followed by e followed by l followed by l and followed by o. Search will happen line by line.

grep “cat\*” filename: As per the **\*** definition it will search for zero or more occurrences of preceding char’s.

Here t will be zero or more occurrences and below is the all possible combination’s.

ca; cat; catt; cattttttttttt; # Repeated t’s will be more occurrences.

grep “kira\*” filename: Here a will be zero or more occurrence.

kir; kira; kiraa; kiraaaa; # Repeated a’s will be more occurrences.

grep “k\*” filename: This is zero or more char’s of k hence it will display lines without k and lines with single k and multiple k’s. # In essence it will display all lines.

grep “.” filename: (.) search for at least one single character in a line and this will ignore empty lines.

grep “.\*” filename: Search for zero or more occurrences of preceding char of \*. Here preceding char is (.), zero means - no char’s more occurrences means - any chars. Hence it will display all the lines.

grep ‘k.ran’ filename – (.) Check’s for any single character in between k & ran.

grep ‘kira..’ filename – (..) Checks for chars with kira with only two char’s total 5 chars.

grep ‘kira.\*’ filename: Check for Zero or more no of char’s after the expression kira

grep ‘kir[abc123]d’ filename- Checks for the 4th char to be either of the available chars with in the provided chars [abc123] and displays the results.

grep ‘kira[1-5]alpha’ filename – Checks for 5th char to be within the range of provided numbers 1-5.

grep ‘kira[0-9 a-z A-Z ]alpha’ filename- 5th char could with in the range of 0-9 or a-z or A-Z.

grep “[^abc]bus” filename: This will search for total 4 char’s and first char shouldn’t be neither a b c.

grep "a\{2\}" sample.txt: This will check for a followed by a (2 a’s) and display the lines where ever there is 2 a’s. It should have exact 2 a’s.

grep "cat\{2\}" sample.txt: This will check for cat followed by cat.

grep "a\{2,\}" sample.txt: This will display with minimum 2 a’s and can be more than 2 a’s.

grep "a\{2,4\}" sample.txt: This will check for a followed by a and so on. Minimum 2 a’s and maximum 4 a’s.

grep “ash\+” filename: As per the definition minimum one h and maximum of any occurrences. In grep “abc\*” this is zero or more occurrences of c.

ash; ashh; ashhhhhhh;

grep “ash\?” filename: As per the definition zero occurrence of h and maximum of one occurrence of h.

as; ash; # Only two possible combinations.

grep “^k” filename: Checks for the letter k in the beginning of the line (^).

grep “k$” filename: Checks for the letter k in the end of the line ($).

grep “^$” filename: This will return the empty lines.

grep “\<as” filename: This will read the beginning of word followed by char a followed by s

grep “and\>” filename: This will read the char a followed by n followed by d followed by word ending.

grep “\<and\>” filename: It will display the word and. First it will read beginning of the word followed by char a followed n followed by d followed by end of the word.

Sed Command (Stream editor): Works like search and replace.

* **Read**: SED reads a line from the input stream (file, pipe, or stdin) and stores it in its internal buffer called **pattern buffer**.
* **Execute**: All SED commands are applied sequentially on the pattern buffer. By default, SED commands are applied on all lines (globally) unless line addressing is specified.
* **Display**: Send the (modified) contents to the output stream. After sending the data, the pattern buffer will be empty.

**Syntax:** *sed options ‘s/regexp/replacement/flags’ filename*

s/- Stands for substitution

/ - delimiter

**Flags:**

/g- is a flag which represents globally. Below are the other available flags.

/n- replace nth instance of the REGEXP with REPLACEMENT.

/I- is flag used for case **I**nsensitive of regexp.

/p – write pattern space to STDOUT if a successful substitution takes place

/w filename – Write the pattern space to file if a successful substitution takes place.

sed ‘s/k/K/’ sample.txt # This will change the small k to capital K in the first occurrence of the line.

Ex: The king is great, king is very tall.

o/p: The King is great, king is very tall.

sed ‘s/k/K/g’ sample.txt # Putting flag g it has changed all the k to K.

o/p: The King is great, King is very tall.

sed ‘s/unix/linux/3’ sample.txt # This will replace third occurrence of unix with linux.

Note: This will note change the actual file unless the below option is specified.

sed -i ‘s/k/K/g’ sample.txt # This will modify the actual file, i=stands for inline.

sed 's/http:\/\//www/' file.txt # Replacing http:// with www used escape char /.

sed 's|http://|www|' file.txt # Alternate way to write the above stmt, many slashes might be confusing.

& Operator in sed:

sed ‘s/unix/&linux/g’ filename: This will search for word unix and concatenate with linux.

sed –n ‘3,-5p’ filename – Print lines from 3rd line to 5th line.

-n – don’t print lines default.

Sed ’10,5d’ filename – Deletes lines from 10-15.

Remote Server Commands:

Ssh

Scp

ftp

find rootdir -name filename\*: Look for filename starting with rootdir or current dir and sub sub dir’s used in Unix.

locate: locate is a file-location tool in Linux distributions. Find the file you're looking for on a vast system, in milliseconds, by searching a database

sudo yum install mlocate; sudo updated;

find: find is also a file-location tool for Unix OS. It does an actual search, much slower, but always updated.

which: locate a command, display its pathname or alias

**Networking Commands**

**Process Commands**

ps –ef: List all the process running in unix system.

kill-9 pid: Kill any process. Signal 9 can't be ignored.

**Admin Tasks**

To create new user:

adduser newusername: will create a new user.

passwd newusername: will ask to set the password for new user.

When we run ‘useradd‘ command in Linux terminal, it performs following major things:

1. It edits /etc/passwd, /etc/shadow, /etc/group and /etc/gshadow files for the newly created User account.
2. Creates and populate a home directory for the new user.
3. Sets permissions and ownerships to home directory.

Once a new user created, its entry automatically added to the ‘/etc/passwd‘ file. The file is used to store user’s information and the entry should be.

kiran:x:504:504::/home/kiran:/bin/bash

The above entry contains a set of seven colon-separated fields, each field has its own meaning. Let’s see what these fields are:

1. Username (kiran): User login name used to login into system. It should be between 1 to 32 characters long.
2. Password (x): User password (or x character) stored in /etc/shadow file in encrypted format.
3. User ID (504): Every user must have a User ID (UID) User Identification Number. By default UID 0 is reserved for root user and UID’s ranging from 1-99 are reserved for other predefined accounts. Further UID’s ranging from 100-999 are reserved for system accounts and groups.
4. Group ID (504): The primary Group ID (GID) Group Identification Number stored in /etc/group file.
5. User Info: This field is optional and allow you to define extra information about the user. For example, user full name. This field is filled by ‘finger’ command.
6. Home Directory (/home/kiran): The absolute location of user’s home directory.
7. Shell (/bin/bash): The absolute location of a user’s shell i.e. /bin/bash.

grep ‘kiran’ /etc/passwd : To verify user ‘kiran’ is created.

Note: By default a group (gid) is created with the same user id (uid) if not specified with some specific group. In the above screen shot both uid and gid is 504.



In the above example user name (user2) and group name (user2) are same.

Create Groups:

addgroup doctors: A new group doctors is created.

Verify by checking the file: /etc/group

usermod -a -G doctors kiran: Adding a user (kiran) to group (doctors).

chown: change owner and change group with root privileges

sudo chown owner filename: This will change the owner of the file.

Sudo chown new\_owner:new\_group file/directory: This will change both the owner and group of file/directory.