



Introduction to Linux

What is Linux?

- Linux(Kernel) + Utilities (GNU, others) = Linux Distribution
- Linux(Distribution) is, in simplest terms, an operating system. Just like any other operating system, it provides an interface between the user and the computer, enabling communication between them, and executing various tasks, but there are some features that make it more extensible and a lot more developer-friendly, some of these feature are –
 - Very powerful command line
 - Extensibility
 - Open source and great community support
 - Stability and robustness (reason most servers and high computing machines run on linux).



Agenda

- History
- Linux Introduction
- Basic features of Linux
- Linux Flavors and System Architecture
- Getting started :-
 - Connecting to the system
 - Basic commands
 - Linux help
 - Editors
- Understanding and managing processes
- Demo



HISTORY OF LINUX

- In 1984 by Richard Stallman aim at developing a complete Unix-like operating system which is free for copying and modification. Stallman built the first free GNU C Compiler in 1991. But still, an OS was yet to be developed
- A famous professor Andrew Tanenbaum developed Minix, a simplified version of UNIX that runs on PC. Minix is for class teaching only and no intention for commercial use.
- In Sept 1991, Linus Torvalds, a second year student of Computer Science at the University of Helsinki, developed the preliminary kernel of Linux, known as Linux version 0.0.1




Continued..


- **Shell** - Linux provides a special interpreter program which can be used to execute commands of the operating system. It also acts as an interface between the kernel and user.
- **Security** - Linux provides user security using authentication features like password protection/ controlled access to specific files/ encryption of data.
- **Stability** – a Linux distribution is used on a variety of devices, and because of open source community, bugs are found and fixed quickly, adding to the stability of linux.
- **Extensibility** – because it's open source, we can pretty do much anything we want, extend and mold the system just like the way we want it to be.



BEFORE LINUX

- In 80's, Microsoft's DOS were the dominant OS for PC and Apple MAC was better, but expensive
 - UNIX was much better, but much, much more expensive. Used in minicomputers for commercial applications
 - People was looking for a UNIX based system, which is cheaper and can run on PC
 - Both DOS, MAC and UNIX were proprietary, i.e., the source code of their kernel is protected
 - No modification is possible without paying high license fees
- 
- A decorative geometric pattern at the bottom of the slide, consisting of various colored triangles (orange, yellow, pink, purple, blue) arranged in a jagged, mountain-like shape.

Basic features of Linux

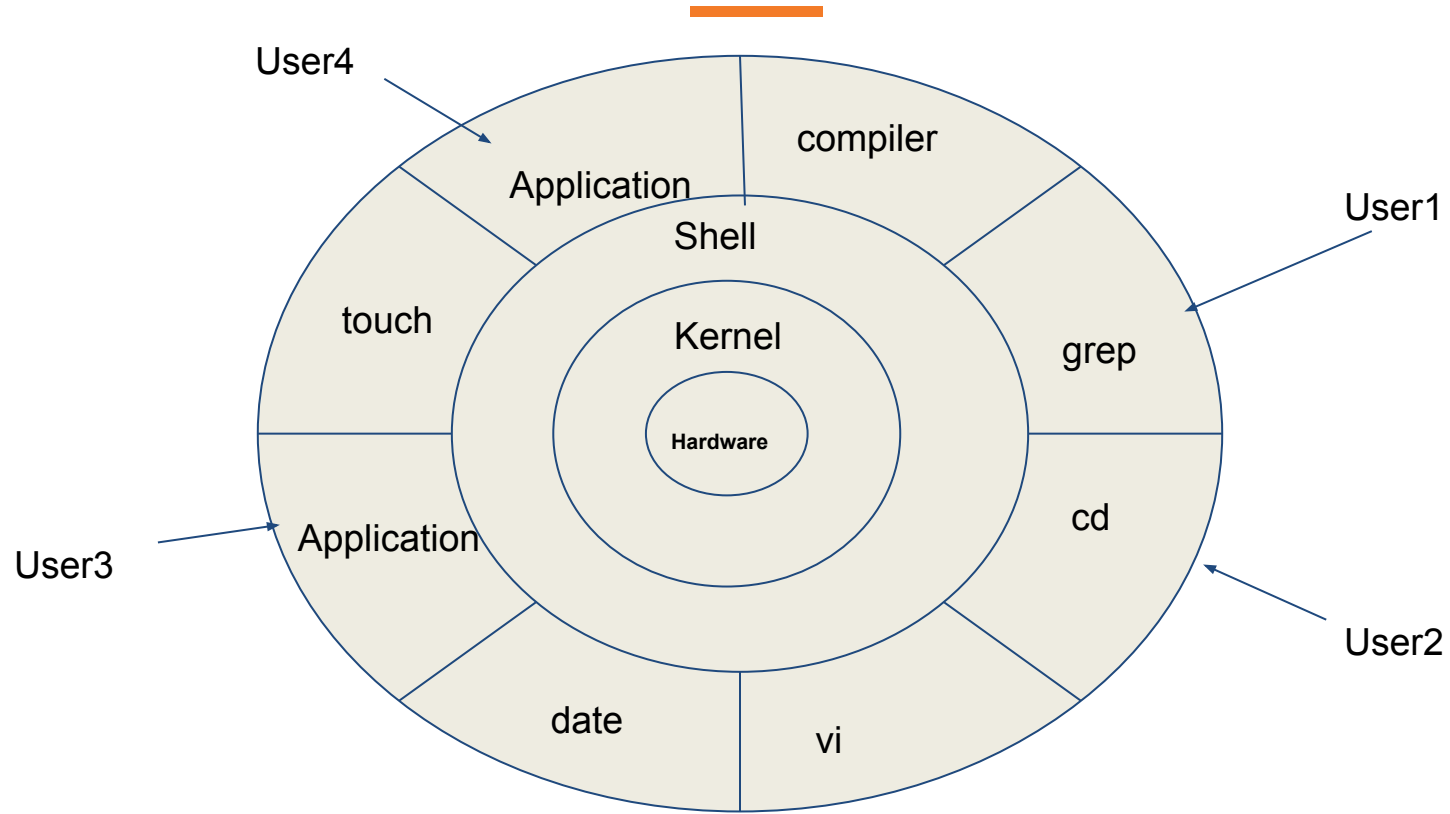
- **Portable** - Portability means softwares can work on different types of hardware in the same way. Linux kernel and application programs support their installation on any kind of hardware platform.
 - **Open Source** - Linux source code is freely available and it is a community based development project. Multiple teams work in collaboration to enhance the capability of Linux operating system and it is continuously evolving.
 - **Multi-User** - Linux is a multiuser system means multiple users can access system resources like memory/ ram/ application programs at the same time.
 - **Multiprogramming** - Linux is a multiprogramming system means multiple applications can run at the same time.
- 
- A decorative geometric pattern at the bottom of the slide, consisting of various colored triangles (orange, yellow, pink, purple, blue) arranged in a jagged, mountain-like shape.

Linux Flavour

- **RED HAT LINUX** • One of the original Linux distribution. • The commercial, nonfree version is Red Hat Enterprise Linux, which is aimed at big companies using Linux servers and desktops in a big way. • Free version: Fedora Project.
- **DEBIAN GNU/LINUX** • A free software distribution. Popular for use on servers. However, Debian is not what many would consider a distribution for beginners, as it's not designed with ease of use in mind.
- **SUSE LINUX** • SuSE was recently purchased by Novell. This distribution is primarily available for pay because it contains many commercial programs, although there's a stripped-down free version that you can download.
- **MANDRAKE LINUX** • Mandrake is perhaps strongest on the desktop. Originally based off of Red Hat Linux.
- **GENTOO LINUX** • Gentoo is a specialty distribution meant for programmers.



Linux Architecture



Continued..

Linux System Architecture consists of following layers :

- **Hardware layer** - Hardware consists of all peripheral devices (RAM/ HDD/ CPU etc).
- **Kernel** - Core component of Operating System, interacts directly with hardware, provides low level services to upper layer components.
- **Shell** - An interface to kernel, hiding complexity of kernel's functions from users. Takes commands from user and executes kernel's functions.
- **Utilities** - Utility programs giving user most of the functionalities of an operating systems.



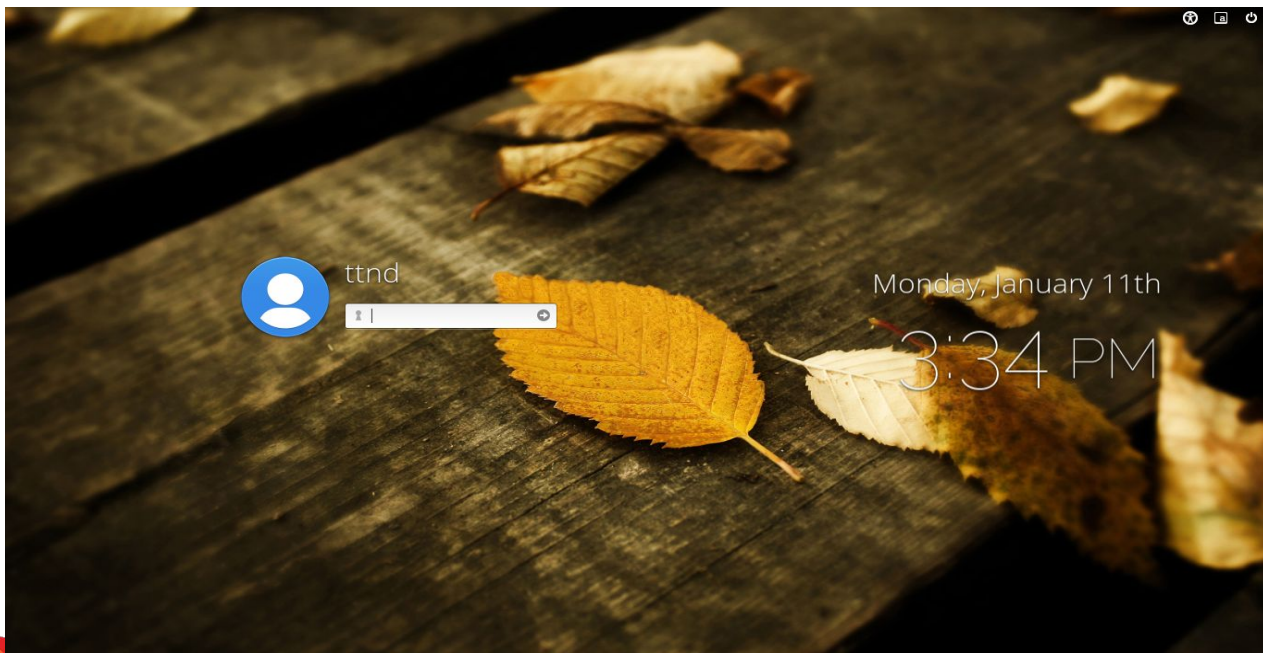
Linux File Structure



Getting Started With Linux

Ubuntu developers made a conscientious decision to disable the administrative root account by default in all Ubuntu installations. This does not mean that the root account has been deleted or that it may not be accessed. It merely has been given a password which matches no possible encrypted value, therefore may not log in directly by itself.

Instead, users are encouraged to make use of a tool by the name of *sudo* to carry out system administrative duties. The default login user name of ubuntu machine is “**ubuntu**”.



Basic Commands

- **ls**: The command "ls" stands for (List Directory Contents), List the contents of the folder
- **ls -l**: The command "ls -l" list the content of folder, in long listing fashion
- **ls -a**: Command "ls -a", list the content of folder, including hidden files starting with '.'

```
root@tecmint:~# ls -l

total 40588
drwxrwxr-x 2 ravisaive ravisaive 4096 May  8 01:06 Android Games
drwxr-xr-x 2 ravisaive ravisaive 4096 May 15 10:50 Desktop
drwxr-xr-x 2 ravisaive ravisaive 4096 May 16 16:45 Documents
drwxr-xr-x 6 ravisaive ravisaive 4096 May 16 14:34 Downloads
drwxr-xr-x 2 ravisaive ravisaive 4096 Apr 30 20:50 Music
drwxr-xr-x 2 ravisaive ravisaive 4096 May  9 17:54 Pictures
drwxrwxr-x 5 ravisaive ravisaive 4096 May  3 18:44 Tecmint.com
drwxr-xr-x 2 ravisaive ravisaive 4096 Apr 30 20:50 Templates
```

- d (stands for directory).
- rwxr-xr-x is the file permission of the file/folder for owner, group and world.
- The 1st ravisaive in the above example means that file is owned by user ravisaive.
- The 2nd ravisaive in the above example means file belongs to user group ravisaive.
- 4096 means file size is 4096 Bytes.
- May 8 01:06 is the date and time of last modification.
- And at the end is the name of the File/Folder.

Continued...

- **pwd** : The command “pwd” (print working directory), prints the current working directory with full path name from terminal.
- **cd** : the frequently used “cd” command stands for (change directory), it change the current working directory.
- **cd ~** : “cd ~” will change the working directory to user’s home directory, and is very useful if a user finds himself lost in terminal.
-
- **cd ..** : “cd ..” will change the working directory to parent directory (of current working directory).


```
root@tecmint:~# pwd
```

```
/home/user/Desktop
```

Continued...

- **which:** “which” search for executables in the directories specified by the environment variable PATH. And if found out, the full pathname of this executable will be printed.
`$ which ls`
- **whereis:** “*whereis*” search for executables, source files, and manual pages using a database built by system automatically.
`$ whereis ls`
- **find:** finds files by filename specified in a location.
`$find /home/username -name ls`
- **locate:** also finds files by filename but does not search the directory structure itself but only a database prepared by updatedb command. The *locate* is faster than *find* but less accurate.
`$locate ls`

Which command should I use?

- It really depends on what you want to do:
 - If you want to **find a linux program** (or its source or documentation) use **whereis**.
 - if you want to **find executable/binary** files use **which**.
 - If you want to **find an often-used file** or want to perform a quick but not too accurate first search for a file use **locate**.
 - If you want to find one of your **personally created** files use **find**.
- 

Continued..

- **touch:** The “touch” command creates the file, only if it doesn’t exist. If the file already exists it will update the timestamp and not the contents of the file.

`$ touch abc.txt`

- **cat:** The “cat” stands for (Concatenation). Concatenate (join) two or more plain file and/or print contents of a file on standard output.

`$ cat abc.txt`

- **cp:** The “copy” stands for (Copy), it copies a file from one location to another location.

`$ cp /home/user/Downloads abc.tar.gz /home/user/Desktop`

- **mv:** The “mv” command move/rename a file from one location to another location.

`$ mv /home/user/Downloads abc.tar.gz /home/user/Desktop`

- **mkdir:** The “mkdir” (Make directory) command create a new directory with name path. However is the directory already exists, it will return an error message “cannot create folder, folder already exists”.

`$ mkdir -p /a/b/c`

- **rm:** The “rm” command removes each specified FILE. By default, it does not remove directories.

`$ rm -i filename` : Prompt before every removal.

`$ rm -f`: forcefully remove files and never ask

`$ rm -r`: Remove directories and their contents recursively.



-
- **tar:** The “tar” command is a Tape Archive is useful in creation of archive, in a number of file format and their extraction.

```
$ sudo apt -get install zip
```

(Remember 'z' for .tar.gz)

```
$ tar -jxvf abc.tar.bz2 (Remember 'j' for .tar.bz2)
```

```
$ tar -cvf archive.tar.gz(.bz2) /path/to/folder/abc
```

- **Note:** A 'tar.gz' means gzipped. 'tar.bz2' is compressed with bzip which uses a better but slower compression method.

other commands similar to tar perform same task

```
$ gzip -9 lab1.tar
```

```
$ untar lab1.tar
```

```
$ ungzip lar1.zip
```



Continued..

- **chown:** The Linux “chown” command stands for (change file owner and group). Every file belongs to a group of user and a owner.

This “chown” command is used to change the file ownership and thus is useful in managing and providing file to authorised user and usergroup only.

lets Do ‘ls -l’ into your directory and you will see something like this.

```
root@tecmint:~# ls -l

drwxr-xr-x 3 server root 4096 May 10 11:14 Binary
drwxr-xr-x 2 server server 4096 May 13 09:42 Desktop
```

```
root@tecmint:~# chown server:server Binary

drwxr-xr-x 3 server server 4096 May 10 11:14 Binary
drwxr-xr-x 2 server server 4096 May 13 09:42 Desktop
```

Continued..

- `chmod`: The Linux “`chmod`” command stands for (change file mode bits). `chmod` changes the file mode (permission) of each given file, folder, script, etc.. according to mode asked for.
- There exist 3 types of permission on a file (folder or anything but to keep things simple we will be using file).

```
Read (r)=4  
Write(w)=2  
Execute(x)=1
```

lets try some combinations

```
$ chmod 666 abc.txt
```

```
-rw-rw-rw-+ 15 staff staff 510 Jan 11 22:08
```

```
$ chmod 777 abc.txt
```

```
-rwxrwxrwx+ 15 staff staff 510 Jan 11 22:08
```

```
$ chmod 555 abc.txt
```

```
-r-xr-xr-x+ 15 staff staff 510 Jan 11 22:08
```

```
$ chmod 333 abc.txt
```

```
-wx-wx-wx-+ 15 staff staff 510 Jan 11 22:08
```

Continued..

- **cal**: The “cal” (Calendar), it is used to displays calendar of the present month or any other month of any year that is advancing or passed.
- **echo**: The echo command is used to print content on the screen
\$echo “hello world”
- **date**: The “date” (Date) command print the current date and time on the standard output, and can further be set.

```
root@tecmin:~# date --set='14 may 2013 13:57'
```

```
Mon May 13 13:57:00 IST 2013
```

Creating User/group

- **Create user** : [useradd](#) -g <group> -d <home directory> -c <comment> -s <shell> username
- **Set password** : passwd username
- **Lock user account** : passwd -l <username>
- **Delete user account**: userdel -r <username>

Groups are great way to share your work

- **Add group**: groupadd <groupname>
- **Group Types** : Primary(user private group) and secondary groups.
- **Modify user account** : [usermod](#) -G <secondaryGroup> -a <username>



Standard Redirections

- By default, standard output directs its contents to the display. To redirect standard output to a file, the ">" character is used like this:
`$ ls > abc.txt`
- Each time the command above is repeated, file_list.txt is overwritten (from the beginning) with the output of the command **ls**. If you want the new results to be *appended* to the file instead, use ">>" like this:
`$ ls >> abc.txt`
- Many commands can accept input from a facility called *standard input*. By default, standard input gets its contents from the keyboard, but like standard output, it can be redirected. To redirect standard input from a file instead of the keyboard, the "<" character is used like this:
`$ sort < abc.txt`
`$ sort < abc.txt > new.txt`
- The most useful and powerful thing you can do with I/O redirection is to connect multiple commands together with what are called *pipes*. With pipes, the standard output of one command is fed into the standard input of another.
`$ ls -l | less`

Basic Commands

- **head** - The *head* command reads the first 10 lines of any file given to it. If it is desired to obtain some number of lines other than the default ten, the *-n* option can be used followed by an integer indicating the number of lines desired

```
$ head -n 15 abc.txt
```

- **tail** - The “*tail*” command reads the last 10 lines of any file given to it. If it is desired to obtain some number of lines other than the default ten, the *-n* option can be used followed by an integer indicating the number of lines desired

```
$ tail -n 15 abc.txt
```

- **Pagers** – As the name implies, roughly a pager is a piece of software that helps the user get the output one page at a time

```
$ less
```

```
$ more
```



Continued..

- **diff**:- The “diff” analyzes two files and prints the lines that are different. Essentially, it outputs a set of instructions for how to change one file in order to make it identical to the second file.
`$ diff abc.txt xyz.txt`
- **wc**: The `wc` (word count) command in Unix/Linux operating systems is used to find out number of newline count, word count, byte and characters count in a files specified by the file arguments.

`$ wc -l` : Prints the number of lines in a file.

`$ wc -w` : prints the number of words in a file.

`$ wc -c` : Displays the count of bytes in a file.

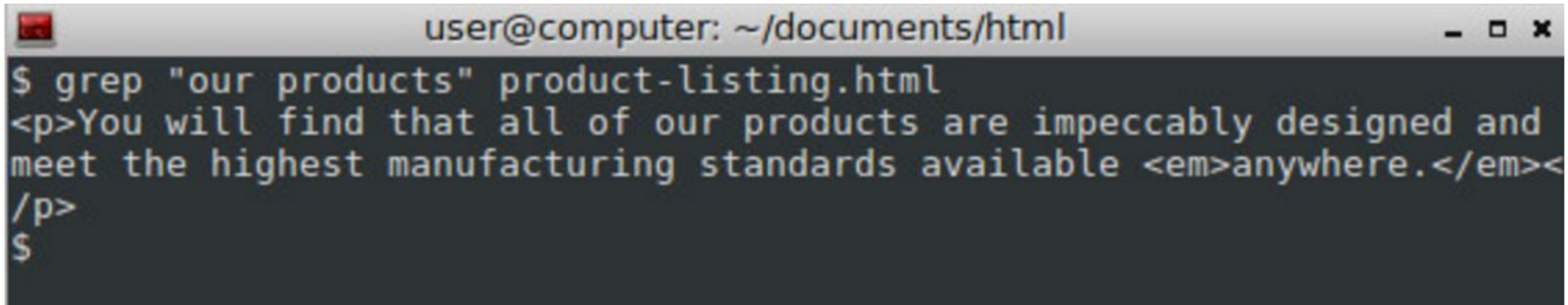
`$ wc -m` : prints the count of characters from a file.

`$ wc -L` : prints only the length of the longest line in a file.



Continued..

grep: It stands for "global regular expression print," processes text line by line and prints any lines which match a specified pattern.

A terminal window with a title bar that reads "user@computer: ~/documents/html". The terminal has a dark background with light-colored text. The command "\$ grep 'our products' product-listing.html" has been executed. The output is displayed in two lines: "<p>You will find that all of our products are impeccably designed and meet the highest manufacturing standards available anywhere.</p>". The prompt "\$" is visible at the bottom left of the terminal area.

```
user@computer: ~/documents/html
$ grep "our products" product-listing.html
<p>You will find that all of our products are impeccably designed and
meet the highest manufacturing standards available <em>anywhere.</em><
/p>
$
```

\$ grep -i "boo" /etc/passwd #You can force grep to **ignore** word case

\$ grep -r "192.168.1.5" /etc/ #You can search **recursively**

\$ grep -v bar /path/to/file #You can use -v option to print **inverts** the match

\$ grep --color vivek /etc/passwd #You can force grep to display output in **colors**

Continued..

Symbolic links - They are kind of shortcuts in the Linux/Unix world.

- **Soft Link** :- Soft links are very similar to what we say "Shortcut" in windows, is a way to link to a file or directory. It creates new file pointing to original file but with different inode.

```
$ ln -s /home/abc/abc.txt /home/user/Desktop/abc.txt
```

- **Hard Link** : Hard link is a bit different from soft link. It create new file with the same inode reference.

```
$ln /home/abc/abc.txt /home/user/Desktop/abc.txt
```

Which one should I use?

1. Link across filesystems: If you want to link files across the filesystems, you can only use symlinks/soft links.
2. Links to directory: If you want to link directories, then you must be using Soft links, as you can't create a hard link to a directory.
3. Storage Space: Hard links takes very negligible amount of space, as there are no new inodes created while creating hard links. In soft links we create a file which consumes space (usually 4KB, depending upon the filesystem)
4. Moving file location: If you move the source file to some other location on the same filesystem, the hard link will still work, but soft link will fail.
5. Redundancy: If you want to make sure safety of your data, you should be using hard link, as in hard link, the data is safe, until all the links to the files are deleted, instead of that in soft link, you will lose the data if the master instance of the file is deleted.

Environment Variables

`env` is a shell command for Linux, Unix, and Unix-like operating systems. It can be used to print a list of the current environment variables, or to run another program in a custom environment

If `env` is run without any options, it prints the variables of the current environment. Otherwise, `env` sets each `NAME` to `VALUE` and executes `COMMAND`.

Below is brief description of some commonly-used environment variables:

\$ **echo \$HOME** : The current user's home directory.

\$ **echo \$LOGNAME** : The name of the current user.

\$ **echo \$PATH** : Pathnames to be searched when executing commands.

\$ **env TZ=MST7MDT date** : To set env variable

\$ **env --unset=NAME** : remove variable from the environment



VI

2 modes:

- Input mode
- Command mode

Cursor movement:

- h (left), j (down), k (up), l (right)
- ^f (page down)
- ^b (page up)
- ^ (first char.)
- \$ (last char.)
- G (bottom page)
- :1 (goto first line)

Switch to input mode:

- a (append)
- i (insert)
- o (insert line after)
- O (insert line before)

Delete:

- dd (delete a line)
- d10d (delete 10 lines)
- d\$ (delete till end of line)
- dG (delete till end of file)
- x (current char.)

Paste:

- p (paste after)
- P (paste before)

Undo:

- u

Search:

- /

Save/Quit:

- :w (write)
- :q (quit)
- :wq (write and quit)
- :q! (give up changes)

Installing Software via command line

Dpkg is filetype and i
for install

- Installed Software - `dpkg -i openssh-server`
- `sudo apt-get`
- `sudo apt-cache search "openssh"`
- `sudo apt-get install <packagename>`
- Info about any installed software/command - which



process management

- **top:** The “top” command is one of the most frequently used commands in our daily system administrative jobs. top command displays processor activity of your Linux box and also displays tasks managed by kernel in real-time. It'll show processor and memory are being used and other information like running processes. This may help you to take correct action
- **ps:** The “ps” command on linux is one of the most basic commands for viewing the processes running on the system. It provides a snapshot of the current processes along with detailed information like user id, cpu usage, memory usage, command name etc.



More Commands

- Display Username: `whoami`
- SuperUser: `sudo`
- Switching users: `su`
- `.bashrc`
- Display help: `cmd --help`, `cmd -h`
- Display Manual: `man cmd`
- Alias: `alias lsit="ls -l"`

`$ list`

`drwx-----+ 20 admin staff 680 Jan 18 12:46 Desktop`

`drwx-----+ 15 admin staff 510 Jan 11 22:08 Documents`



Connect to remote Machine

- **ssh** :ssh (SSH client) is a program for logging into a remote machine and for executing commands on a remote machine.

```
localhost:[~]> ssh -l username remote-server
username@remote-server password:
remote-server:[~]>
```

```
localhost:[~]> ssh username@remote-server
username@remote-server password:
remote-server:[~]>
```

```
localhost:[~]> ssh user@remote-host "ls test"
online-backup.dat
oracle-storage.bat
unix-dedicated-server.txt
```


Remotely Copy Content

scp: It allows files to be copied to, from, or between different hosts. It uses ssh for data transfer and provides the same authentication and same level of security as ssh.

- Copy the file "foobar.txt" from a remote host to the local host

```
$ scp your_username@remotehost.edu:foobar.txt /some/local/directory
```

- Copy the file "foobar.txt" from the local host to a remote host

```
$ scp foobar.txt your_username@remotehost.edu:/some/remote/directory
```

- Copy the file "foobar.txt" from remote host "rh1.edu" to remote host "rh2.edu"

```
$ scp your_username@rh1.edu:/some/remote/directory/foobar.txt your_username@rh2.edu:/some/remote/directory/
```



Some useful commands

- What did I do: `history`
- How to download : `wget <link>`
- `ping`: check network connectivity with host machine
- `export`: mark a variable to be exported to child-processes
- Setting your hostname: `hostname <name>`





Questions....?



1. Create a directory "exercise" inside your home directory and create nested (dir1/dir2/dir3) directory structure inside "exercise" with single command.
2. Create two empty files inside dir2 directory: emptyFile1, emptyFile2 in single command and Remove the dir2 directory
3. Create one file file.txt containing text "hello world" and save it.
4. Find a "passwd" file using find command inside /etc. copy this file as passwd_copy and then rename this file as passwd_backup.
5. Try reading passwd_backup file in multiple tools: less, more, cat, strings etc and find the difference in their usage.
6. Find out the number of line in password_backup containing "/bin/false".
7. Redirect the output of the above commands into file "output".
8. Create a "test" user, create its password and find out its uid and gid.
9. Login as test user and edit the "output" file created above. Since the permission won't allow you to save the changes.
10. Configure such that test user can edit it.
 - a. Add group owner of the "output" file as the secondary group of testuser and check/change the "output" file permission if it is editable by group. Once done revert the changes.
 - b. Make the file editable to the world so that test user can access it. Revert the changes after verification
 - c. Change the ownership to edit the file.
- Try all the options and recommend best of them.
11. Create alias with your name so that it creates a file as "/tmp/alias_testing".
12. Edit ~/.bashrc file such that when you change to "test" user it should clear the screen and print "Welcome".
13. Install zip package.
14. Compress "output" and "password_backup" files into a tar ball. List the files present inside the tar created.
15. scp this file to test user
16. Unzip this tar ball by logging into the remote server.
17. Download any image from web and move to desktop