**File Commands**

**– ls**

directory listing

list directory content of files and directories.

-t sorts file by modification time (last modified is first)

-1 one file per line

-a display all files including hidden files

can merge/call more than one command in one

i.e. ls -t -1

**– ls -al**

display all files including hidden file + details

**– cd dir**

Change directory to dir

cd .. one level up dir

**– cd**

The cd command will allow you to change directories. When you

open a terminal you will be in your home directory. To move around

the file system you will use cd.

**– pwd**

stads for Print Working Directory

shows current working directory

-L (Logical) use PWD from environment, even if it contains symbolic

links

-P (Physical) Avoid all symbolic links

If both ‘-L‘ and ‘-P‘ options are used, option ‘L‘ is taken into priority.

If no option is specified at the prompt, pwd will avoid all symlinks,

i.e., take option ‘-P‘ into account.

Exit status 0 for sucess and non zero for failure

**– mkdir dir**

create a dirictory dir

stands for make directory

mkdir {dir1,dir2,dir3} - creates multiple directories in the current

location. Do not use space inside {}.

mkdir -p directory/path/newdir - creates a directory structure with

the missing parent directory (if any)

mkdir -m777 dir - creates a directory and sets full read, write, and

execute permission for all users

**– rm file**

Delete file

**– rm -r dir**

Delete directory dir

**– rm -f file**

Force remove file

**– rm -rf dir**

Force remove directory dir

**– cp file1 file2**

Copy file1 to file2

Overwrite the existing content of the file2 by file 1

**– cp -r dir1 dir2**

Copy dir1 to dir2, if not exist creates one

**– mv file1 file2**

Move file1 to file2

Overwrite the existing content of file2

Deletes the fiel1

**– ln -s file link**

Creates symbolic link to file in same directory and the link in the file

**– touch file**

Creates or updates the file

**– cat > file**

Print the content the file in the terminal

**– more file**

Gives detail info of the file

**– head file**

Print first 10 lines of the file

**– tail file**

Print last 10 line of the file

**– tail -f file**

Print last 10 line of file and open the cursor to write but do not modify the file

**• Process Management**

**– ps**

Currently active processes

**– top**

Display all running processes

Pid user cpu usage(%) memory usage(%) time command

Keep updating all the time

**– kill pid**

Kill process with specified process id

**– killall proc**

Kill process with name proc

**– bg**

Place foreground job in background

**– fg**

Brings the most recent job to foreground

**– fg n**

Brings job n to foreground

**• System Information**

**– date**

Prints date in weakDay date month(String) Year hh:mm:ss AM/PM IST

**– cal**

Prints the calendar month

Month(string) Year

Sunday to Saturdat

dates

**– uptime**

It is used to find out how long the system is active (running).

**-whoami**

Returns the username

**-finger user**

Display user name

User directory

Since when it is login

For how long it is logged in

Is there any mail

Or some task/plan in shor todo list kind of

**-uname -a**

Show kernel information

Username

Kernel version

System architecture information(32bit 64bit etc.)

**-cat /proc/cpuinfo**

Cpu information

Lots of information

**-cat /proc/maminfo**

Memory information

Same as above lots of information

**-man command**

Shows the user manual of the command specified

**-df**

Stands for disk free

Show disk usage

Partitioned wise- how much is used how much available

**-du**

Shows directory space usage

-a list all directory and sizes

-h display in human readable format

-c display grand total in output

-s display only total

**-whereis app**

If we write app it just prints appName that is written as it is

But if we write the command then it returns the location of the binary file of the command where it is located

**-which app**

Locates the executable file of the specified command(not app)

* **NETWORK Commands**
  + **-ping host**
    - Generally to check the internet connection
    - If no packet loss then internet working is fine
    - To check the internet stack within the machine ping 127.0.0.1 is use
  + **-whois domain**
    - Eg whois 216.58.206.46

Gives detailed information about the host who has ip as specified

In this case it is ip registered with google

* + **-dig domain**
  + **-dig -x host**
  + **-wget file**

To download the file generally the link to be given

Generates a log file also which has information about the download link and time etc

* + **-wget -c file**
* **Installation**
  + **-./configure**
  + **-make**
  + **-make install**

**Configure** the softwareThe configure script is responsible for getting ready to build the software on your specific system. It makes sure all of the dependencies for the rest of the build and install process are available, and finds out whatever it needs to know to use those dependencies.  
Unix programs are often written in C, so we’ll usually need a C compiler to build them. In these cases the configure script will establish that your system does indeed have a C compiler, and find out what it’s called and where to find it.

**Build** the software **(make)**  
Once configure has done its job, we can invoke make to build the software. This runs a series of tasks defined in a Makefile to build the finished program from its source code.  
The tarball you download usually doesn’t include a finished Makefile. Instead it comes with a template called Makefile.in and the configure script produces a customised Makefile specific to your system.

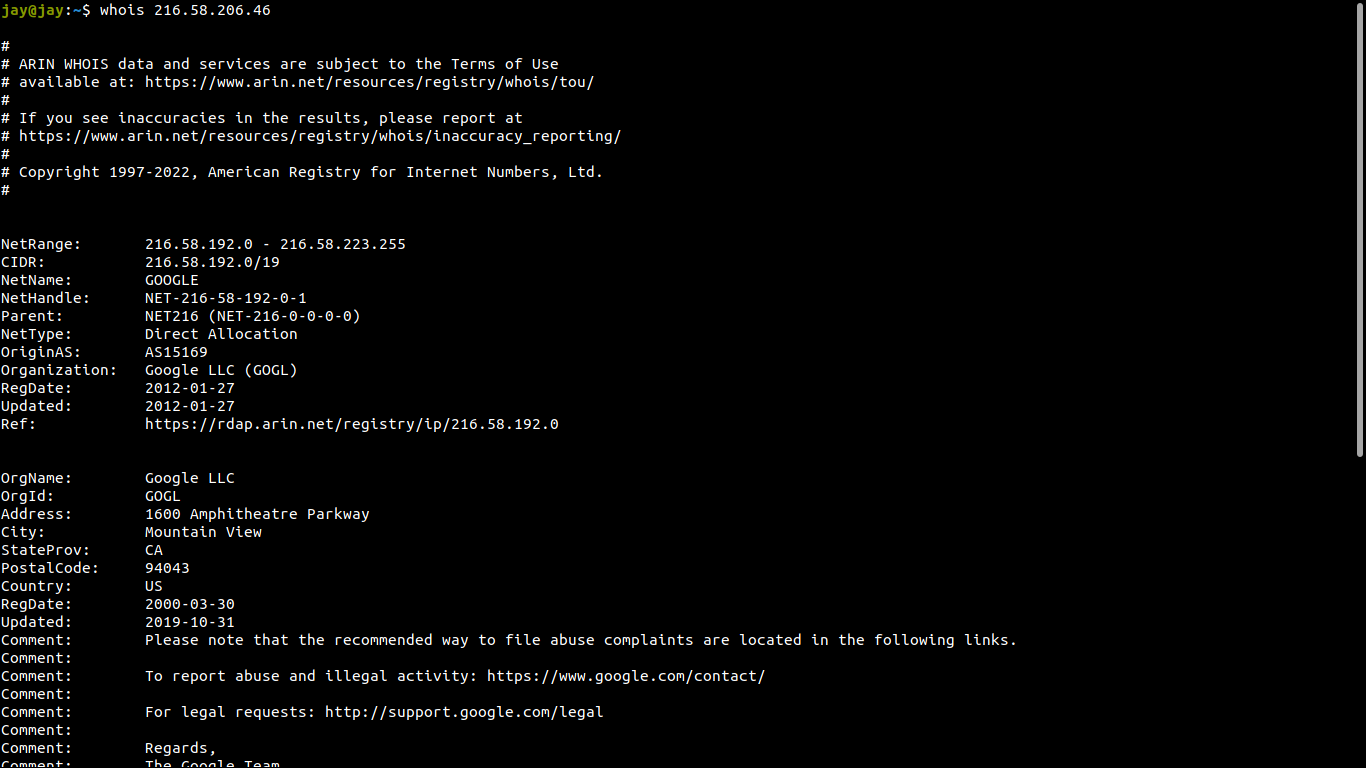
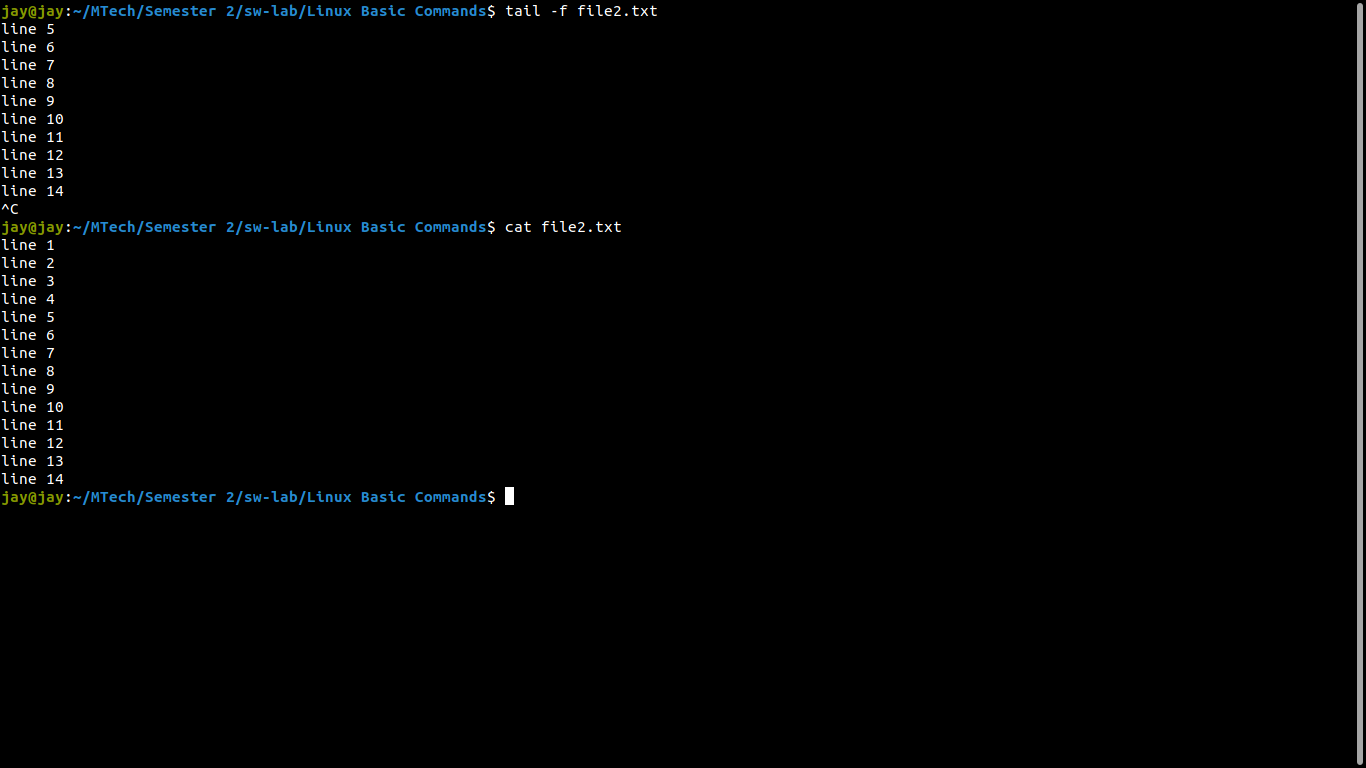
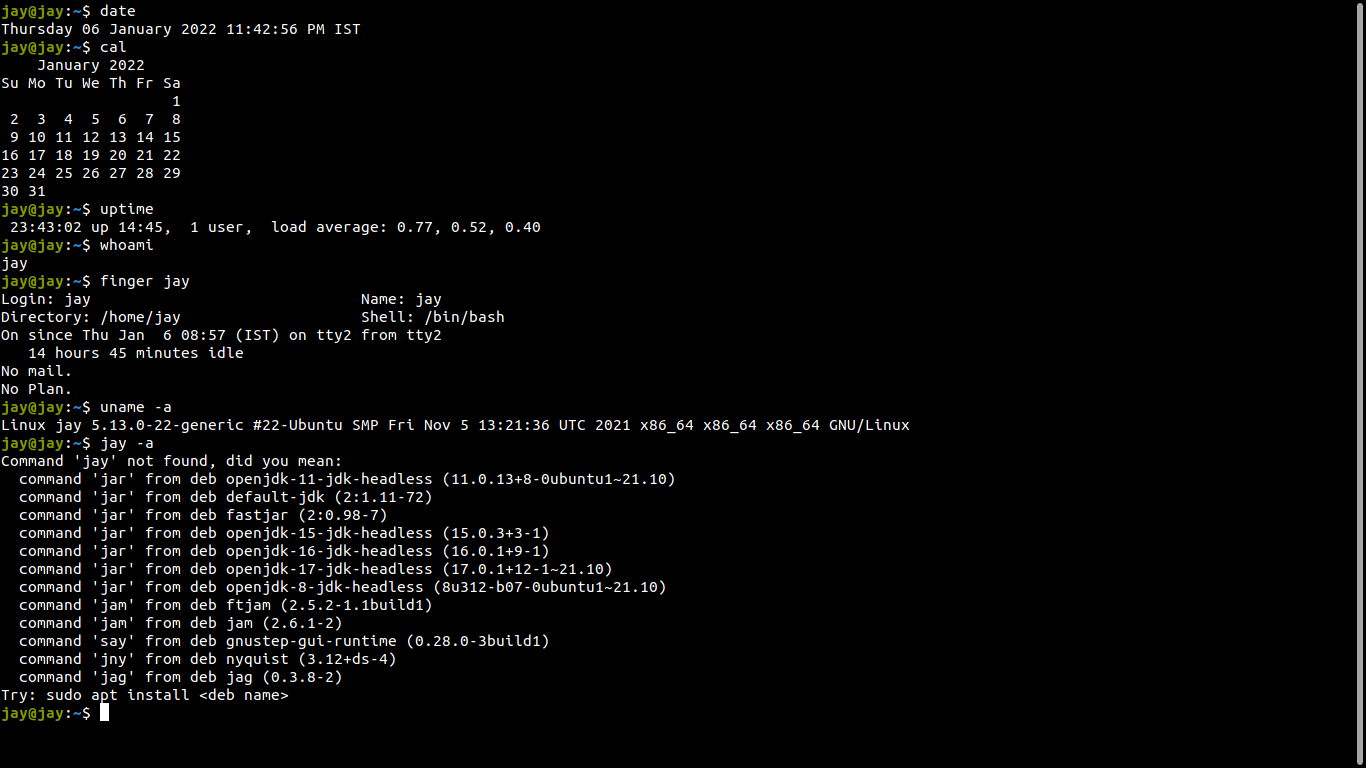
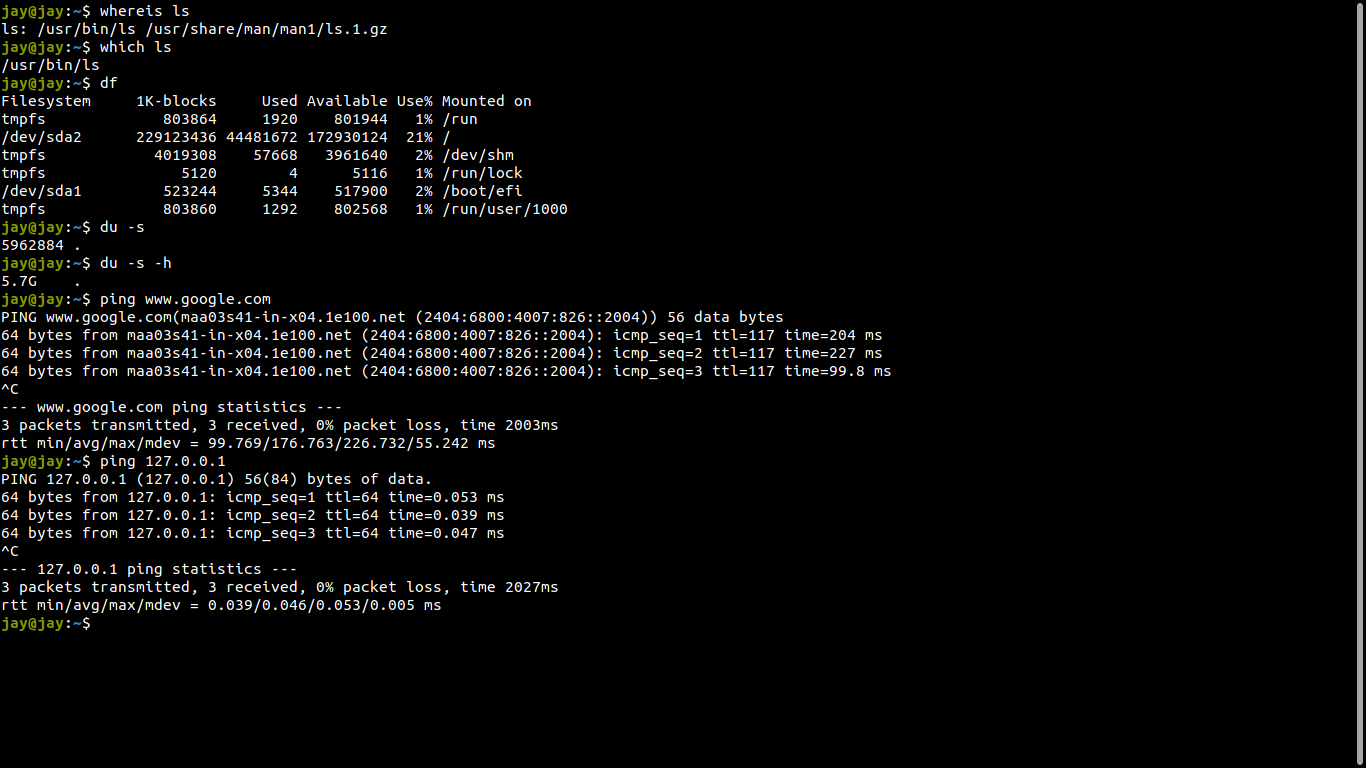
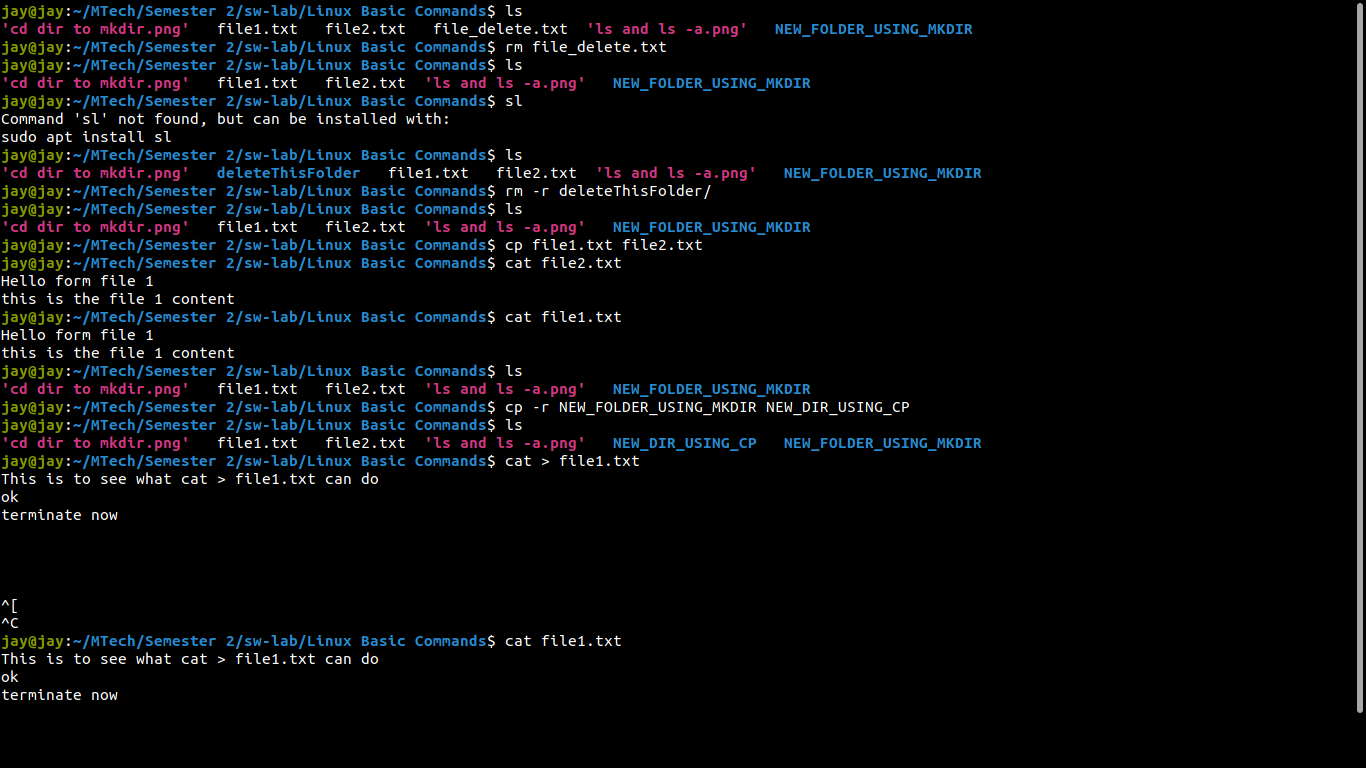
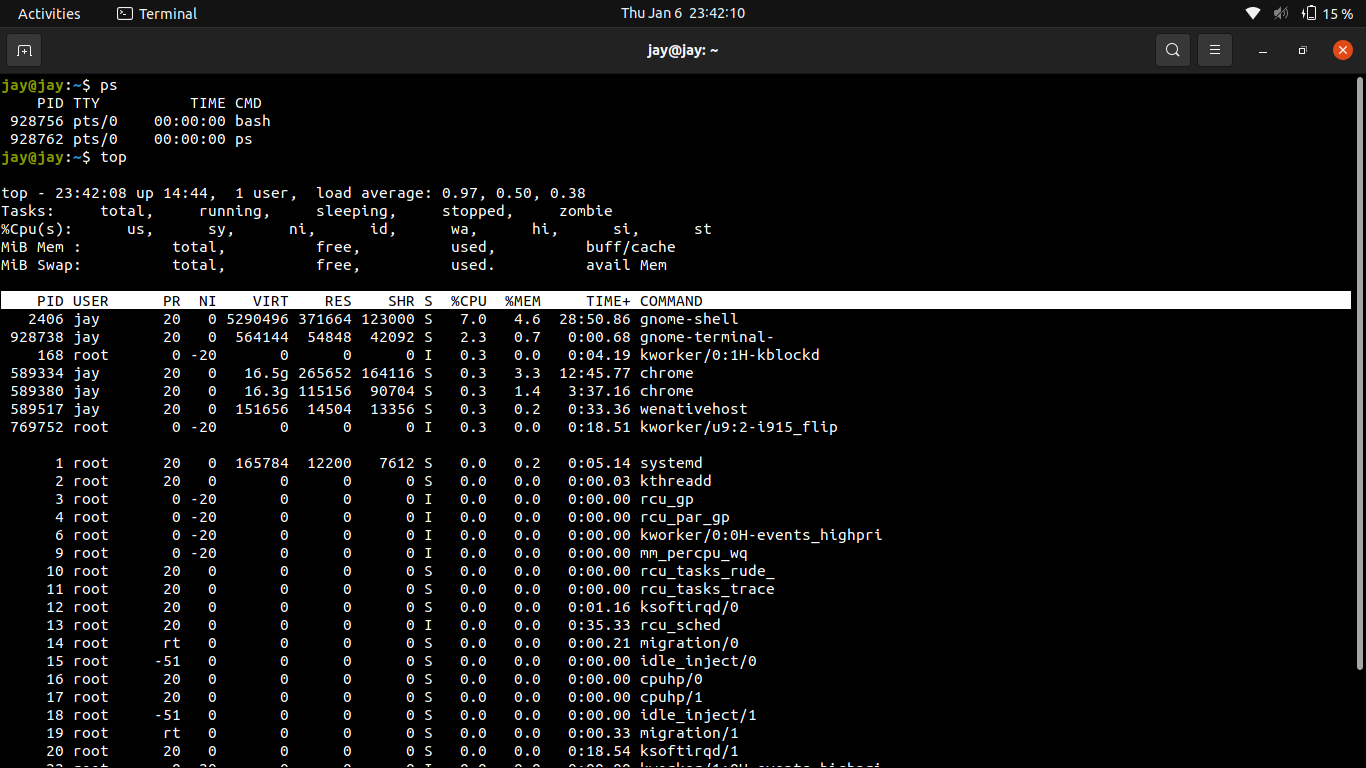
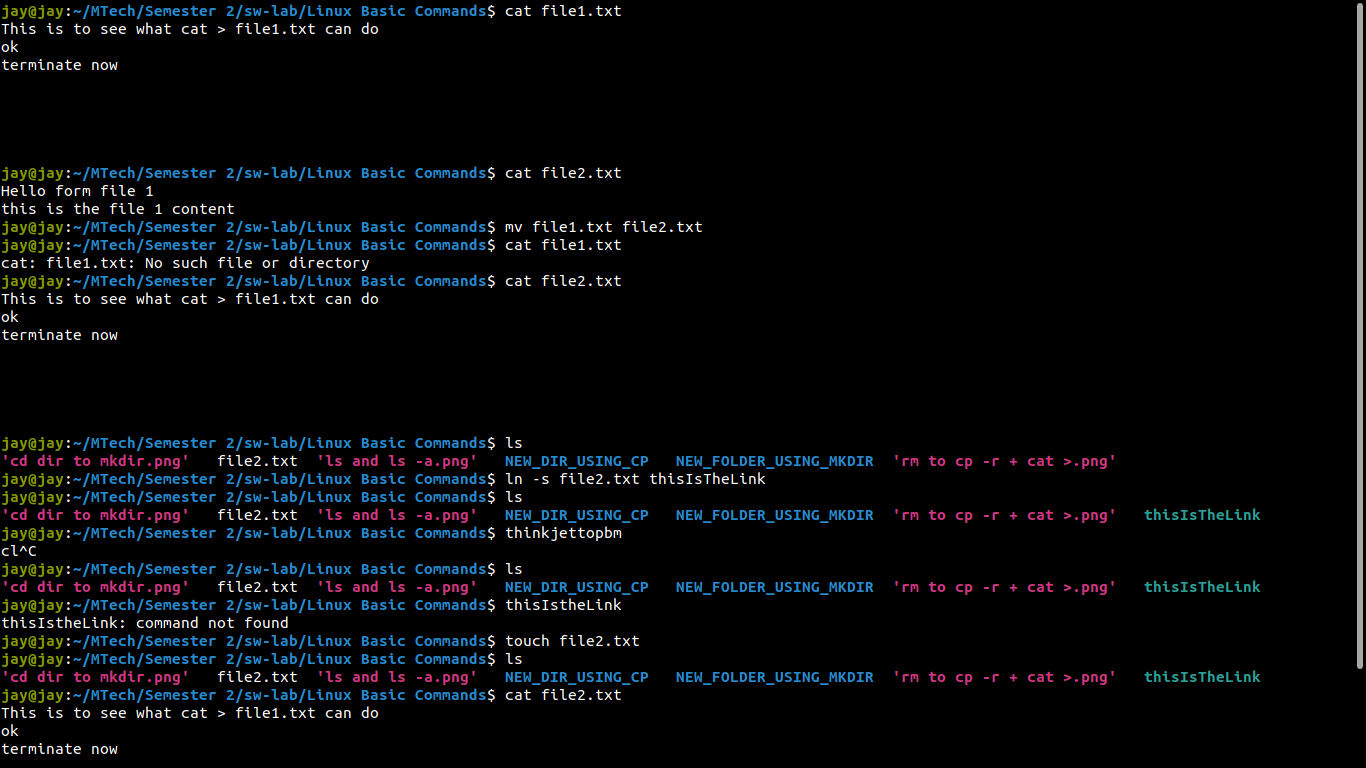
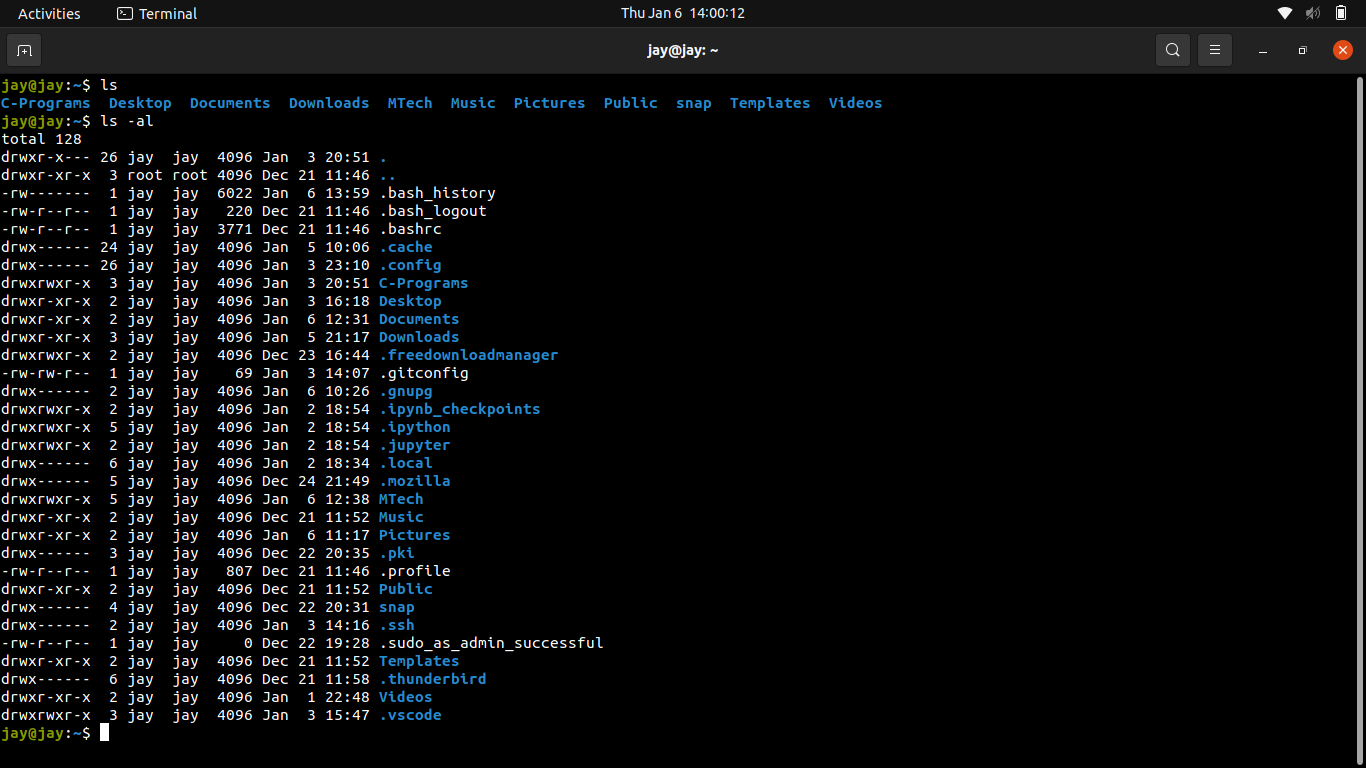
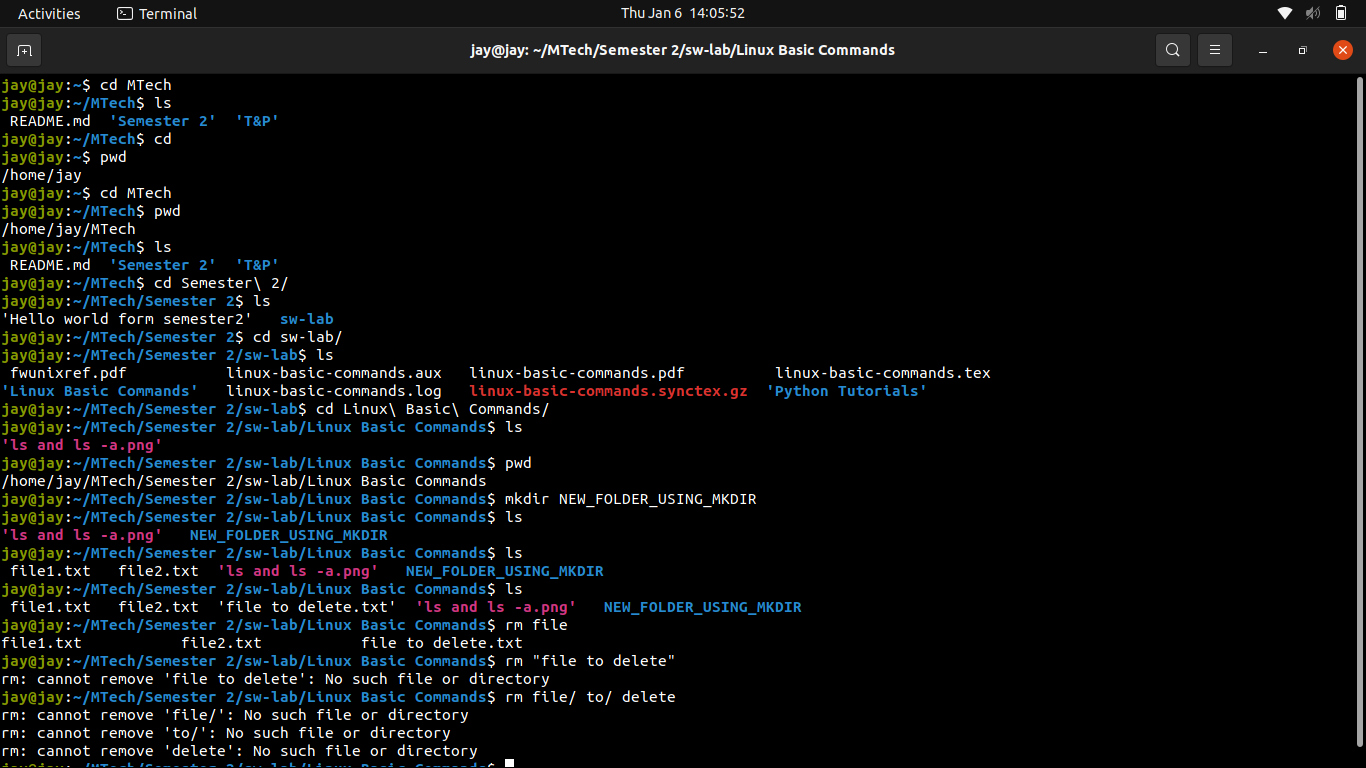
**make install**

Install the software  
Now that the software is built and ready to run, the files can be copied to their final destinations. The make install command will copy the built program, and its libraries and documentation, to the correct locations.  
This usually means that the program’s binary will be copied to a directory on your PATH, the program’s manual page will be copied to a directory on your MANPATH, and any other files it depends on will be safely stored in the appropriate place.  
Since the install step is also defined in the Makefile, where the software is installed can change based on options passed to the configure script, or things the configure script discovered about your system.  
Depending on where the software is being installed, you might need escalated permissions for this step so you can copy files to system directories. Using sudo will often do the trick.

Reference

https://thoughtbot.com/blog/the-magic-behind-configure-make-make-install

* + **-dpkg -i pkg.deb**
    - Install a package with .deb extension
  + **-rpm -Uvh pkg.rpm** 
    - Install a package with .rpm extension

****

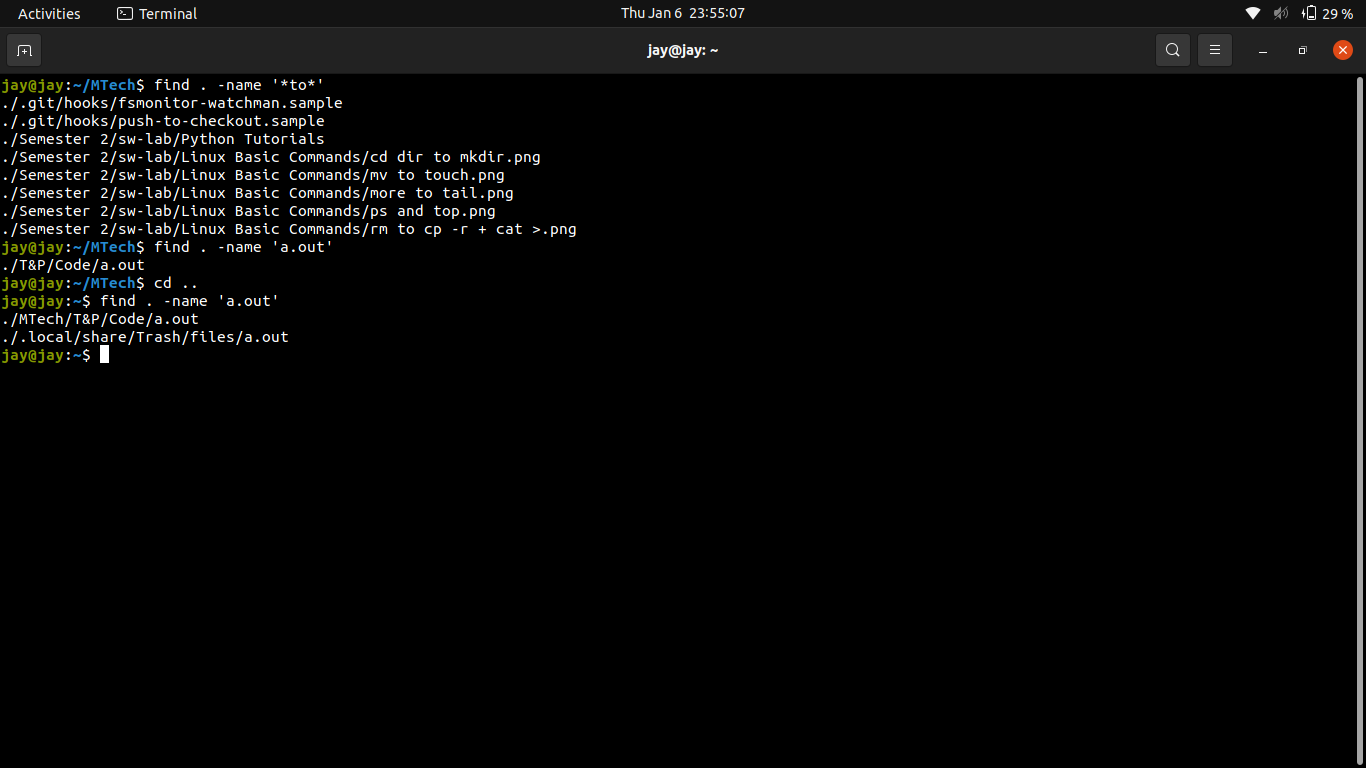
**Search all files containing string "text"**

Using grep command

find . -name ‘\*to\*’ - search all file that contail ‘to’ in file name

**Search all files from / with filename 'a.out',**

find . -name ‘a.out’ - find all files with name a.out



**List all users :**

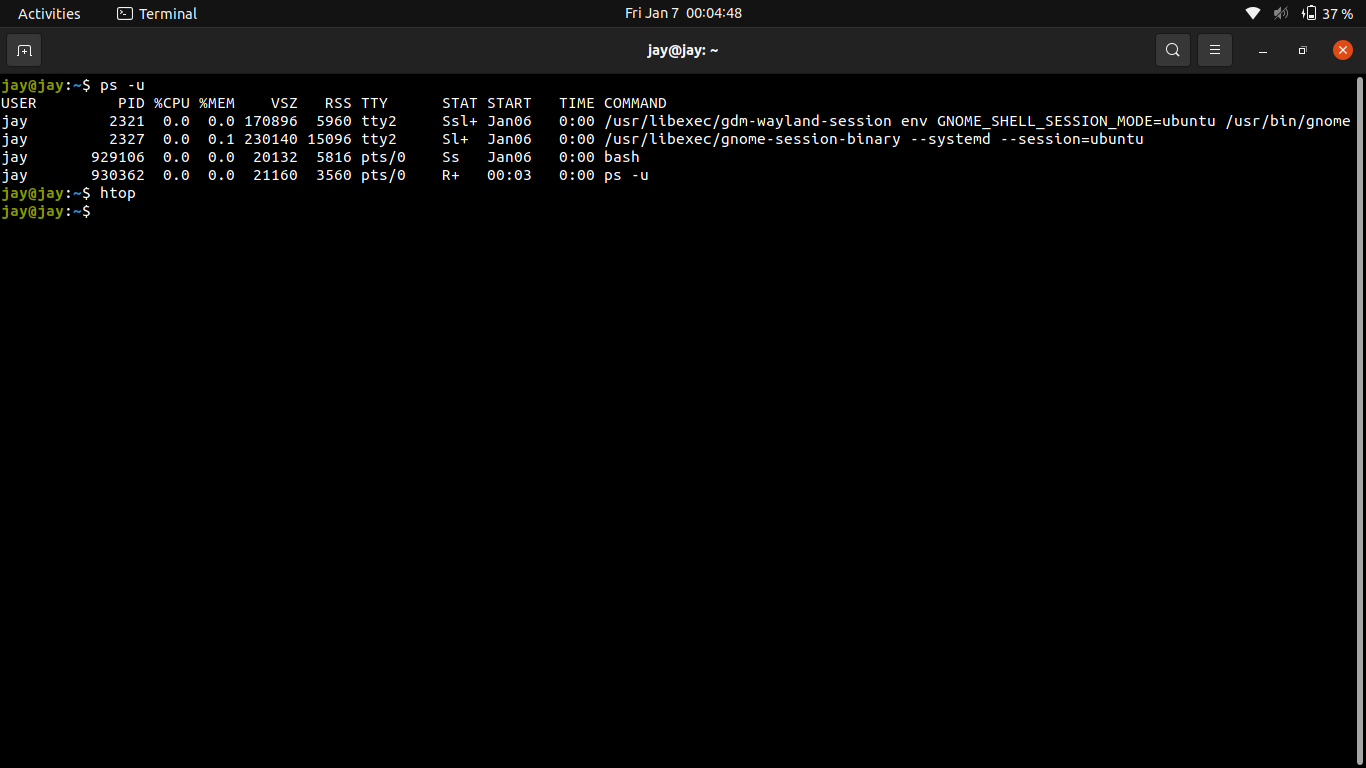
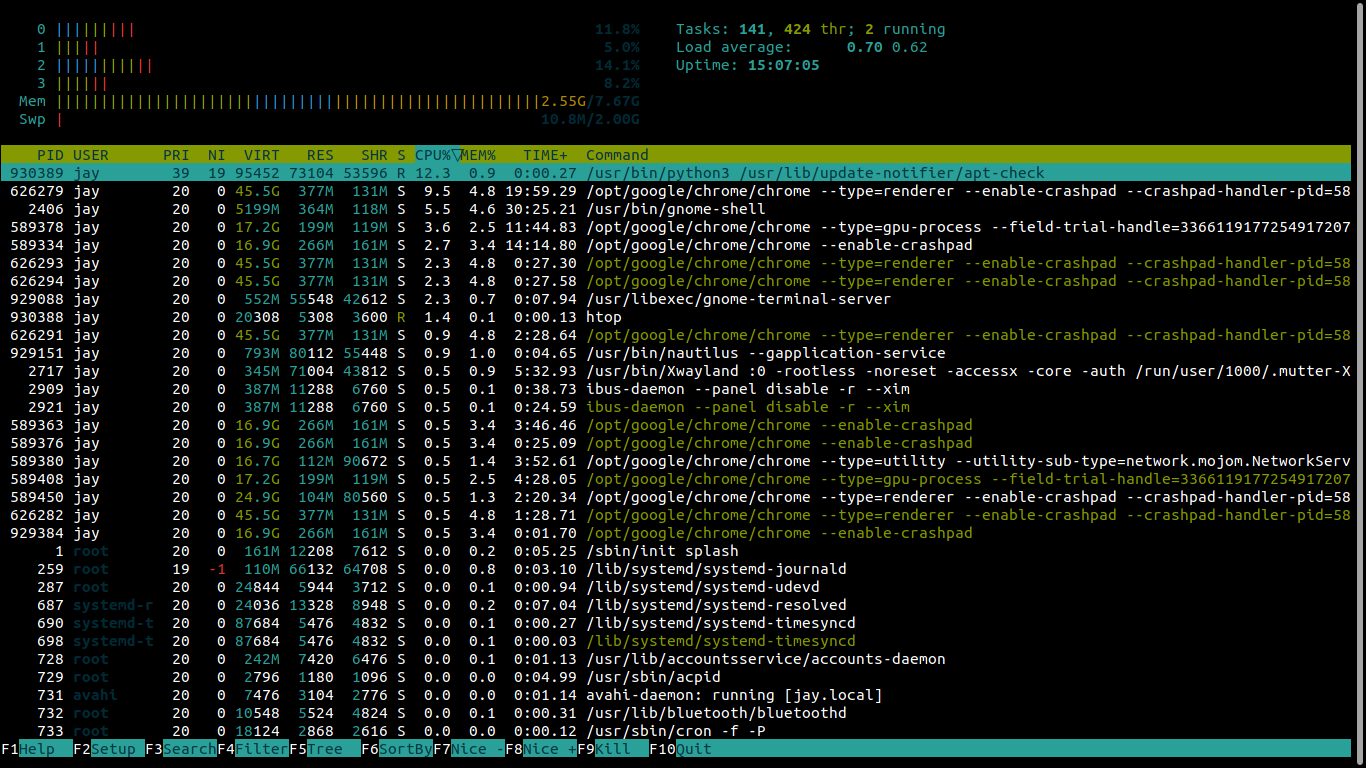
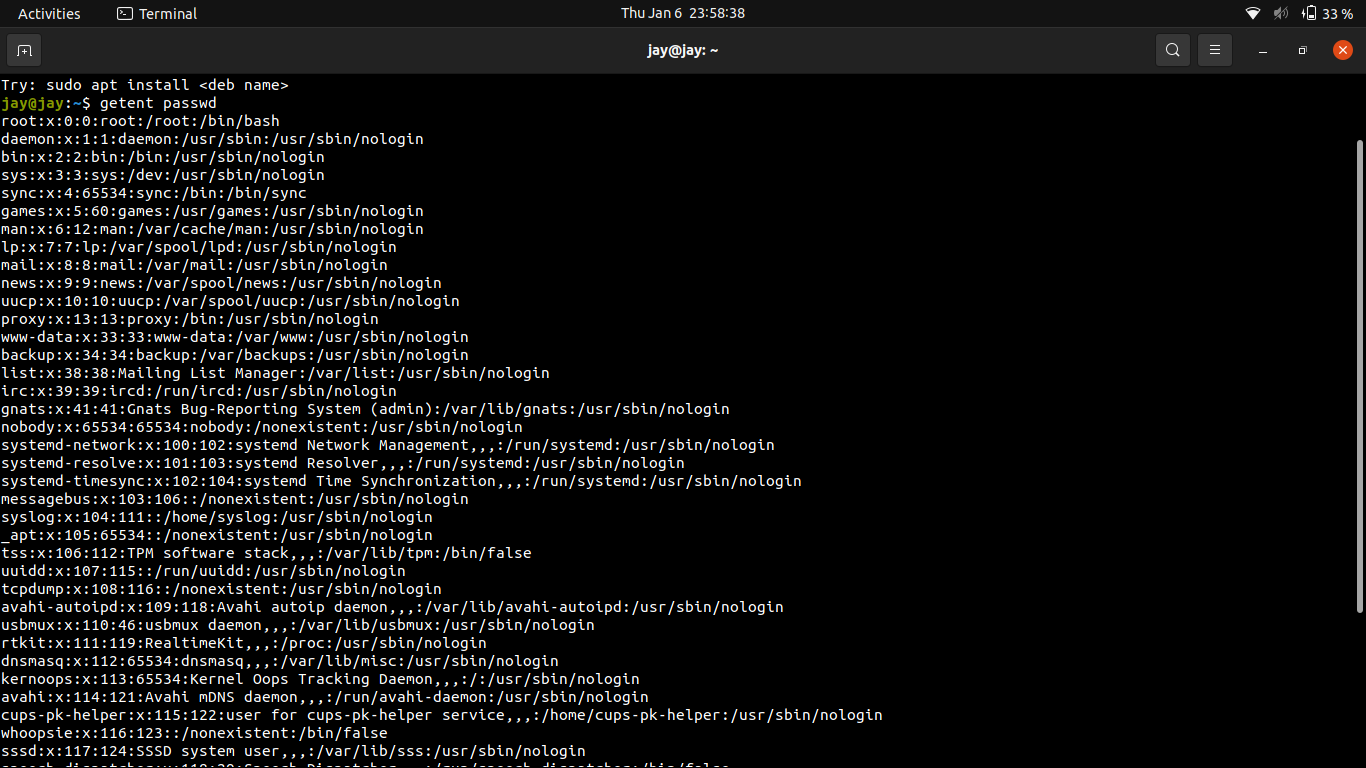
getent passwd

**Process run by user**

ps -u

**Process with high cpu time**

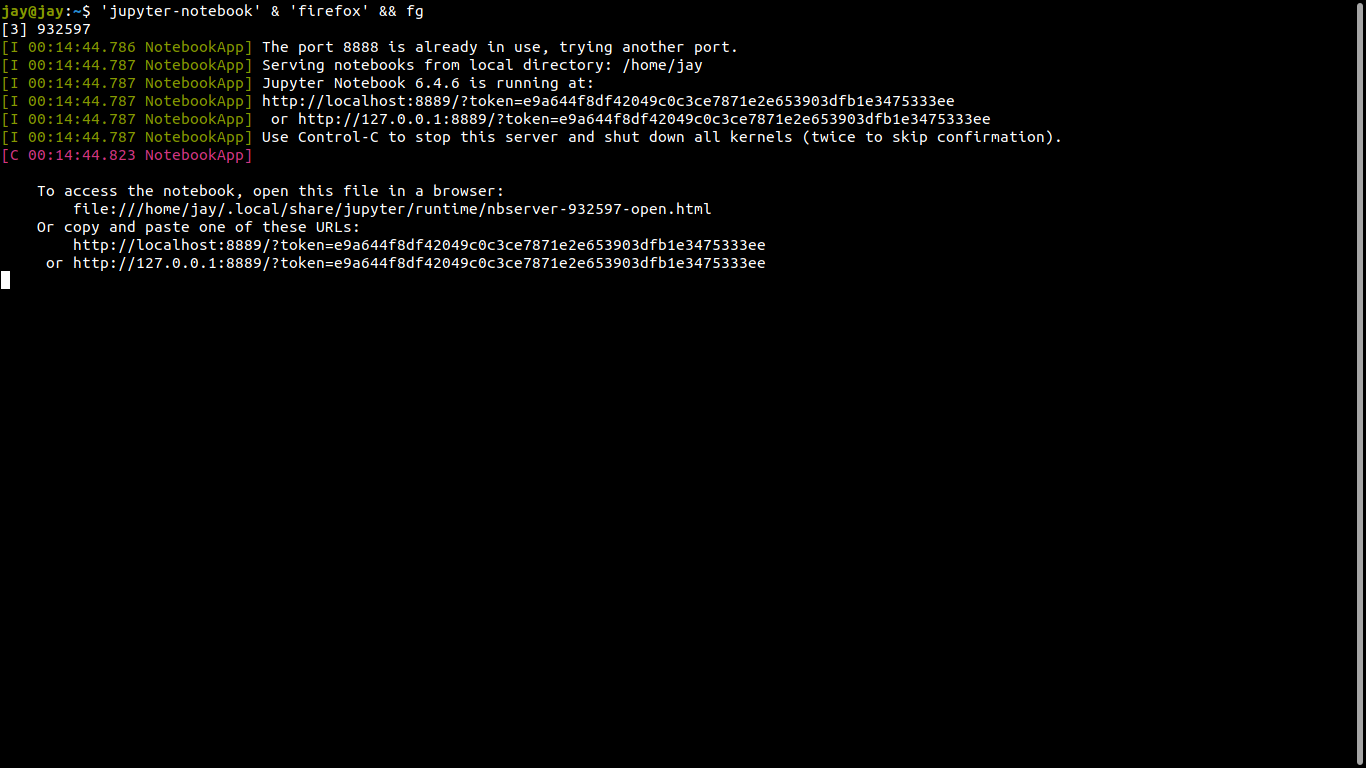
htop



**Run multiple process from terminal in foreground and background**

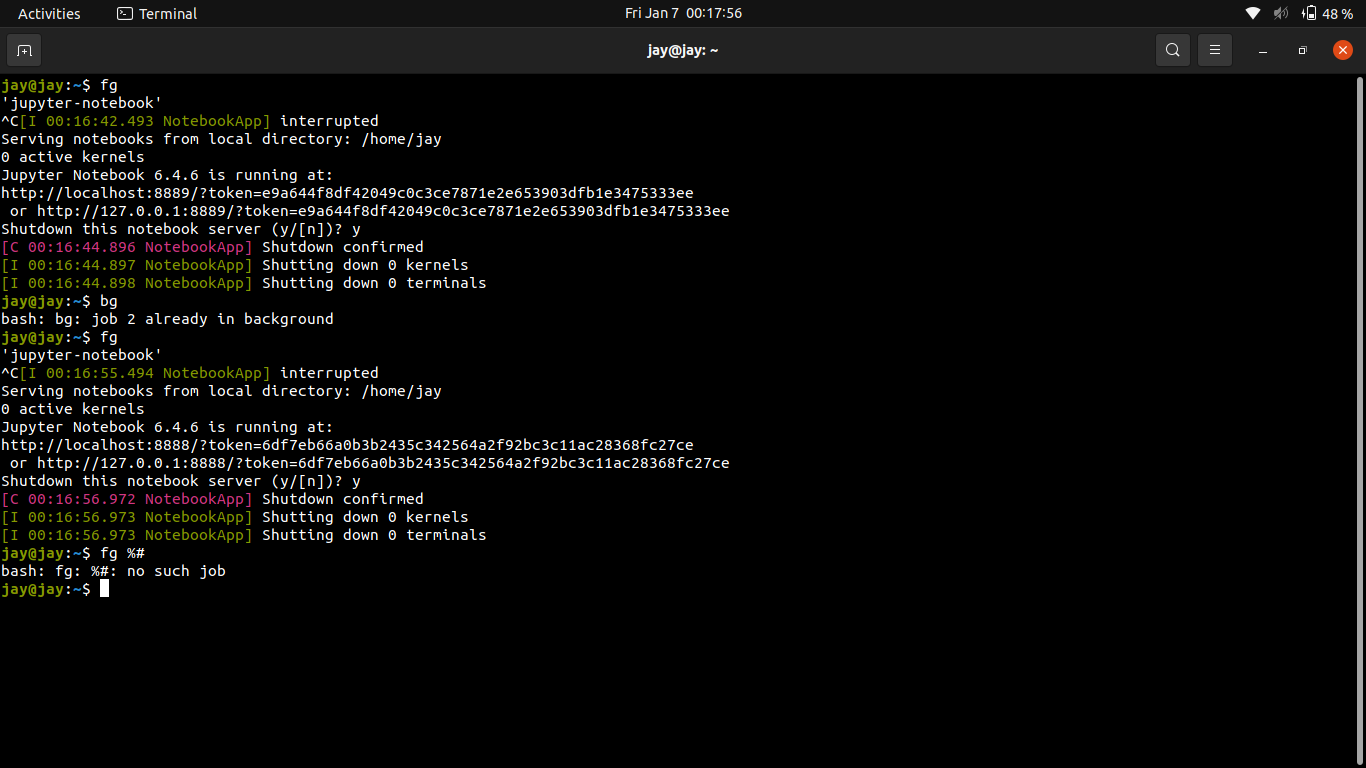
**app1 & app2 && fg for foreground**

**app1 & app2 && bg for background**

****

**To bring a background process to the foreground, enter: fg.**

**More than one job suspended in the background fg %#**

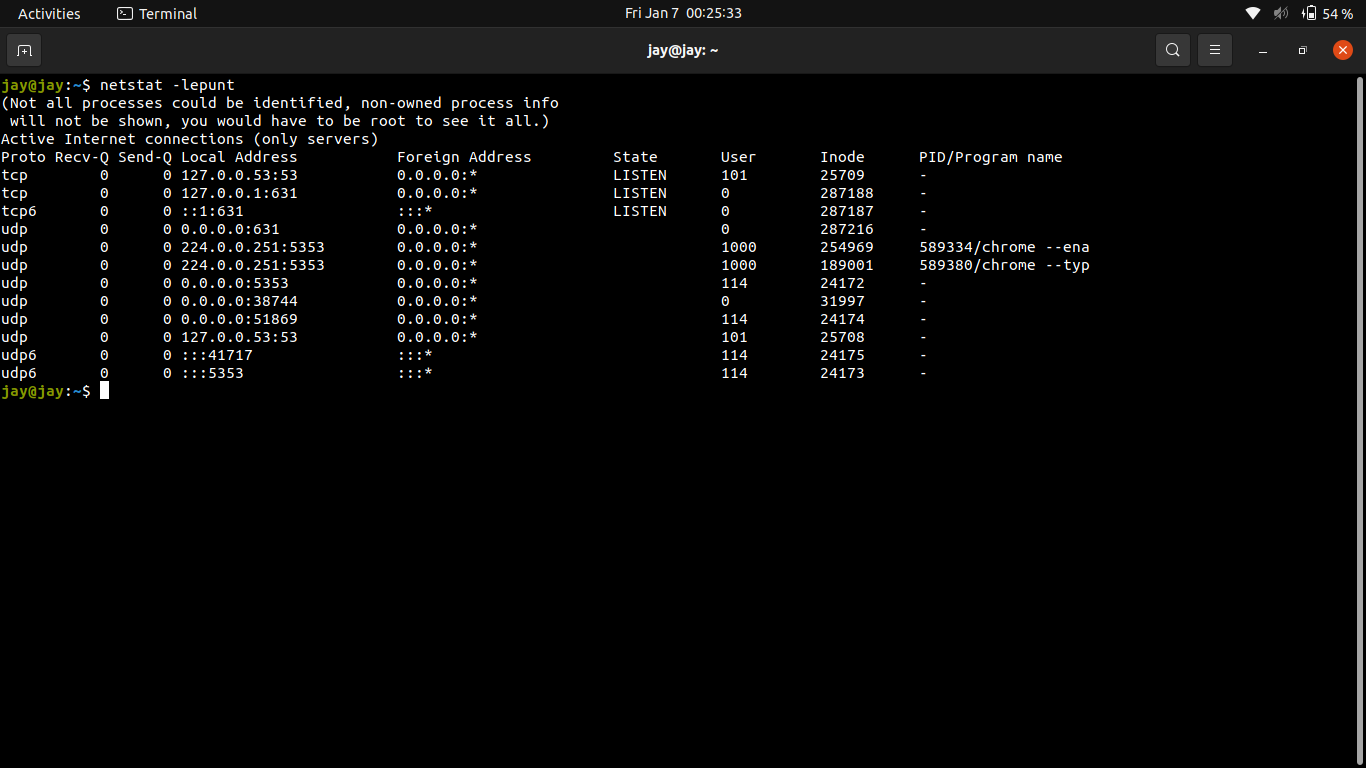
****

**Installation of packages like gcc gzip etc**

Just run : **sudo apt install build-essential**

**To see which servers are online**

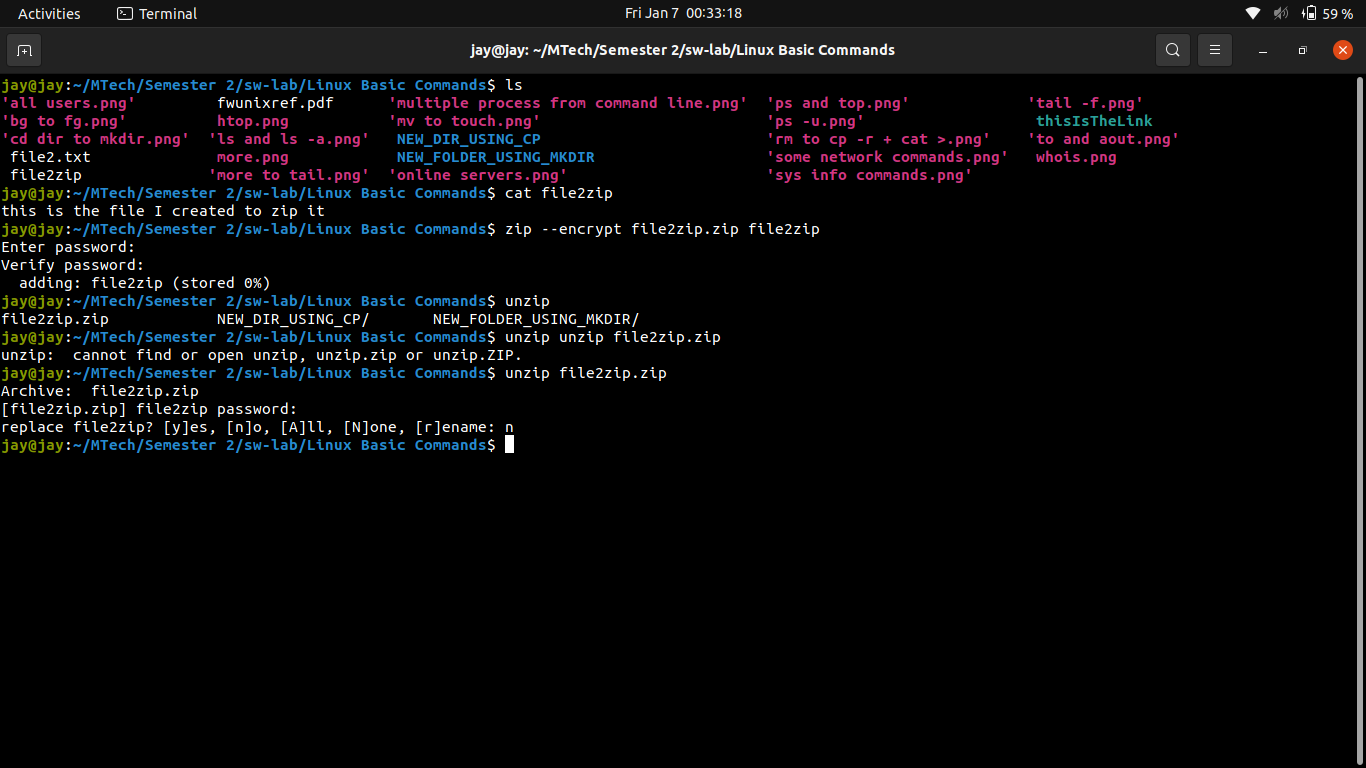
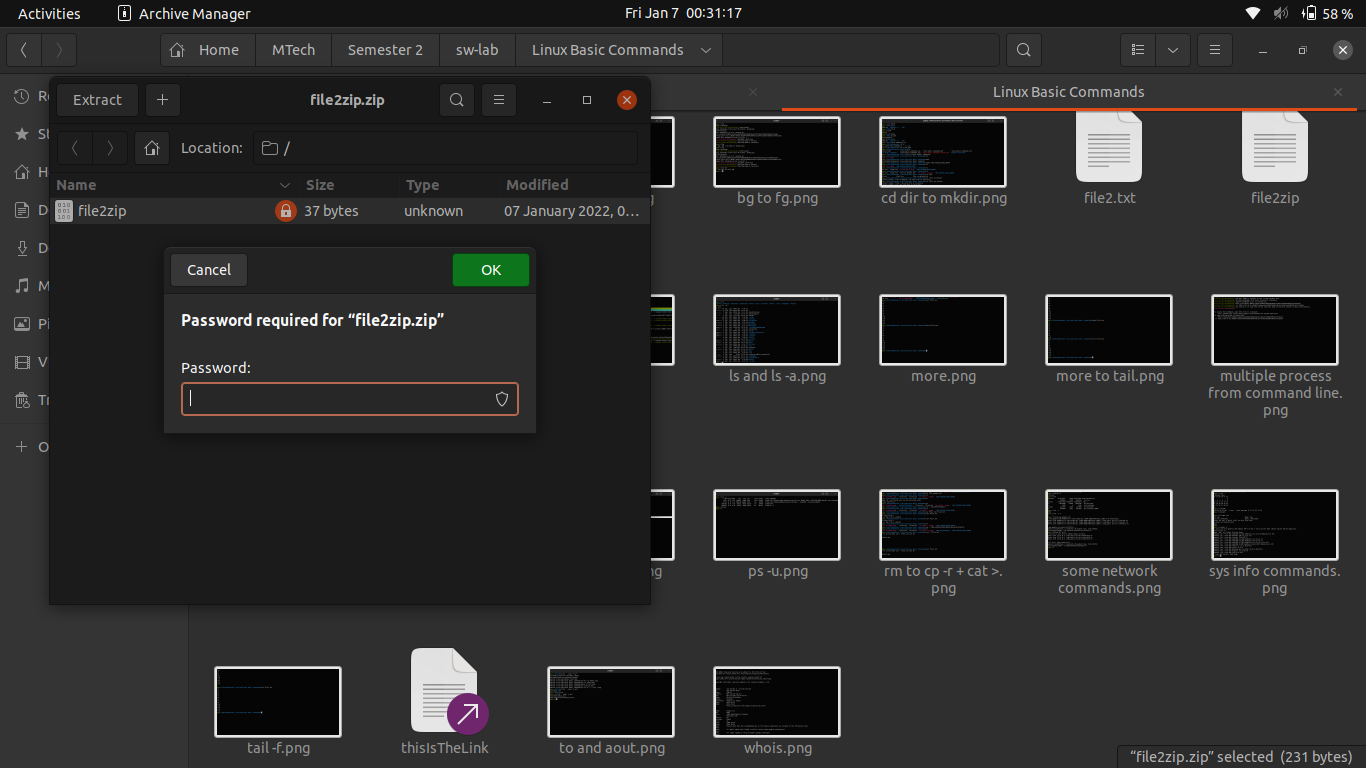
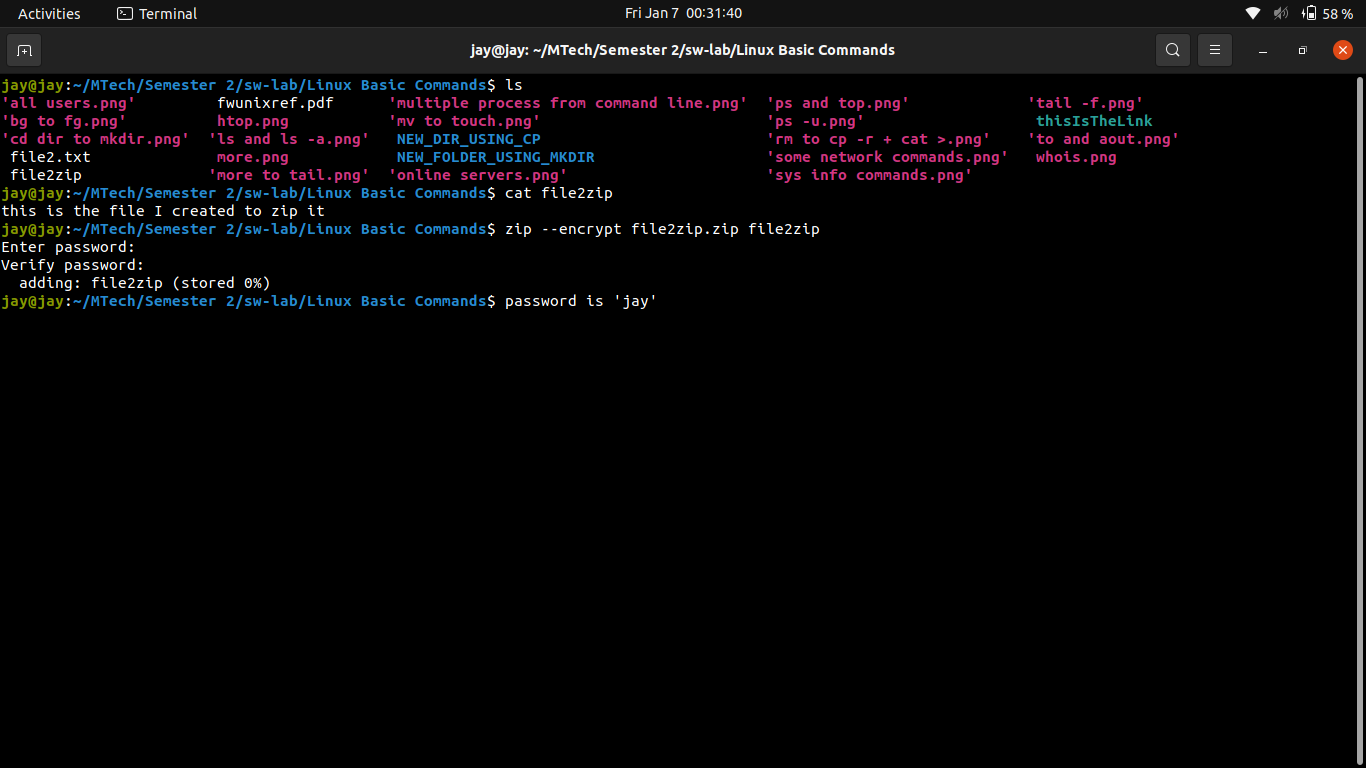
**netstat -lepunt**

****

**Zip and unzip in protected mode**

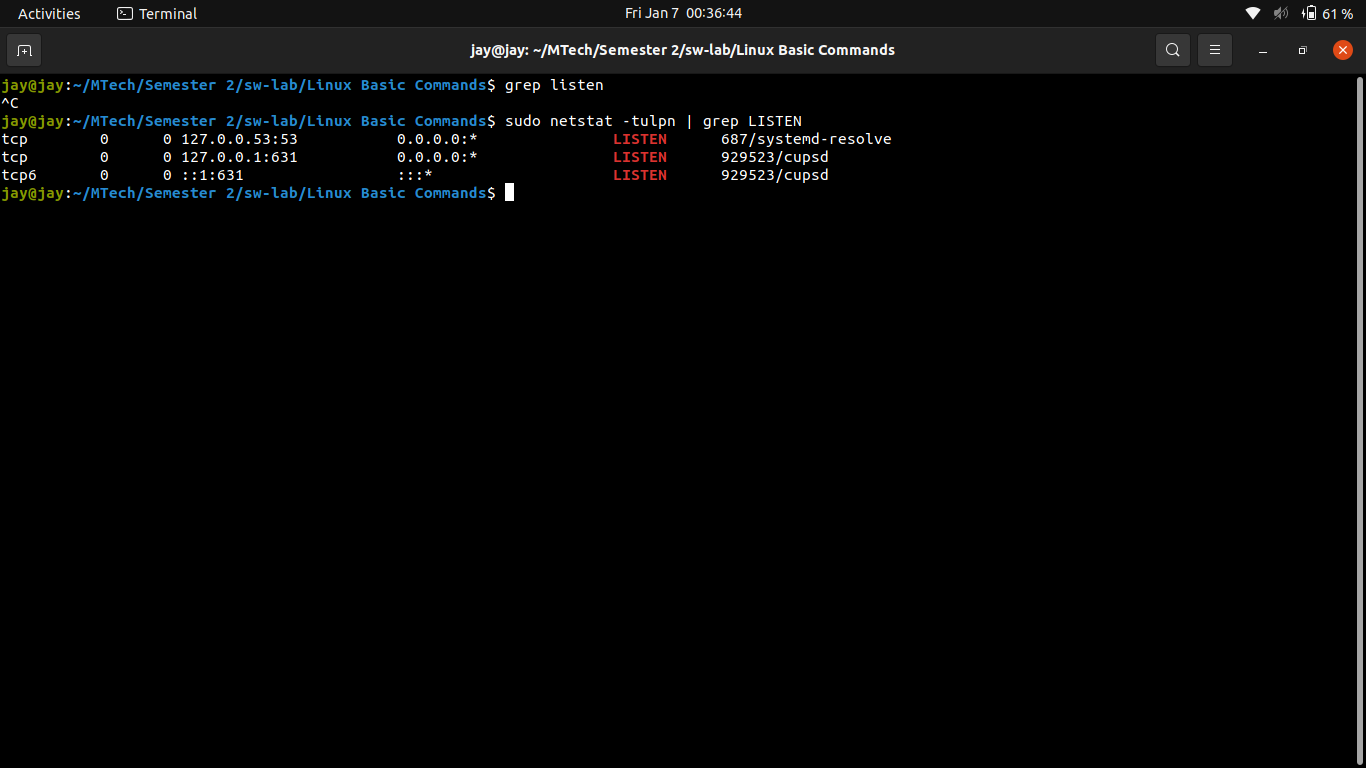
**zip –encrypt file.zip file**

**Then enter password (SS)**

****

**Open ports**

**sudo netstat -tulpn | grep LISTEN**

****

**Other commands like disk size etc are as seen above in file and process commands**