PRACTICAL FILE

LINUX FOR DEVICES

Course Code: CSE438

SUBMITTED TO:
Dr. VIBHA NEHRA

SUBMITTED BY:
Name: V Sri Hrudya
Enrolment No.: A2035221030
B.Tech. 5CSE6-X

AMITY SCHOOL OF ENGINEERING &TECHNOLOGY,
AMITY UNIVERSITY,
NOIDA,
UTTAR PRADESH

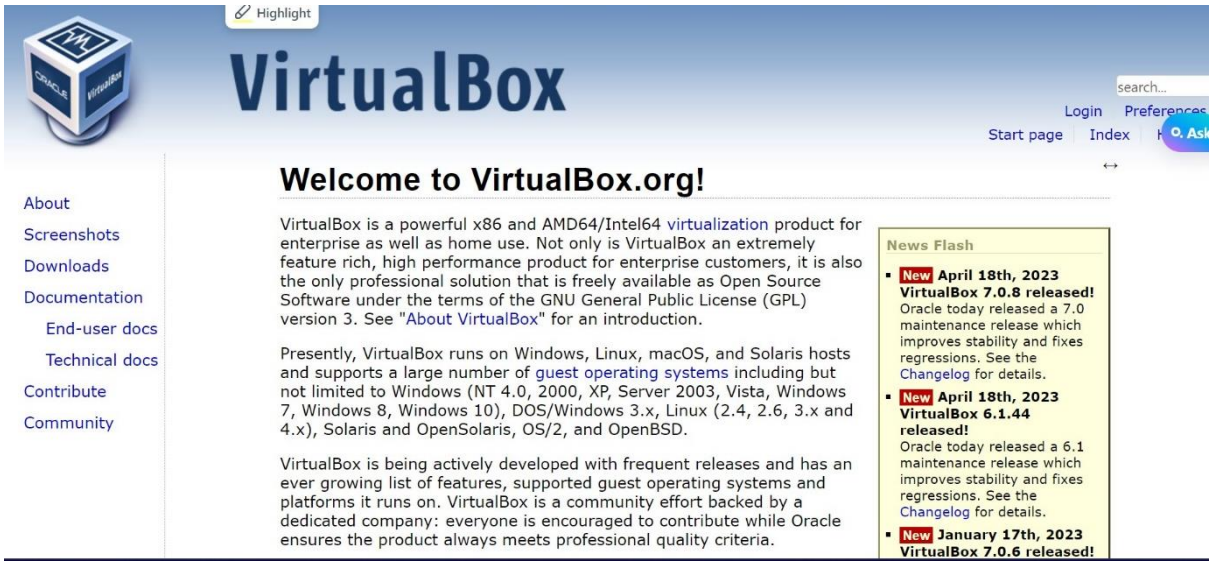
INDEX

LAB 1- USING VIRTUAL BOX TO RUN LINUX

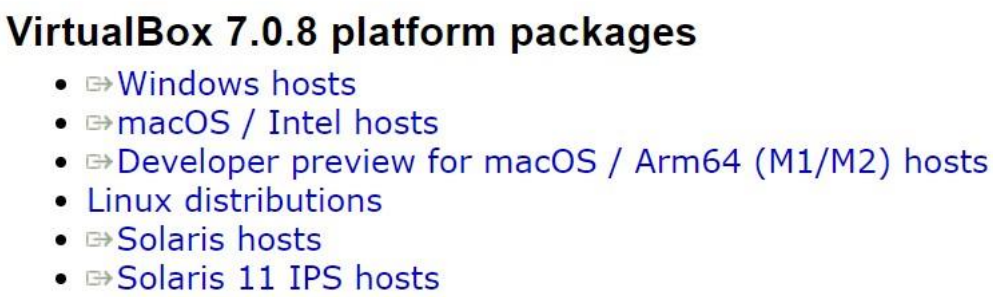
AIM: To download virtual box and load LINUX.

Procedure:

STEP1: Go to https://www.virtualbox.org .



STEP2: Select Downloads from the left navigation pane.

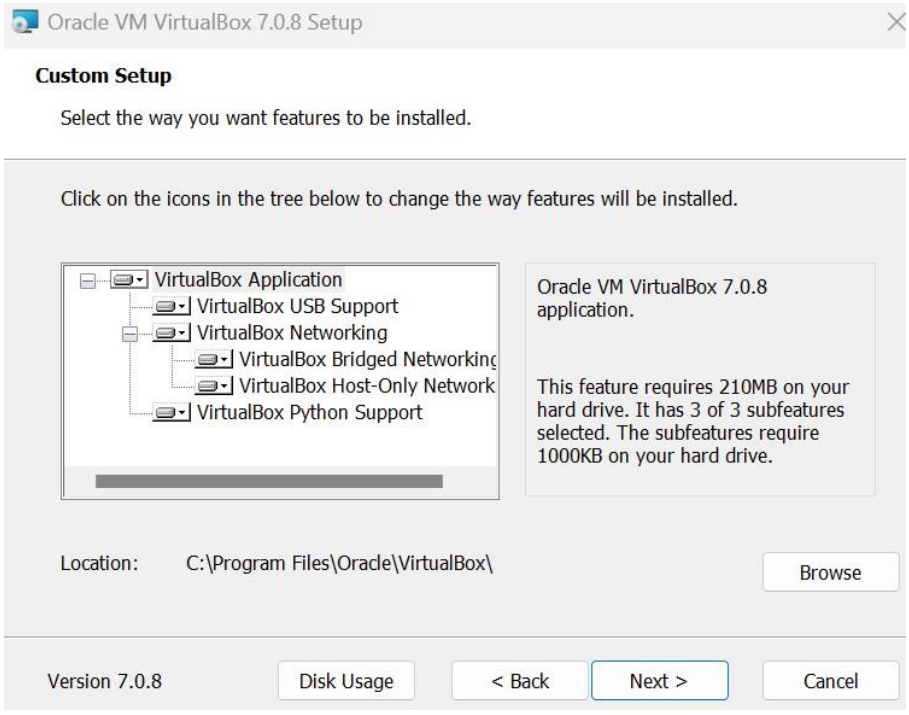


STEP3: Select the host name. The file will get downloaded.

STEP4: Click on the downloaded file and give the required permissions.

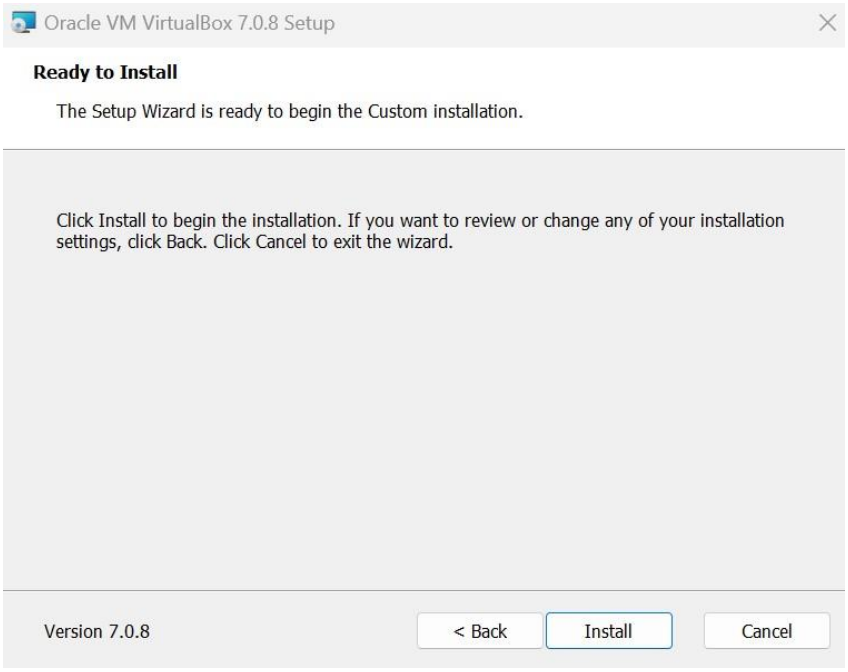
STEP5: Click on Next>

STEP6: Select the features you want to install and click on Next>

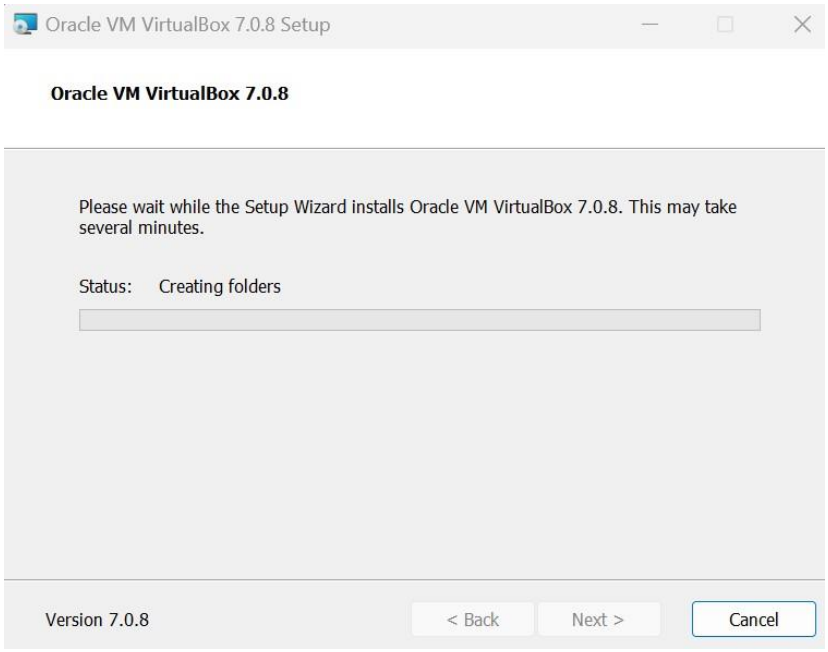


STEP7: Click on Yes.

STEP8: Click on Yes

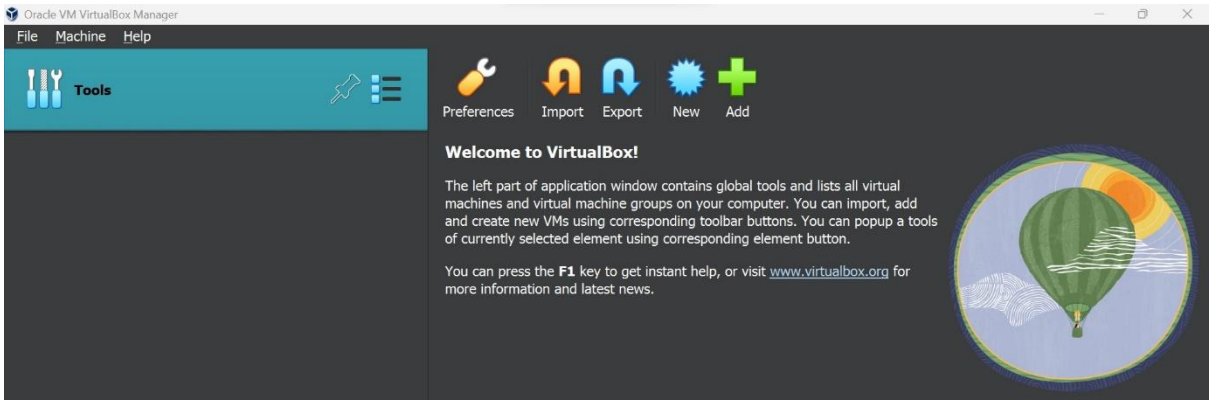


STEP9: After installation click on Finish.

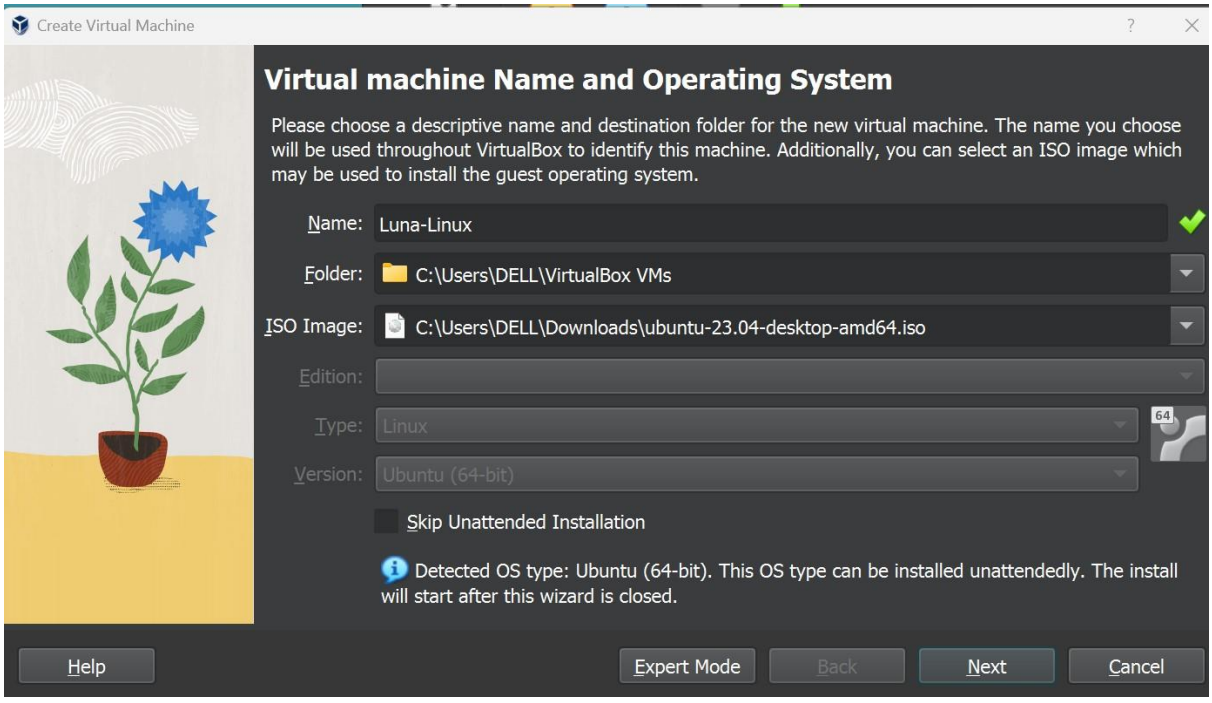


STEP10: Open Virtual Box.

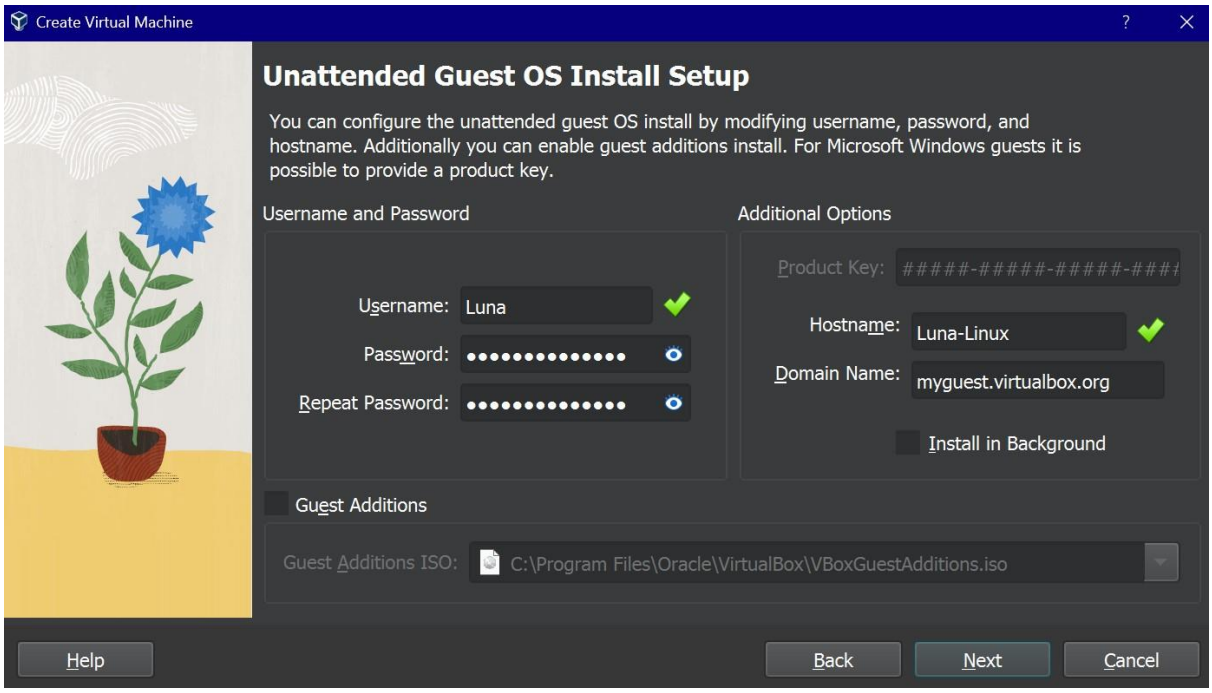
Step11: Click on Add



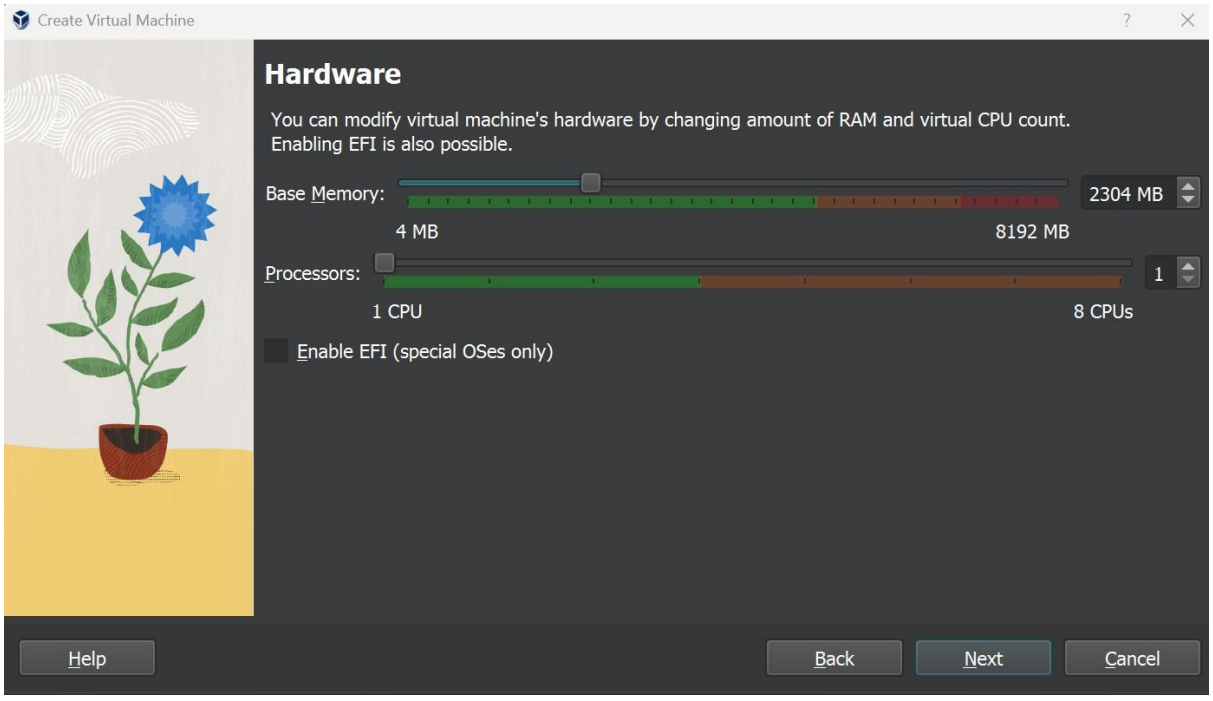
STEP12: Enter the required details.



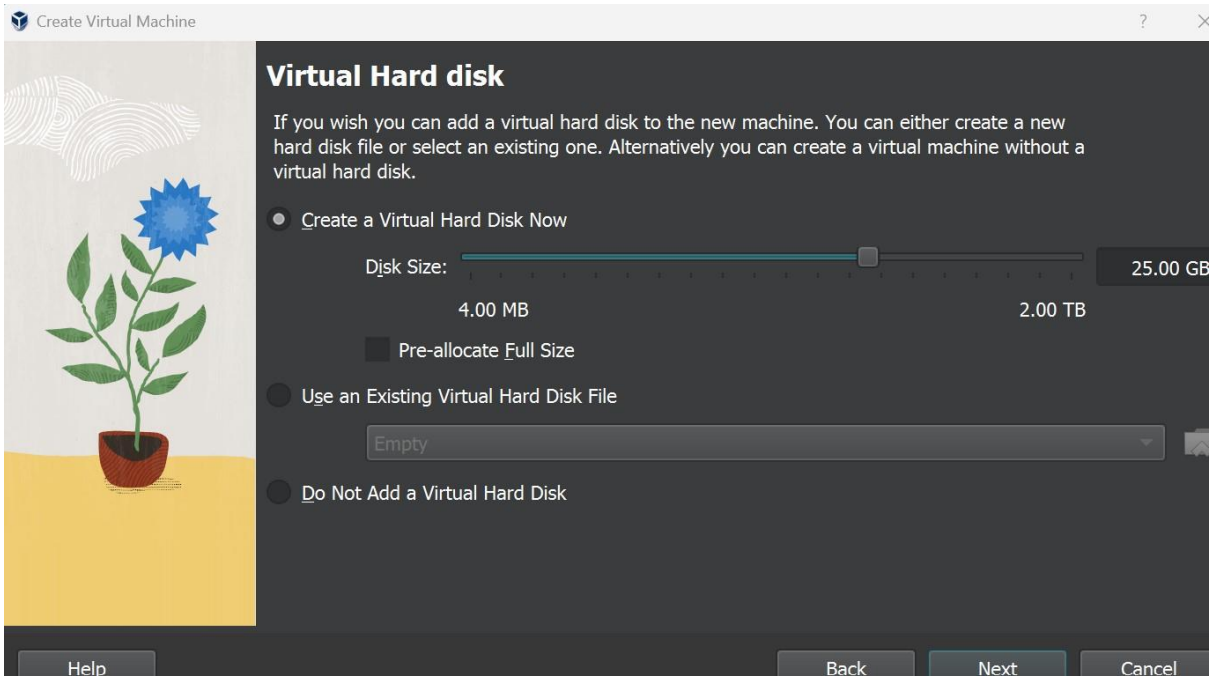
STEP13: Configure the OS by setting username and password.



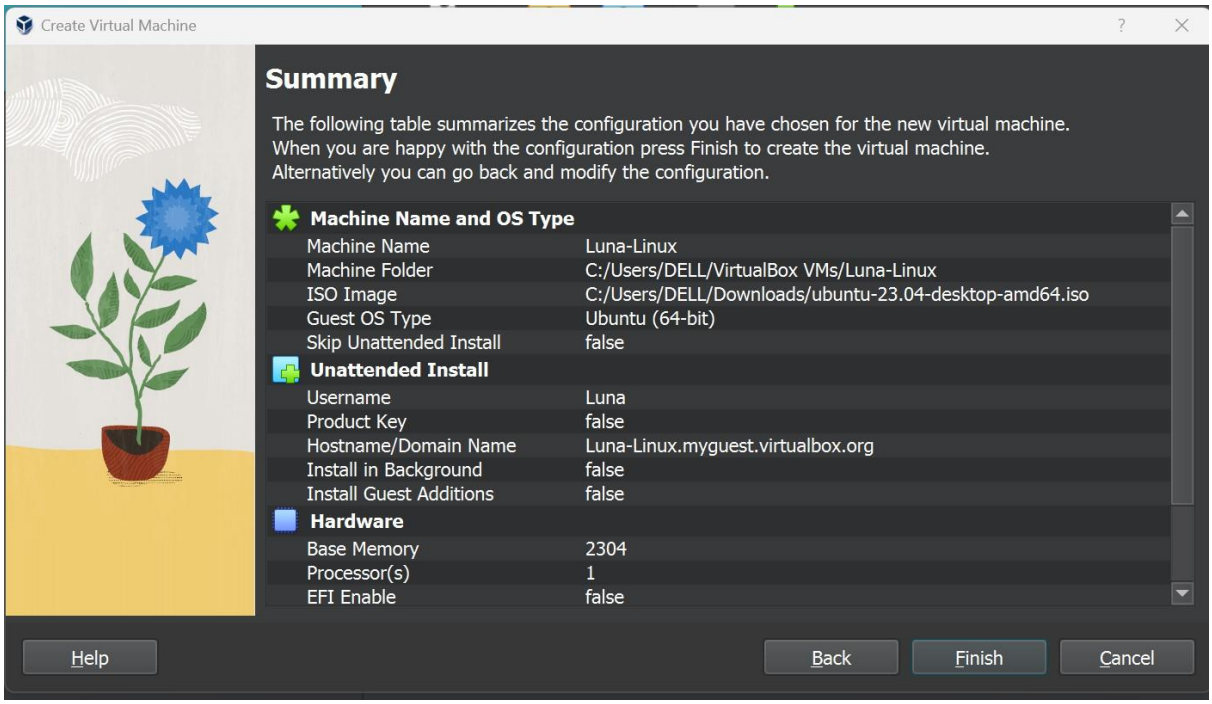
STEP14: Select the machine's hardware.



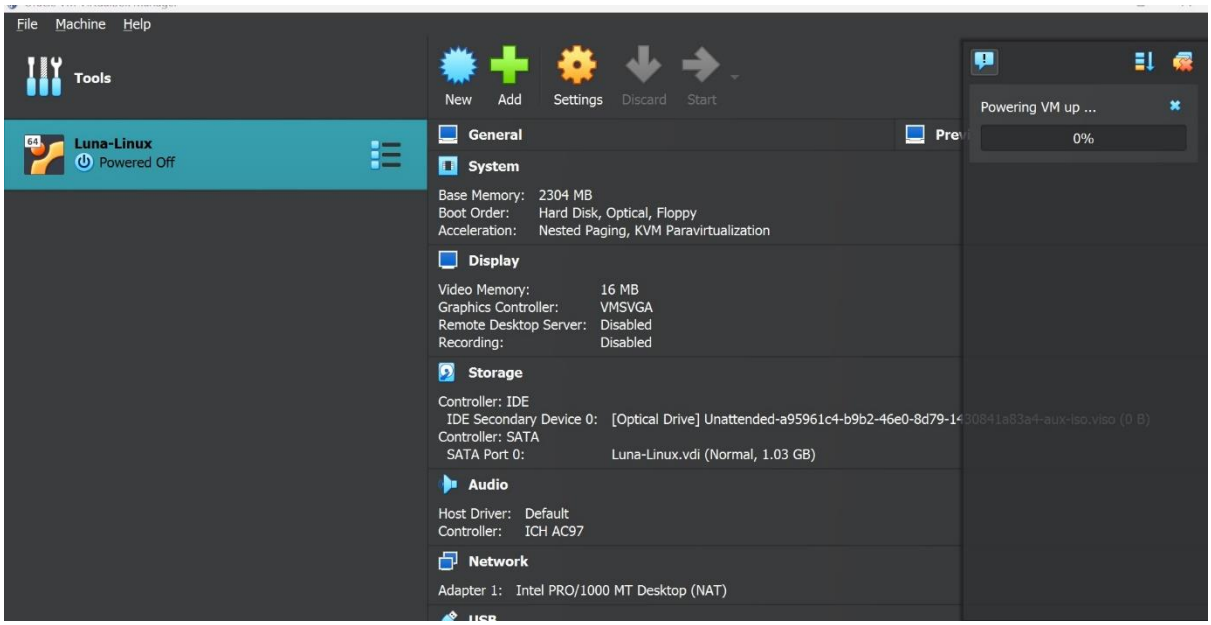
STEP15: Configure the Hard Disk.



STEP16: Go through the summary and make changes (if required) and click on Finish.



STEP17: Run the Virtual Machine.



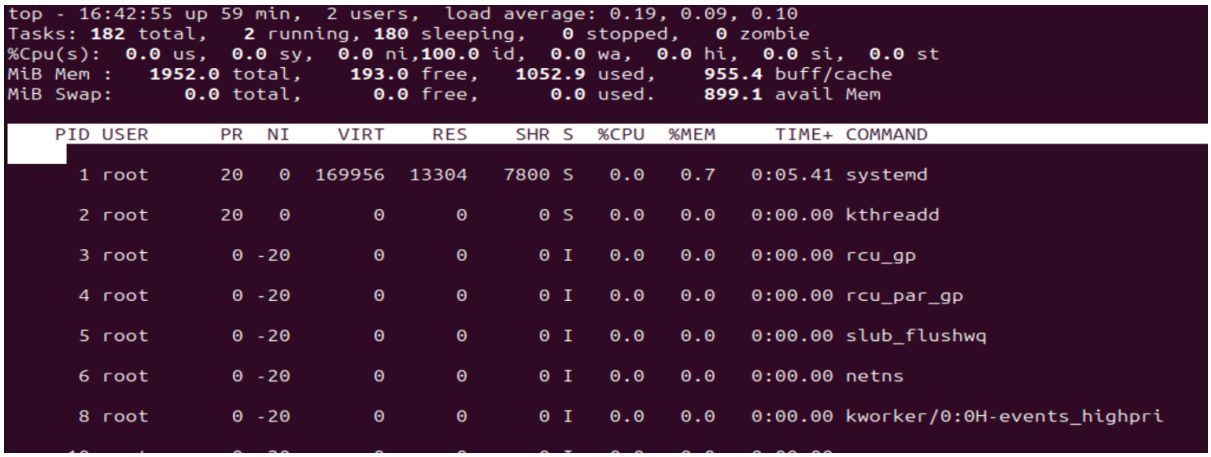
LAB 2.2: BASIC LINUX COMMANDS

AIM: To study basic UNIX/LINUX commands.

THEORY:

1. *top:* Used to show dynamic realtime view of running system.

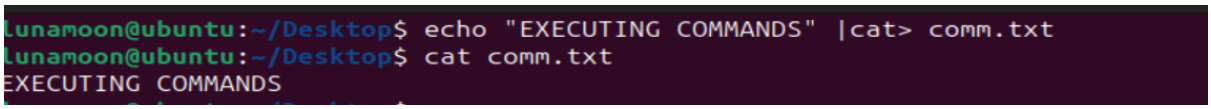
SYNTAX: top



2. **stdin:** Used to take text as input.

SYNTAX: cat 0< [FILE_NAME.txt]

3. **stdout:** Used to text output of any command you type in the terminal.
   SYNTAX: echo "TEXT TO BE DISPLAYED" | cat>[filename].txt



4. **pipes:** *Displays the output of command 1 to the input of command 2.*

```
lunamoon@ubuntu:~/Desktop$ who | sort
lunamoon :1           2023-10-05 16:34 (:1)
lunamoon seat0        2023-10-05 16:33 (login screen)
```

5. **cat:** Used to display the contents of a file.

SYNTAX: cat <FILE_NAME.txt>

```
lunamoon@ubuntu:~/Desktop$ cat hello.txt
HELLO WORLD!!!
```

6. **stderr** : Invoked whenever a command faces an error, then that error message gets stored.

```
lunamoon@ubuntu:~/Desktop$ printf "%s\n" "Hello"
Hello
lunamoon@ubuntu:~/Desktop$ pirntf "%s\n" "Hello" 2> /dev/null
```

7. **wildcard[]:** Used to match characters enclosed in square braces.

*SYNTAX: <command> [character]**

```
lunamoon@ubuntu:~/Desktop$ ls [f]*
f1:

f2:

f3:
lunamoon@ubuntu:~/Desktop$
```

8. **wildcard{}:** Used to specify multiple expressions at once.

*SYNTAX: <command> {*character1,*character2}*

```
lunamoon@ubuntu:~/Desktop$ ls {*.pdf,*.txt}
f1.pdf  f8.txt  hello.txt  l1.txt
```

9. **chgrp:** Used to change group ownership.

SYNTAX: *sudo chgrp <groupname> <filename>*

```
lunamoon@ubuntu:~/Desktop$ sudo chgrp person1 hello.txt
lunamoon@ubuntu:~/Desktop$ ls -l
total 4
-rw-rw-r-- 1 lunamoon lunamoon  0 Oct  5 17:52 f1.pdf
-rw-rw-r-- 1 lunamoon lunamoon  0 Oct  5 17:52 f8.txt
-rw-rw-r-- 1 lunamoon person1  16 Oct  5 16:43 hello.txt
-rw-rw-r-- 1 lunamoon lunamoon  0 Oct  5 17:52 l1.txt
```

10. **and (&&):** Command succeeding this operator will execute only if the previous one executes.

SYNTAX: <operation1> && <operation2>

```
lunamoon@ubuntu:~/Desktop$ touch hel.txt && echo 'This is hrudya
This is hrudya
lunamoon@ubuntu:~/Desktop$ ls
f1.pdf  f8.txt  hello.txt  hel.txt  l1.txt
```

11. **or(||):** Command succeeding this operator will execute even if the previous one does not executes.

SYNTAX: <operation1> || <operation2>

```
lunamoon@ubuntu:~/Desktop$ ls fq.txt || echo 'Hello'
ls: cannot access 'fq.txt': No such file or directory
Hello
```

12. **locate:** Used find file address using name.

SYNTAX: locate <filename>

```
lunamoon@ubuntu:~/Desktop$ locate f1.pdf
/home/lunamoon/Desktop/f1.pdf
```

13. **find:** Used to walk a file hierarchy.

SYNTAX: find <command>

```
lunamoon@ubuntu:~/Desktop$ find -empty
./dir1/f1.pdf
./hel.txt
./f8.txt
./l1.txt
./f1.pdf
```

14. **prune:** Used to exclude a directory within a tree.

```
lunamoon@ubuntu:~/Desktop$ tree
locales-launch: Data of en_US locale not found, generating, please wait...
.
├── dir1
│   └── f1.pdf
├── f1.pdf
├── f8.txt
├── hello.txt
├── hel.txt
└── l1.txt

1 directory, 6 files
lunamoon@ubuntu:~/Desktop$ find -path .fi.pdf -prune -o -print
.
./dir1
./dir1/f1.pdf
./hel.txt
./f8.txt
./l1.txt
./f1.pdf
./hello.txt
lunamoon@ubuntu:~/Desktop$
```

15. **zip:** Used to compress files and reduce their sizes

SYNTAX: zip <zipname> <filenames>

```
lunamoon@ubuntu:~/Desktop$ zip zip1 f8.txt hel.txt
  adding: f8.txt (stored 0%)
  adding: hel.txt (stored 0%)
```

16. **kill:** Used to list and extract files from a zip archive

SYNTAX: unzip <zipname>

```
┌──(luna㉿Luna)-[~/Desktop]
└─$ unzip zip_test
Archive:  zip_test.zip
 extracting: pd.pdf
 extracting: f2.txt
 extracting: g2.txt

┌──(luna㉿Luna)-[~/Desktop]
└─$ ls
WALLPAPER.jpeg  calculator.sh  f2.txt  g2.txt  hello.sh  odd_even.sh  pd.pdf  zip_test.zip
```

17. **gzip:** Used to zip files.

SYNTAX: gzip <filename>

```
┌──(luna㉿Luna)-[~/Desktop]
└─$ gzip f2.txt

┌──(luna㉿Luna)-[~/Desktop]
└─$ tree
.
├── WALLPAPER.jpeg
├── calculator.sh
└── f2.txt.gz
```

18. **bzip2:** Used to compress and decompress files i.e, it helps in better binding into a single file and takes less space than original.

SYNTAX: bzip2 -<option> <filename>

19. **setfacl:** It lets you modify and set access control to regular files and directories.

SYNTAX: setfacl <option> u:<host>:r <filename>

```
┌──(luna㉿Luna)-[~/Desktop]
└─$ setfacl -m u:luna:r demo.txt
```

20. **getfacl:** Used to get control list.

SYNTAX: getfacl <filename>

```
┌──(luna㉿Luna)-[~/Desktop]
└─$ getfacl demo.txt
# file: demo.txt
# owner: luna
# group: luna
user::rw-
user:luna:r--
group::r--
mask::r--
other::r--
```

LAB 2: BASIC LINUX COMMANDS

AIM: To study basic UNIX/LINUX commands.

THEORY:

1. *mkdir:* Used to create a new directory.

SYNTAX: mkdir <Directory_Name>

```
lunamoon@ubuntu:~/Desktop$ mkdir Main
```

2. cd: Used to open a directory or file.

SYNTAX: cd <Directory/File_Name>

```
lunamoon@ubuntu:~/Desktop$ cd Main
```

3. pwd: Used to display the address of currently working directory.

SYNTAX: pwd

```
lunamoon@ubuntu:~/Desktop/Main$ pwd
/home/lunamoon/Desktop/Main
```

4. date: Used to display the date.

SYNTAX: date

```
lunamoon@ubuntu:~/Desktop/Main$ date
Thu Jul 13 03:58:12 PM IST 2023
```

5. time: Used to display the information about used resources and time by the given command.

SYNTAX: time <Command_Name>

```
lunamoon@ubuntu:~/Desktop/Luna$ time cat

real    0m6.764s
user    0m0.004s
sys     0m0.000s
```

6. history: Used to display various commands performed.

SYNTAX: history

```
lunamoon@ubuntu:~/Desktop/Main$ history
    1  man
    2  Man
    3  rv
    4  rm
    5  rm --help
    6  mkdir Luna
    7  cd Luna
    8  cat f1
    9  echo
   10  more
   11  date
   12  time
   13  kill
```

7. shutdown: Used to shut down a system.

SYNTAX: shutdown

```
lunamoon@ubuntu:~/Desktop/Main$ shutdown
Shutdown scheduled for Thu 2023-07-13 16:12:34 IST, use 'shutdown -c' to cancel.
```

8. rmdir: Used to remove empty directories.

SYNTAX: rmdir <Directory_Name>

```
lunamoon@ubuntu:~/Desktop$ mkdir Main
lunamoon@ubuntu:~/Desktop$ ls
Main
lunamoon@ubuntu:~/Desktop$ rmdir Main
lunamoon@ubuntu:~/Desktop$ ls
lunamoon@ubuntu:~/Desktop$
```

9. cp: Used to copy files.

SYNTAX: cp

```
lunamoon@ubuntu:~/Desktop/Luna/dir1$ cp --help
Usage: cp [OPTION]... [-T] SOURCE DEST
  or:  cp [OPTION]... SOURCE... DIRECTORY
  or:  cp [OPTION]... -t DIRECTORY SOURCE...
Copy SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY.

Mandatory arguments to long options are mandatory for short options too.
  -a, --archive                same as -dR --preserve=all
      --attributes-only        don't copy the file data, just the attributes
      --backup[=CONTROL]       make a backup of each existing destination file
  -b                           like --backup but does not accept an argument
      --copy-contents          copy contents of special files when recursive
  -d                           same as --no-dereference --preserve=links
  -f, --force                  if an existing destination file cannot be
                                 opened, remove it and try again (this option
                                 is ignored when the -n option is also used)
  -i, --interactive            prompt before overwrite (overrides a previous -n
                                  option)
  -H                           follow command-line symbolic links in SOURCE
  -l, --link                   hard link files instead of copying
  -L, --dereference            always follow symbolic links in SOURCE
  -n, --no-clobber             do not overwrite an existing file (overrides
                                 a previous -i option)
  -P, --no-dereference         never follow symbolic links in SOURCE
  -p                           same as --preserve=mode,ownership,timestamps
```

10. **cat:** Used to edit files.

SYNTAX: cat <Operation_To_Be_Done> File_Name.txt

```
lunamoon@ubuntu:~/Desktop/Luna/dir1$ cat --help
Usage: cat [OPTION]... [FILE]...
Concatenate FILE(s) to standard output.

With no FILE, or when FILE is -, read standard input.

  -A, --show-all           equivalent to -vET
  -b, --number-nonblank    number nonempty output lines, overrides -n
  -e                       equivalent to -vE
  -E, --show-ends          display $ at end of each line
  -n, --number             number all output lines
  -s, --squeeze-blank      suppress repeated empty output lines
  -t                       equivalent to -vT
  -T, --show-tabs          display TAB characters as ^I
  -u                       (ignored)
  -v, --show-nonprinting   use ^ and M- notation, except for LFD and TAB
      --help        display this help and exit
      --version     output version information and exit
```

11. **man:** Used to display the user manual of any command that we can run on the terminal.

SYNTAX: man <Command_Name>

```
lunamoon@ubuntu:~/Desktop/Luna/dir1$ man cp
```

```
CP(1)                                                User Commands

NAME
       cp - copy files and directories

SYNOPSIS
       cp [OPTION]... [-T] SOURCE DEST
       cp [OPTION]... SOURCE... DIRECTORY
       cp [OPTION]... -t DIRECTORY SOURCE...

DESCRIPTION
       Copy SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY.

       Mandatory arguments to long options are mandatory for short options too.
```

12. **ps:** Writes the states of active processes.

SYNTAX: ps

```
lunamoon@ubuntu:~/Desktop/Luna/dir1$ ps
    PID TTY          TIME CMD
   5373 pts/0    00:00:00 bash
   5765 pts/0    00:00:00 ps
```

13. **ls:** Lists the names of files in a particular directory.

SYNTAX: ls

```
lunamoon@ubuntu:~/Desktop/Luna$ ls
dir1  f1.txt
```

14. **rm:** Used to remove a flie.

SYNTAX: rm <Directory_Name>

```
lunamoon@ubuntu:~/Desktop/Luna$ ls
dir1  f1.txt
lunamoon@ubuntu:~/Desktop/Luna$ rm f1.txt
lunamoon@ubuntu:~/Desktop/Luna$ ls
dir1
```

15. **more:** Used to read files and display text one screen at a time.

SYNTAX: more <File_Name>

```
lunamoon@ubuntu:~/Desktop/Luna$ more f1.txt
Hello
This is Luna
How do you do???????
```

16. **kill:** Used to terminate processes manually.

SYNTAX: kill <Signal> PID

```
lunamoon@ubuntu:~/Desktop/Luna$ kill --help
kill: kill [-s sigspec | -n signum | -sigspec] pid | jobspec ... or kill -l [sigspec]
    Send a signal to a job.

    Send the processes identified by PID or JOBSPEC the signal named by
    SIGSPEC or SIGNUM.  If neither SIGSPEC nor SIGNUM is present, then
    SIGTERM is assumed.

    Options:
      -s sig    SIG is a signal name
      -n sig    SIG is a signal number
      -l        list the signal names; if arguments follow `-l' they are
                assumed to be signal numbers for which names should be listed
      -L        synonym for -l
```

17. **chmod:** Used to change permissions of a file.

SYNTAX: chmod <Options> <Mode> <File_Name>

```
lunamoon@ubuntu:~/Desktop/Luna$ chmod --help
Usage: chmod [OPTION]... MODE[,MODE]... FILE...
  or:  chmod [OPTION]... OCTAL-MODE FILE...
  or:  chmod [OPTION]... --reference=RFILE FILE...
Change the mode of each FILE to MODE.
With --reference, change the mode of each FILE to that of RFILE.

  -c, --changes          like verbose but report only when a change is made
  -f, --silent, --quiet  suppress most error messages
  -v, --verbose          output a diagnostic for every file processed
      --no-preserve-root  do not treat '/' specially (the default)
      --preserve-root    fail to operate recursively on '/'
      --reference=RFILE  use RFILE's mode instead of MODE values
  -R, --recursive        change files and directories recursively
      --help         display this help and exit
      --version      output version information and exit
```

18. **chown:** Used to change the file owner or group.

SYNTAX: chown <Option>… <Owner><:Group> File

```
lunamoon@ubuntu:~/Desktop/Luna$ chown --help
Usage: chown [OPTION]... [OWNER][:[GROUP]] FILE...
  or:  chown [OPTION]... --reference=RFILE FILE...
Change the owner and/or group of each FILE to OWNER and/or GROUP.
With --reference, change the owner and group of each FILE to those of RFILE.

  -c, --changes          like verbose but report only when a change is made
  -f, --silent, --quiet  suppress most error messages
  -v, --verbose          output a diagnostic for every file processed
      --dereference      affect the referent of each symbolic link (this is
                         the default), rather than the symbolic link itself
  -h, --no-dereference   affect symbolic links instead of any referenced file
                         (useful only on systems that can change the
                         ownership of a symlink)
      --from=CURRENT_OWNER:CURRENT_GROUP
                         change the owner and/or group of each file only if
                         its current owner and/or group match those specified
                         here.  Either may be omitted, in which case a match
                         is not required for the omitted attribute
      --no-preserve-root  do not treat '/' specially (the default)
      --preserve-root    fail to operate recursively on '/'
      --reference=RFILE  use RFILE's owner and group rather than
                         specifying OWNER:GROUP values
  -R, --recursive        operate on files and directories recursively
```

19. **who:** It lets you display the users currently logged in to your UNIX or Linux operating system.

SYNTAX: who

```
lunamoon@ubuntu:~/Desktop/Luna$ who
lunamoon seat0        2023-07-13 15:27 (login screen)
lunamoon :1           2023-07-13 15:27 (:1)
```

20. **exit:** Used to exit Linux terminal.

SYNTAX: exit

```
lunamoon@ubuntu:~/Desktop/Luna$ exit
```

LAB 3: FILE TRANSFER PROTOCOL

AIM: To setup FTP server in Ubuntu in Virtual Box.

MACHINE 1:

```
lunamoon@ubuntu:~/Desktop$ sudo apt update
[sudo] password for lunamoon:
Ign:1 cdrom://Ubuntu 23.04 _Lunar Lobster_ - Release amd64 (20230418) lunar InRelease
Hit:2 cdrom://Ubuntu 23.04 _Lunar Lobster_ - Release amd64 (20230418) lunar Release
Get:4 http://archive.ubuntu.com/ubuntu lunar InRelease [267 kB]
Ign:5 http://security.ubuntu.com/ubuntu lunar-security InRelease
Ign:6 http://archive.ubuntu.com/ubuntu lunar-updates InRelease
Get:5 http://security.ubuntu.com/ubuntu lunar-security InRelease [109 kB]
Get:7 http://archive.ubuntu.com/ubuntu lunar-backports InRelease [99.8 kB]
```

```
Reading package lists... Done
E: Release file for http://security.ubuntu.com/ubuntu/dists/lunar-security/InRelease is not valid yet
her 13d 22h 1min 53s). Updates for this repository will not be applied.
E: Release file for http://archive.ubuntu.com/ubuntu/dists/lunar-backports/InRelease is not valid yet
her 13d 22h 2min 23s). Updates for this repository will not be applied.
lunamoon@ubuntu:~/Desktop$
```

```
lunamoon@ubuntu:~/Desktop$ sudo apt install vsftpd
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
vsftpd is already the newest version (3.0.5-0ubuntu1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

```
lunamoon@ubuntu:~/Desktop$ sudo nano /etc/vsftpd.conf
lunamoon@ubuntu:~/Desktop$
```

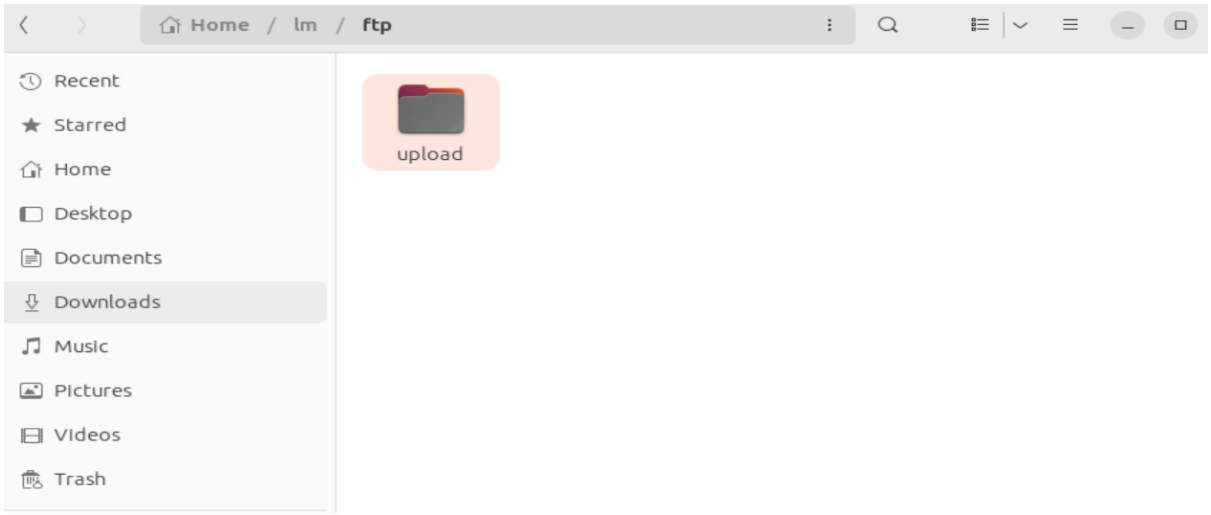
```
lunamoon@ubuntu:~/Desktop$ sudo ufw allow from any to any port 20,21,10000:10100 proto tcp
Rules updated
Rules updated (v6)
```

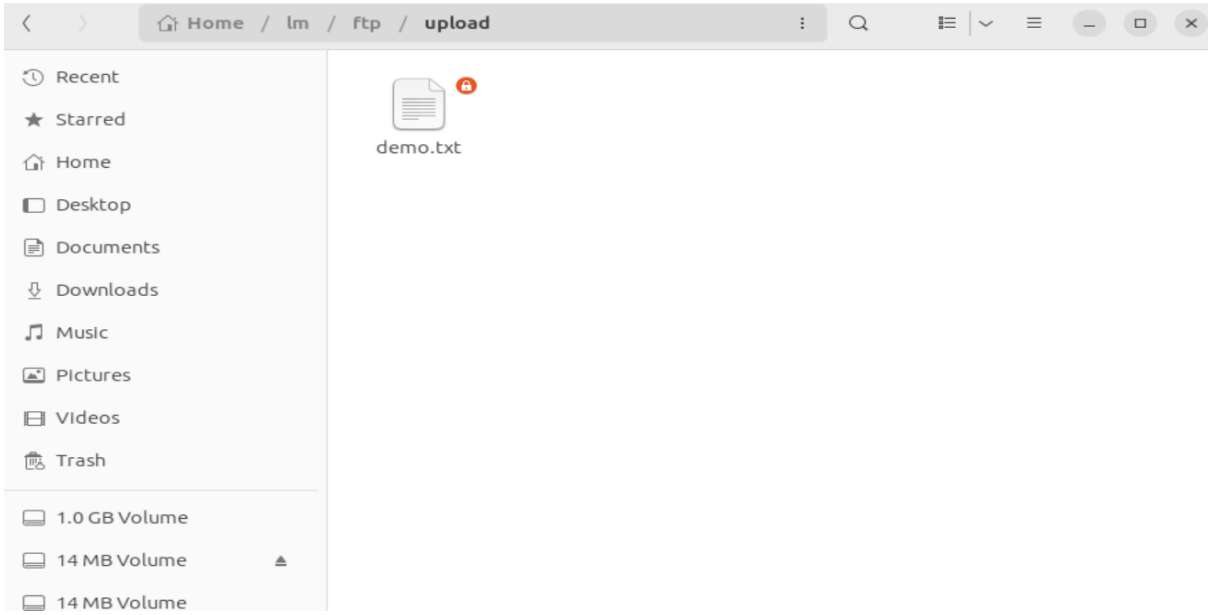
```
lunamoon@ubuntu:~$ sudo mkdir /home/lunamoon/lm/ftp
mkdir: cannot create directory ‘/home/lunamoon/lm/ftp’: File exists
```

```
lunamoon@ubuntu:~$ sudo -i
root@ubuntu:~# chown nobody:nogroup /home/lunamoon/lm/ftp
root@ubuntu:~# chmod a-w /home/lunamoon/lm/ftp
root@ubuntu:~# exit
logout
lunamoon@ubuntu:~$ sudo mkdir /home/lunamoon/lm/ftp/upload
```

```
lunamoon@ubuntu:~$ sudo -i
root@ubuntu:~# chown lunamoon:lunamoon /home/lunamoon/lm/ftp/upload
root@ubuntu:~# exit
logout
```

```
lunamoon@ubuntu:~$ echo "My FTP Server" | sudo tee /home/lunamoon/lm/ftp/upload/demo.txt
My FTP Server
lunamoon@ubuntu:~$
```





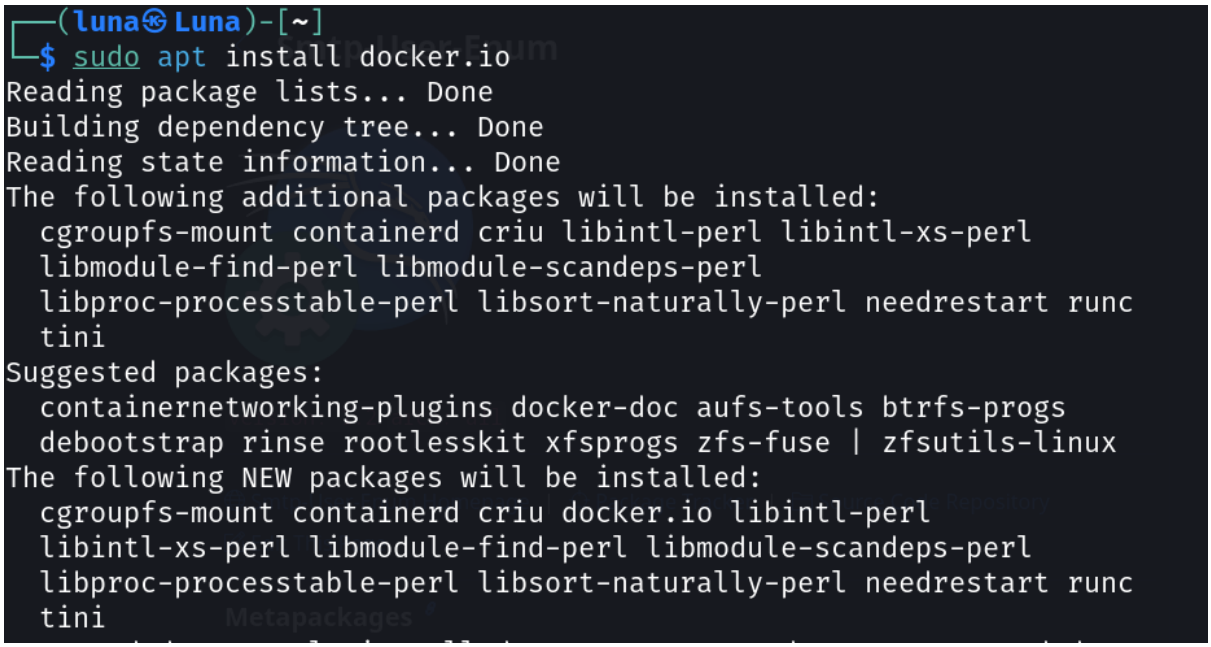

```
lunamoon@ubuntu:~$ sudo ls -la /home/lunamoon/lm/ftp
total 0
dr-xr-xr-x 3 nobody   nogroup  60 Sep 14 16:28 .
drwxrwxr-x 3 lunamoon lunamoon 60 Aug 31 17:43 ..
drwxr-xr-x 2 lunamoon lunamoon 60 Sep 14 16:32 upload
lunamoon@ubuntu:~$ echo "lunamoon" | sudo tee -a /etc/vsftpd.userlist
lunamoon
lunamoon@ubuntu:~$ sudo systemctl restart vsftpd
lunamoon@ubuntu:~$ sudo nano /etc/vsftpd.conf
lunamoon@ubuntu:~$
```

```
#
# Uncomment this to indicate that vsftpd use a utf8 filesystem.
#utf8_filesystem=YES
user_sub_token=$USER
local_root=/home.$USER/ftp
pasv_min_port=10000
pasv_max_port=10100
userlist_enable=YES
userlist_file=/etc/vsftpd.userlist
userlist_deny=NO
```

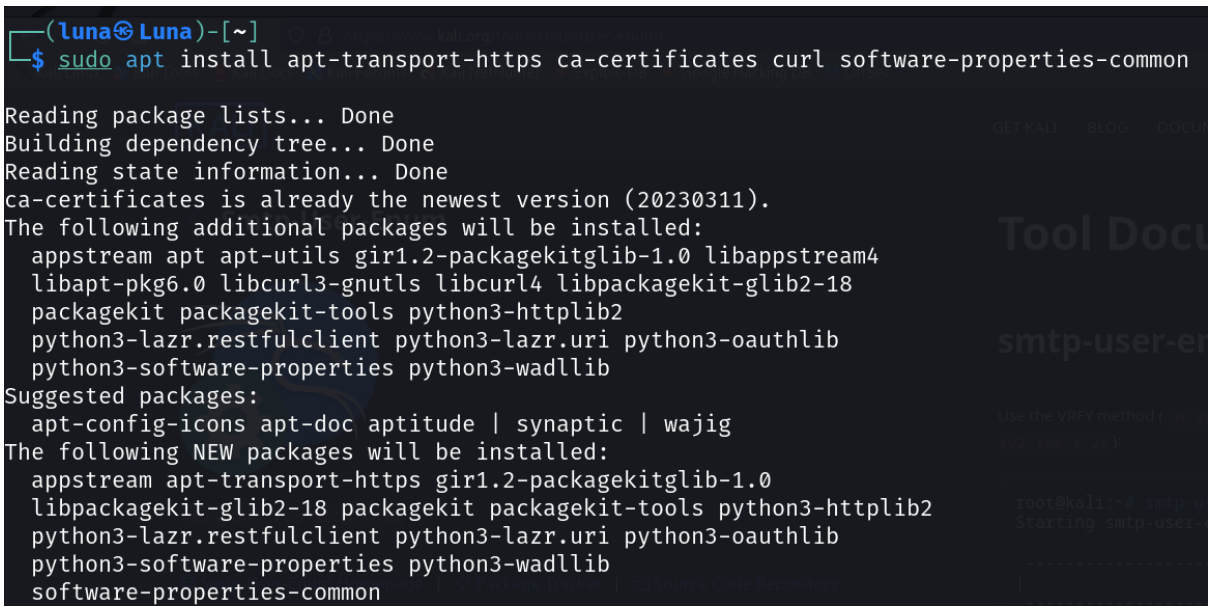
```
lunamoon@ubuntu:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::356f:5e29:ca6d:26fd  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:97:f9:2b  txqueuelen 1000  (Ethernet)
        RX packets 28703  bytes 40616555 (40.6 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 9734  bytes 667935 (667.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 19437  bytes 1487153 (1.4 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 19437  bytes 1487153 (1.4 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

LAB 4: DOCKER

AIM: To setup DOCKER in Ubuntu in Virtual Box.



```
┌──(luna㉿Luna)-[~]
└─$ sudo apt install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  cgroupfs-mount containerd criu libintl-perl libintl-xs-perl
  libmodule-find-perl libmodule-scandeps-perl
  libproc-processtable-perl libsort-naturally-perl needrestart runc
  tini
Suggested packages:
  containernetworking-plugins docker-doc aufs-tools btrfs-progs
  debootstrap rinse rootlesskit xfsprogs zfs-fuse | zfsutils-linux
The following NEW packages will be installed:
  cgroupfs-mount containerd criu docker.io libintl-perl
  libintl-xs-perl libmodule-find-perl libmodule-scandeps-perl
  libproc-processtable-perl libsort-naturally-perl needrestart runc
  tini
```



```
┌──(luna㉿Luna)-[~]
└─$ sudo apt install apt-transport-https ca-certificates curl software-properties-common

Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20230311).
The following additional packages will be installed:
  appstream apt apt-utils gir1.2-packagekitglib-1.0 libappstream4
  libapt-pkg6.0 libcurl3-gnutls libcurl4 libpackagekit-glib2-18
  packagekit packagekit-tools python3-httplib2
  python3-lazr.restfulclient python3-lazr.uri python3-oauthlib
  python3-software-properties python3-wadllib
Suggested packages:
  apt-config-icons apt-doc aptitude | synaptic | wajig
The following NEW packages will be installed:
  appstream apt-transport-https gir1.2-packagekitglib-1.0
  libpackagekit-glib2-18 packagekit packagekit-tools python3-httplib2
  python3-lazr.restfulclient python3-lazr.uri python3-oauthlib
  python3-software-properties python3-wadllib
  software-properties-common
```

```
┌──(luna㉿Luna)-[~]
└─$ curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

┌──(luna㉿Luna)-[~]
└─$ echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/debian $(lsb_release -cs) st
able" | sudo tee /etc/apt/sources.list.d/docker.list
```

```
deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/debian kali-rolling stabl

┌──(luna㉿Luna)-[~]
└─$ sudo apt update

Ign:1 https://download.docker.com/linux/debian kali-rolling InRelease
Err:3 https://download.docker.com/linux/debian kali-rolling Release
  404  Not Found [IP: 108.158.245.93 443]
Hit:2 http://kali.download/kali kali-rolling InRelease
Reading package lists... Done
E: The repository 'https://download.docker.com/linux/debian kali-rolling Release' does not have a Release file.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.

┌──(luna㉿Luna)-[~]
└─$ sudo apt install docker-ce docker-ce-cli containerd.io

Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

```
┌──(luna㉿Luna)-[~]
└─$ sudo systemctl start docker
sudo systemctl enable docker

Synchronizing state of docker.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable docker

┌──(luna㉿Luna)-[~]
└─$ sudo docker --version

Docker version 20.10.25+dfsg1, build b82b9f3
```

```
┌──(luna㉿Luna)-[~]
└─$ sudo docker run hello-world


Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

LAB 5: SHELL SCRIPT

who: To show who is logged in. This lists the users with id and the time and date of user login.

```
┌──(luna㉿Luna)-[~]
└─$ who
luna     seat0        2023-10-18 19:54 (login screen)
luna     :1           2023-10-18 19:54 (:1)
luna     pts/2        2023-10-18 20:18
luna     pts/1        2023-10-18 19:58
luna     pts/4        2023-10-18 20:19
```

2. **users:** This command will print the usernames of logged-in to the current host.

```
┌──(luna㉿Luna)-[~]
└─$ users
luna luna luna luna luna luna luna luna luna luna
```

(B) Write a shell script program to display "HELLO WORLD".

Step 1: Create an empty file named "heloo.sh" in the terminal.

```
┌──(luna㉿Luna)-[~/Desktop]
└─$ touch hello.sh

┌──(luna㉿Luna)-[~/Desktop]
└─$ ls
WALLPAPER.jpeg  hello.sh
```

Step 2: Open file.

```
┌──(luna㉿Luna)-[~/Desktop]
└─$ nano hello.sh
```

Step 3. The editor is open, write the following code in a text editor.

```
  GNU nano 7.2                    hello.sh
#!/bin/sh

# PROGRAM TO DISPLAY HELLO WORLD

echo "HELLO WORLD!!"
```

Step 4: Give permission to execute the script.

```
┌──(luna㉿Luna)-[~/Desktop]
└─$ chmod +x hello.sh
```

Step 5: Run the script.

```
┌──(luna㉿Luna)-[~/Desktop]
└─$ ./hello.sh
HELLO WORLD!!
```

(C) Write a shell Script program to check whether the given number is even or odd.

```
┌──(luna㉿Luna)-[~/Desktop]
└─$ touch odd_even.sh

┌──(luna㉿Luna)-[~/Desktop]
└─$ nano odd_even.sh
```

```
  GNU nano 7.2                    odd_even.sh
#!/bin/sh
echo "Enter a number: "
read n
rem=$(( $n % 2 ))
if [ $rem -eq 0 ]
then
echo "$n IS EVEN."
else
echo "$n IS ODD."
fi
```

```
┌──(luna㉿Luna)-[~/Desktop]
└─$ chmod +x odd_even.sh

┌──(luna㉿Luna)-[~/Desktop]
└─$ ./odd_even.sh
Enter a number:
105
105 IS ODD.
```

(D) Write a shell script program to develop a scientific calculator.

THEORY:

```
GNU nano 7.2
#!/bin/sh
echo "SCIENTIFIC CALCULATOR"
sum=0
i="y"
while [ $i = "y" ]
do
echo "1) SQUARE ROOT"
echo "2) LOGARITHMETIC FUNCTION (e)"
echo "3) Sine"
echo "4) Cosine"
echo "ENTER YOUR CHOICE: "
read ch

case $ch in
1)echo "Enter the number: "
read e
n= echo - | awk '{print sqrt('$e')}'
echo $n;;

2)echo "Enter the number: "
read e
n= echo - | awk '{print log('$e')}'
echo $n;;

3)echo "Enter the degree: : "
read e
```

```
s= echo - | awk '{print (('$e' * 3.14159) / 180 )}'
n= echo - | awk '{print sin('$s')}'
echo $n;;

4)echo "Enter the degree: : "
read e
s= echo - | awk '{print (('$e' * 3.14159) / 180 )}'
n= echo - | awk '{print cos('$s')}'
echo $n;;

esac
echo "DO YOU WAN TO CONTINUE? [Y/N]"
read i
if [ $i != "y"]
then
exit
fi
done
```

```
┌──(luna㉿Luna)-[~/Desktop]
└─$ touch calculator.sh

┌──(luna㉿Luna)-[~/Desktop]
└─$ nano calculator.sh
```

```
┌──(luna㉿Luna)-[~/Desktop]
└─$ nano calculator.sh

┌──(luna㉿Luna)-[~/Desktop]
└─$ bash calculator.sh
SCIENTIFIC CALCULATOR
1) SQUARE ROOT
2) LOGARITHMETIC FUNCTION (e)
3) Sine
4) Cosine
ENTER YOUR CHOICE:
1
Enter the number:
9
3

DO YOU WAN TO CONTINUE? [Y/N]
n
calculator.sh: line 40: [: missing `]'

┌──(luna㉿Luna)-[~/Desktop]
└─$ nano calculator.sh
```

LAB 6: NFS SERVER

Step 1: Install NFS Server

```
┌──(luna㉿Luna)-[~]
└─$ sudo apt-get install nfs-kernel-server
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  nfs-kernel-server
0 upgraded, 1 newly installed, 0 to remove and 951 not upgraded.
Need to get 162 kB of archives.
After this operation, 693 kB of additional disk space will be used.
Get:1 http://kali.download/kali kali-rolling/main amd64 nfs-kernel-server amd64 1:2.6.3-3 [162 kB]
Fetched 162 kB in 4s (38.5 kB/s)
Selecting previously unselected package nfs-kernel-server.
(Reading database ... 437179 files and directories currently installed.)
Preparing to unpack .../nfs-kernel-server_1%3a2.6.3-3_amd64.deb ...
```

Step-2: Create a Directory to Share

```
┌──(luna㉿Luna)-[~]
└─$ sudo mkdir /nfs_share
```

Step-3: Configure NFS Exports

```
┌──(luna㉿Luna)-[~]
└─$ sudo nano /etc/exports
```

Step-4: Restart NFS Server

```
┌──(luna㉿Luna)-[~]
└─$ sudo systemctl restart nfs-kernel-server
```

Step-5: Open Port 2049 in Firewall

```
┌──(luna㉿Luna)-[~]
└─$ sudo ufw allow 2049/tcp
Rules updated
Rules updated (v6)
```

Step-6: Test NFS Share

```
┌──(luna㉿Luna)-[~]
└─$ sudo mkdir /mnt/nfs_mount

┌──(luna㉿Luna)-[~]
└─$ sudo mount -t nfs server ip:/nfs_share /mnt/nfs_mount
```

Step-7: Unmount the NFS Share

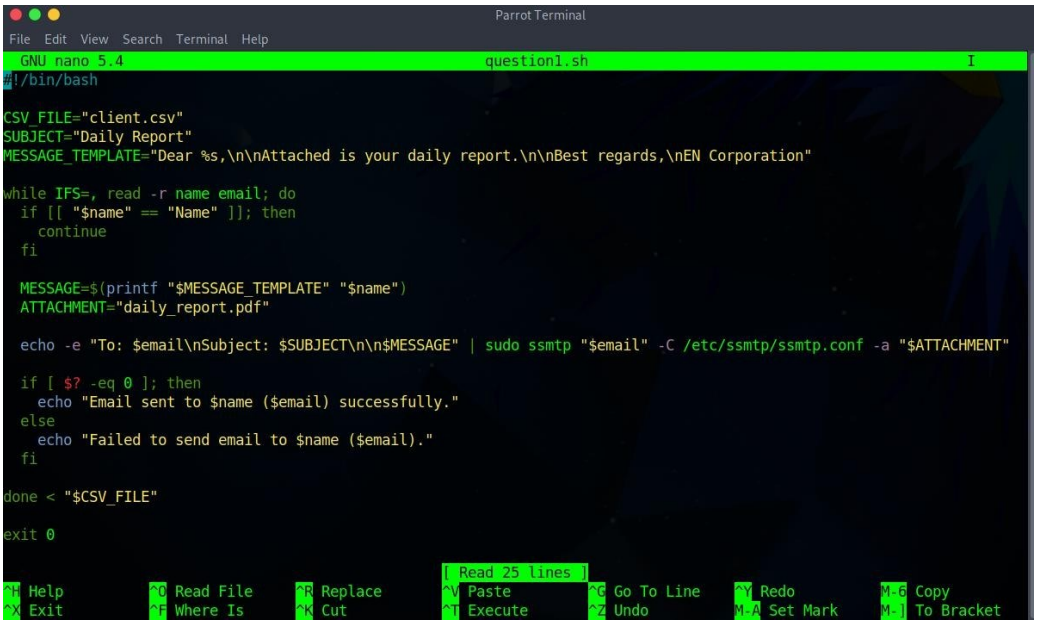
```
┌──(luna㉿Luna)-[~]
└─$ sudo umount /mnt/nfs_mount
```

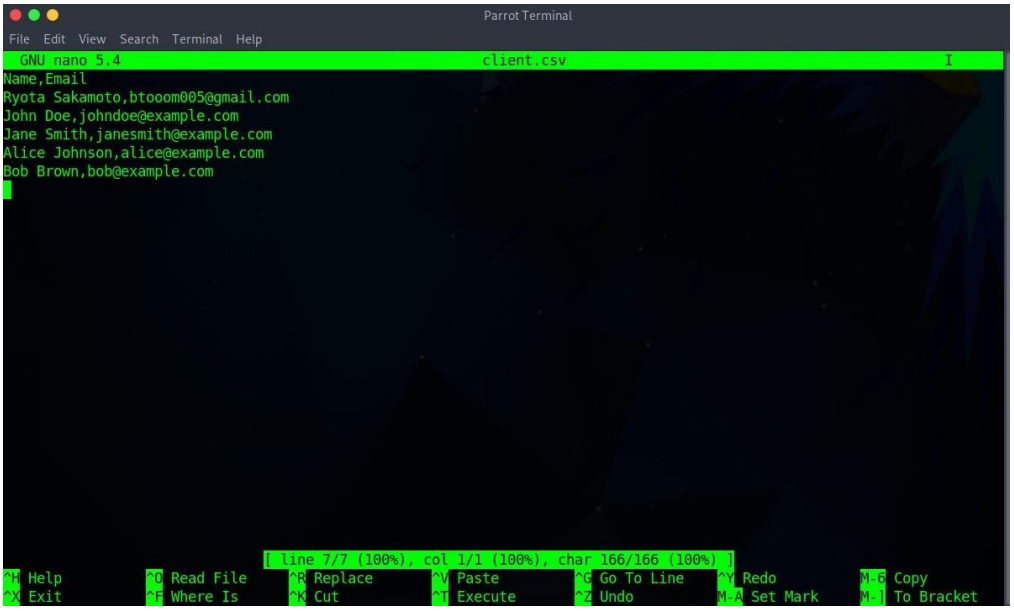
LAB 7: SHELL SCRIPTING (SMTP)

1. **Scenario:** Imagine a small business wants to automate the process of sending daily reportsto their clients via email. They have data in CSV files and want to send personalized emails with attachments.

Question: Write a shell script that reads data from a CSV file containing client informationand sends personalized emails to each client with a daily report attached. Use control instructions to handle the process efficiently.

Answer:



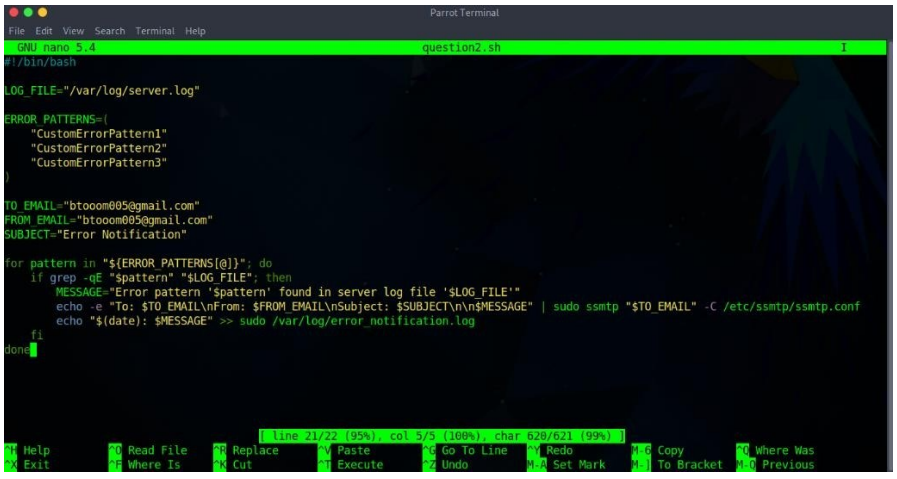


```
┌[nitesh@parrot]-[~/Desktop]
└─ $sudo nano client.csv
┌[nitesh@parrot]-[~/Desktop]
└─ $sudo nano question1.sh
┌[nitesh@parrot]-[~/Desktop]
└─ $sudo chmod +x question1.sh
┌[nitesh@parrot]-[~/Desktop]
└─ $./question1.sh
Email sent to Ryota Sakamoto (btooom005@gmail.com) successfully.
Email sent to John Doe (johndoe@example.com) successfully.
Email sent to Jane Smith (janesmith@example.com) successfully.
Email sent to Alice Johnson (alice@example.com) successfully.
Email sent to Bob Brown (bob@example.com) successfully.
```

2. Scenario: A system administrator needs to monitor server logs for specific error messagesand send alerts via email when critical errors occur.

Question: Develop a shell script that regularly scans server log files, searches for predefinederror patterns using regular expressions, and sends an email notification if any errors are found.

Answer:



```
┌─[nitesh@parrot]─[~/Desktop]
└──╼ $sudo nano question2.sh
┌─[nitesh@parrot]─[~/Desktop]
└──╼ $sudo chmod +x question2.sh
┌─[nitesh@parrot]─[~/Desktop]
└──╼ $./question2.sh
```

Error Notification Inbox ×

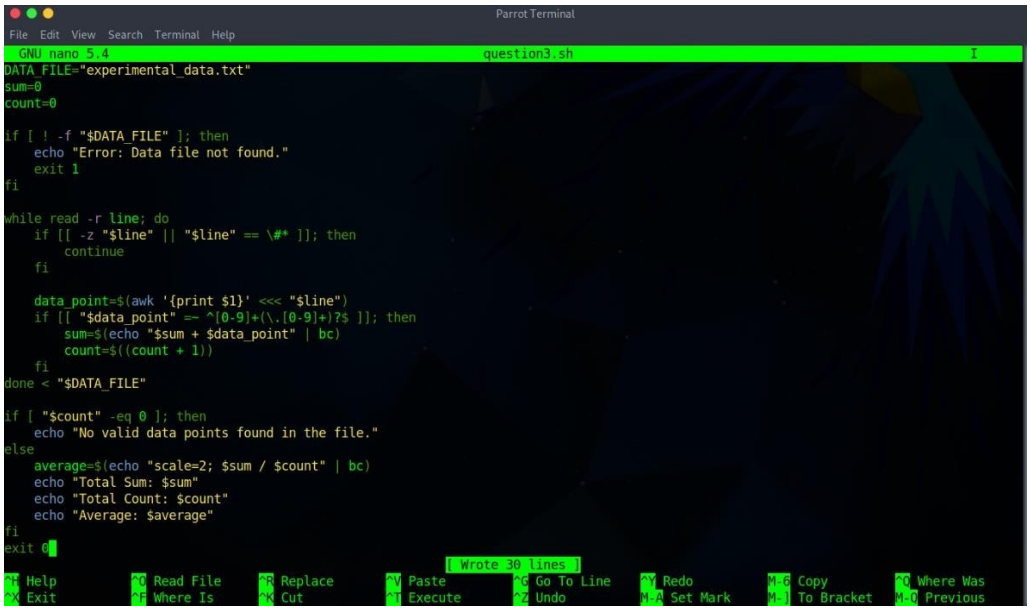
R **btooom005@gmail.com** 9:36 PM (0 minutes ago)
to me

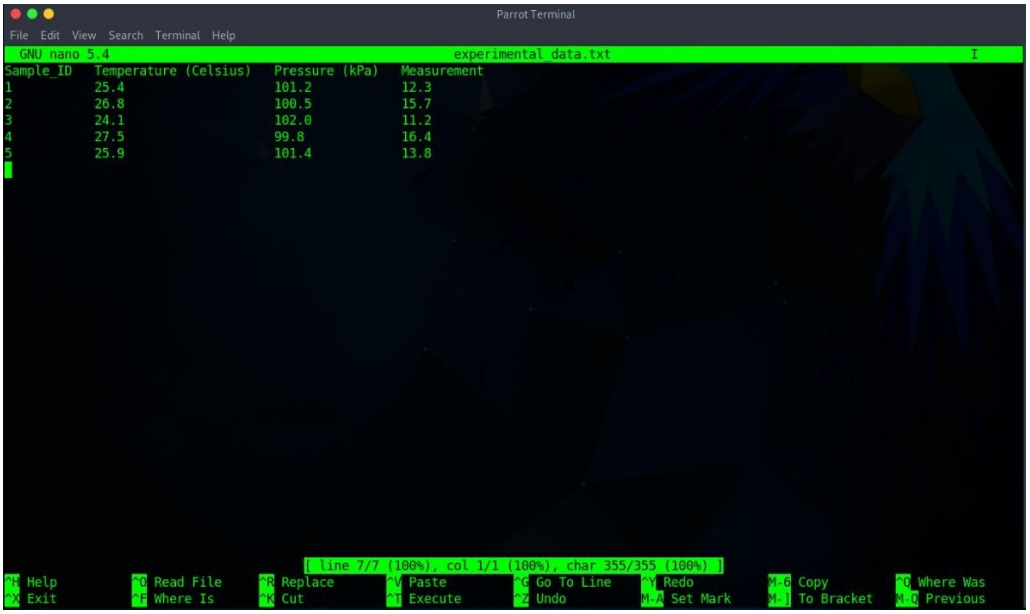
Error pattern 'CustomErrorPattern1' found in server log file '/var/log/server.log'

3. Scenario: A system administrator needs to monitor server logs for specific error messagesand send alerts via email when critical errors occur.

Question: Develop a shell script that regularly scans server log files, searches for predefinederror patterns using regular expressions, and sends an email notification if any errors are found.

Answer:



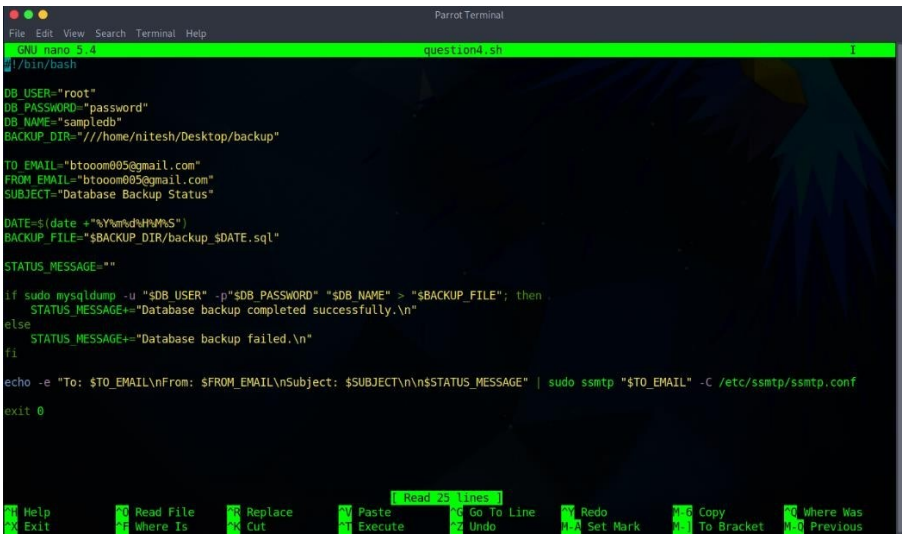


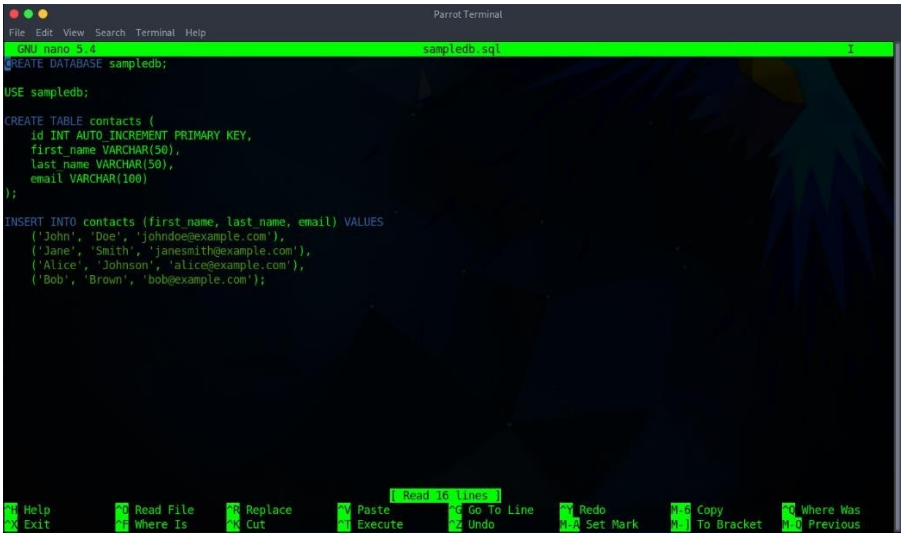

```
┌─[nitesh@parrot]─[~/Desktop]
└──╼ $sudo nano question3.sh
┌─[nitesh@parrot]─[~/Desktop]
└──╼ $sudo nano experimental_data.txt
┌─[nitesh@parrot]─[~/Desktop]
└──╼ $sudo chmod +x question3.sh
┌─[nitesh@parrot]─[~/Desktop]
└──╼ $./question3.sh
Total Sum: 15
Total Count: 5
Average: 3.00
```

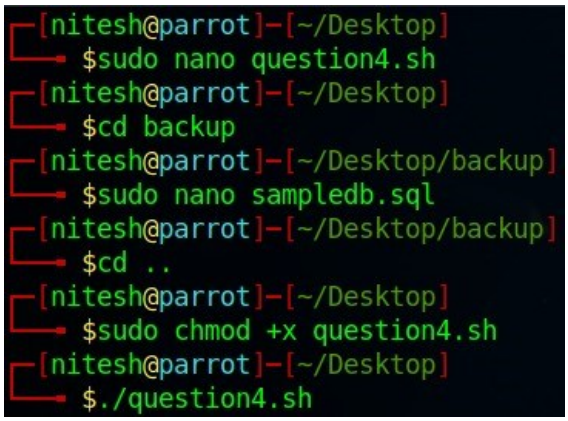
1. Scenario: A database administrator needs to automate database backups. They want a script that creates backups, stores them locally, and sends email notifications with backupstatus.

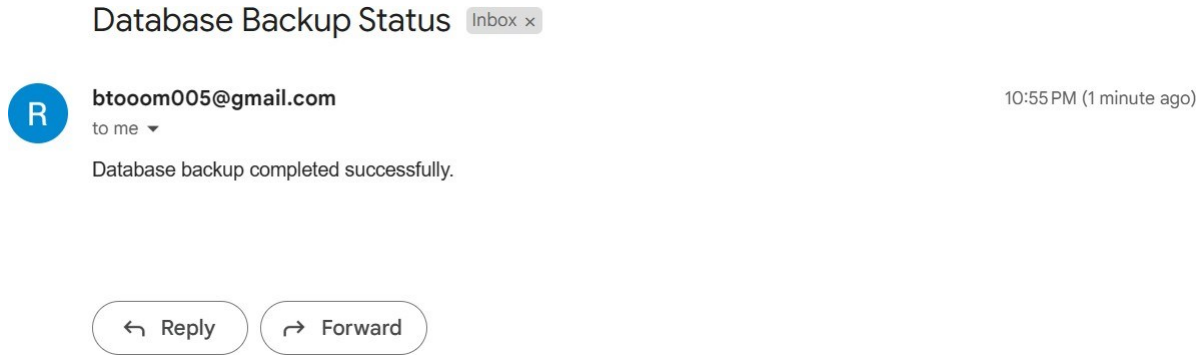
   Question: Write a shell script that performs database backups, stores them, and sends emailnotifications with backup status information. Implement control structures for backup and email handling.

Answer:


Parrot Terminal
File Edit View Search Terminal Help
GNU nano 5.4 question4.sh
#!/bin/bash
DB_USER="root"
DB_PASSWORD="password"
DB_NAME="sampledb"
BACKUP_DIR="///home/nitesh/Desktop/backup"
TO_EMAIL="btooom005@gmail.com"
FROM_EMAIL="btooom005@gmail.com"
SUBJECT="Database Backup Status"
DATE=$(date +"%Y%m%d%H%M%S")
BACKUP_FILE="$BACKUP_DIR/backup_$DATE.sql"
STATUS_MESSAGE=""
if sudo mysqldump -u "$DB_USER" -p"$DB_PASSWORD" "$DB_NAME" > "$BACKUP_FILE"; then
STATUS_MESSAGE+="Database backup completed successfully.\n"
else
STATUS_MESSAGE+="Database backup failed.\n"
fi
echo -e "To: $TO_EMAIL\nFrom: $FROM_EMAIL\nSubject: $SUBJECT\n\n$STATUS_MESSAGE" | sudo ssmtp "$TO_EMAIL" -C /etc/ssmtp/ssmtp.conf
exit 0
[Read 25 lines]
^G Help ^O Read File ^R Replace ^V Paste ^G Go To Line ^Y Redo M-6 Copy ^Q Where Was
^X Exit ^F Where Is ^K Cut ^T Execute ^Z Undo M-A Set Mark M-] To Bracket M-O Previous


Parrot Terminal
File Edit View Search Terminal Help
GNU nano 5.4 sampledb.sql
CREATE DATABASE sampledb;
USE sampledb;
CREATE TABLE contacts (
id INT AUTO_INCREMENT PRIMARY KEY,
first_name VARCHAR(50),
last_name VARCHAR(50),
email VARCHAR(100)
);
INSERT INTO contacts (first_name, last_name, email) VALUES
('John', 'Doe', 'johndoe@example.com'),
('Jane', 'Smith', 'janesmith@example.com'),
('Alice', 'Johnson', 'alice@example.com'),
('Bob', 'Brown', 'bob@example.com');
[Read 16 lines]
^G Help ^O Read File ^R Replace ^V Paste ^G Go To Line ^Y Redo M-6 Copy ^Q Where Was
^X Exit ^F Where Is ^K Cut ^T Execute ^Z Undo M-A Set Mark M-] To Bracket M-O Previous

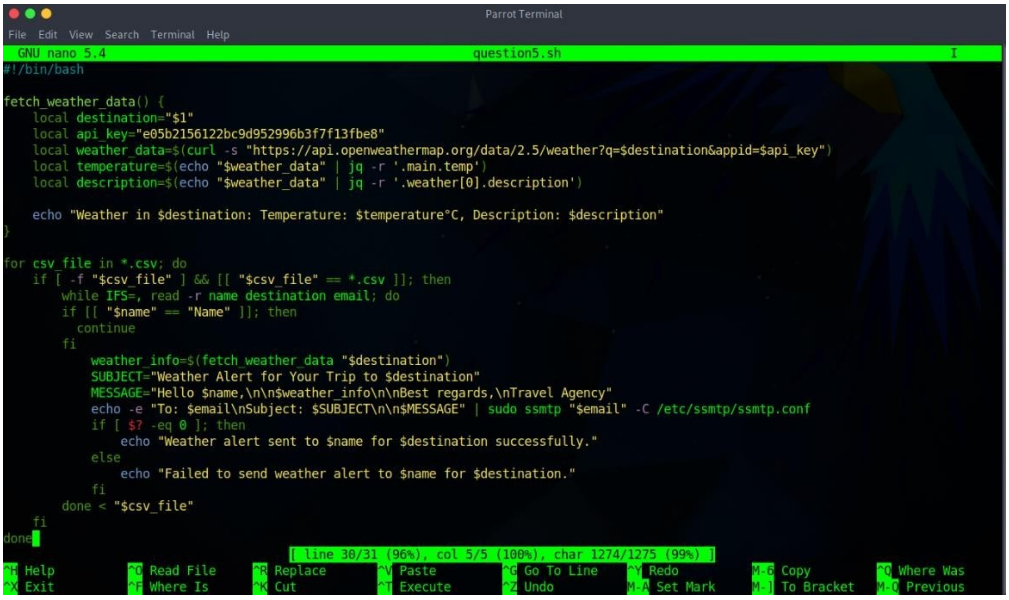

[nitesh@parrot]-[~/Desktop]
$sudo nano question4.sh
[nitesh@parrot]-[~/Desktop]
$cd backup
[nitesh@parrot]-[~/Desktop/backup]
$sudo nano sampledb.sql
[nitesh@parrot]-[~/Desktop/backup]
$cd ..
[nitesh@parrot]-[~/Desktop]
$sudo chmod +x question4.sh
[nitesh@parrot]-[~/Desktop]
$./question4.sh

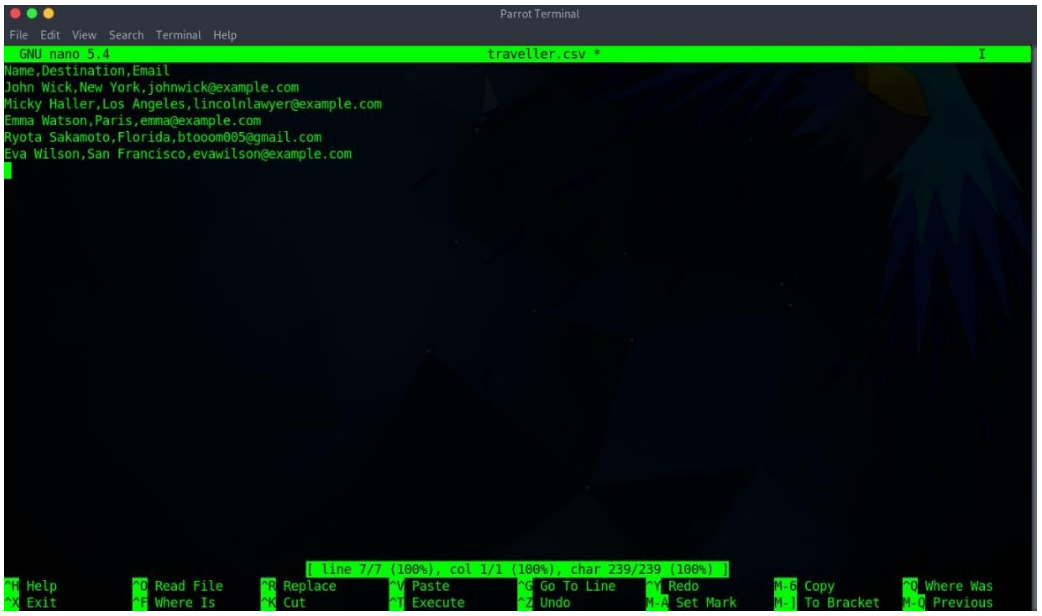


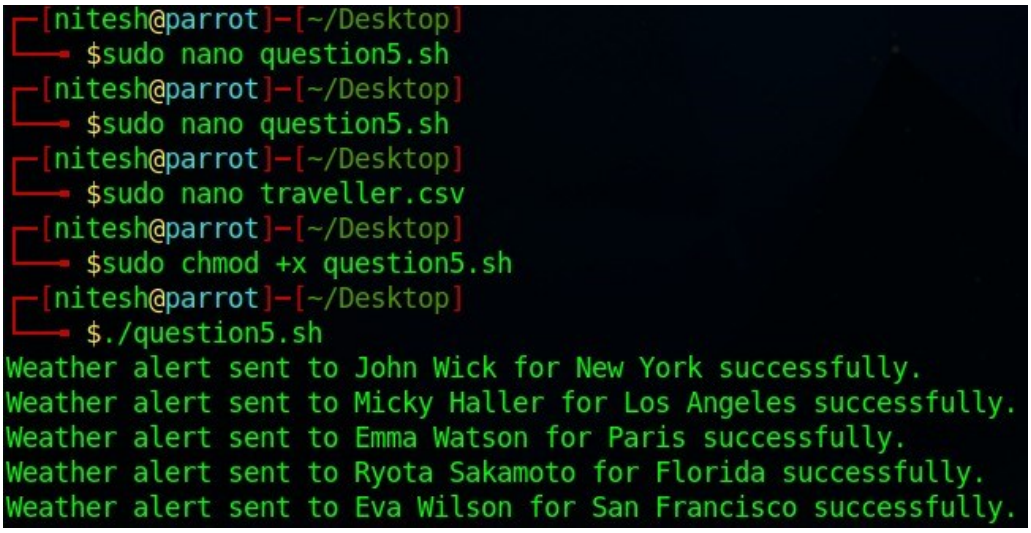
2. Scenario: A travel agency wants to provide weather alerts to customers traveling to different destinations. They have CSV files with traveler itineraries and need to send weatheralerts for each location.

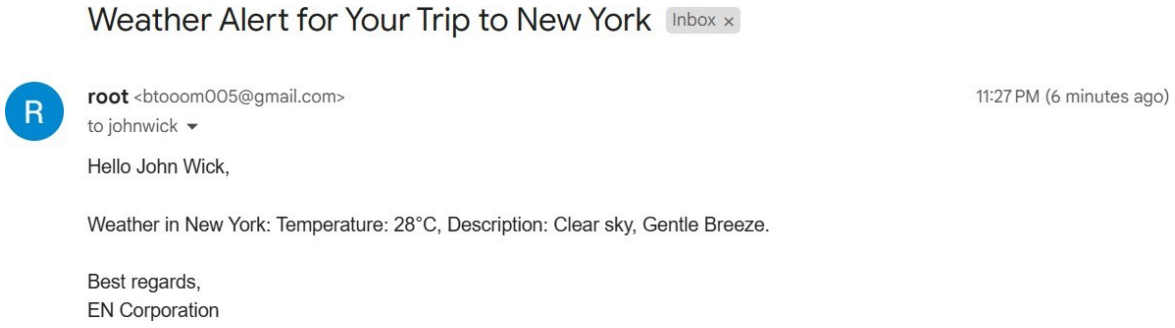
Question: Create a shell script that reads traveler itineraries from CSV files, fetches weatherforecasts for each destination, and sends weather alerts to travelers. Use control structures for data processing and email alerts.

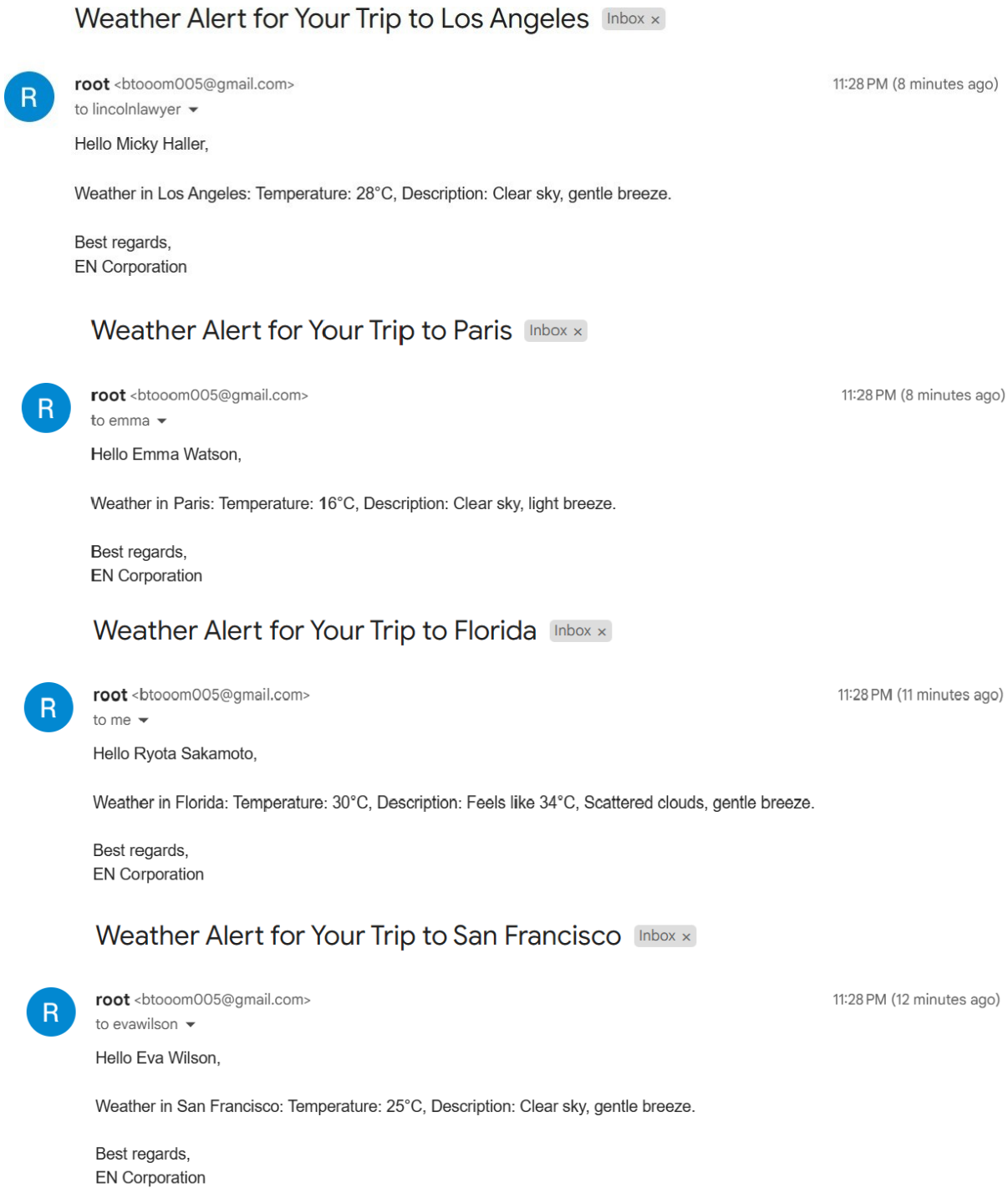
Answer:




Parrot Terminal
File Edit View Search Terminal Help
GNU nano 5.4 traveller.csv *
Name,Destination,Email
John Wick,New York,johnwick@example.com
Micky Haller,Los Angeles,lincolnlawyer@example.com
Emma Watson,Paris,emma@example.com
Ryota Sakamoto,Florida,btooom005@gmail.com
Eva Wilson,San Francisco,evawilson@example.com
[line 7/7 (100%), col 1/1 (100%), char 239/239 (100%)]
^G Help ^O Read File ^R Replace ^V Paste ^G Go To Line ^Y Redo M-6 Copy ^Q Where Was
^X Exit ^F Where Is ^K Cut ^T Execute ^Z Undo M-A Set Mark M-] To Bracket M-Q Previous

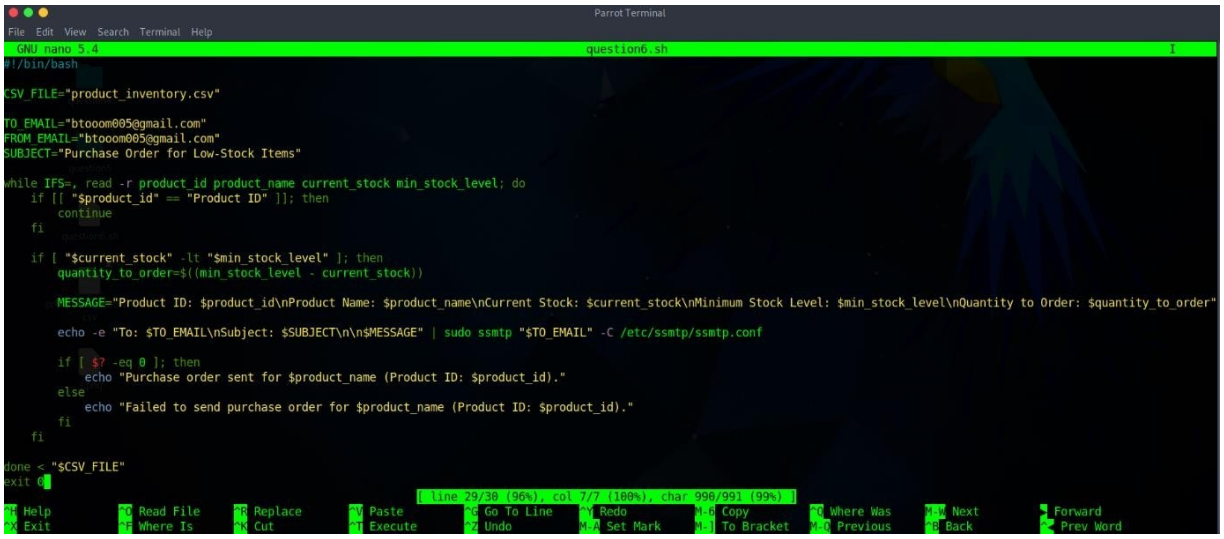

┌[nitesh@parrot]-[~/Desktop]
└─$sudo nano question5.sh
┌[nitesh@parrot]-[~/Desktop]
└─$sudo nano question5.sh
┌[nitesh@parrot]-[~/Desktop]
└─$sudo nano traveller.csv
┌[nitesh@parrot]-[~/Desktop]
└─$sudo chmod +x question5.sh
┌[nitesh@parrot]-[~/Desktop]
└─$./question5.sh
Weather alert sent to John Wick for New York successfully.
Weather alert sent to Micky Haller for Los Angeles successfully.
Weather alert sent to Emma Watson for Paris successfully.
Weather alert sent to Ryota Sakamoto for Florida successfully.
Weather alert sent to Eva Wilson for San Francisco successfully.


Weather Alert for Your Trip to New York
Inbox ×
R
root <btooom005@gmail.com>
11:27 PM (6 minutes ago)
to johnwick
Hello John Wick,
Weather in New York: Temperature: 28°C, Description: Clear sky, Gentle Breeze.
Best regards,
EN Corporation



3. Scenario: A store manager wants to automate inventory replenishment orders. They have aCSV file with product data and minimum stock levels. They want to send purchase orders viaemail when stock is low.

Question: Write a shell script that reads the product CSV file, checks stock levels, and sendspurchase orders via email for low-stock items. Use control structures for inventory management and email notifications.



Purchase Order for Low-Stock Items Inbox ×

R **root** <btooom005@gmail.com>
to me

Product ID: 101
Product Name: Widget A
Current Stock: 50
Minimum Stock Level: 100
Quantity to Order: 50

Purchase Order for Low-Stock Items Inbox ×

R **root** <btooom005@gmail.com>
to me

Product ID: 102
Product Name: Widget B
Current Stock: 75
Minimum Stock Level: 150
Quantity to Order: 75

Purchase Order for Low-Stock Items Inbox ×

R **root** <btooom005@gmail.com>
to me

Product ID: 103
Product Name: Widget C
Current Stock: 40
Minimum Stock Level: 80
Quantity to Order: 40

Purchase Order for Low-Stock Items Inbox ×

root <btooom005@gmail.com>
to me

Product ID: 104
Product Name: Widget D
Current Stock: 110
Minimum Stock Level: 200

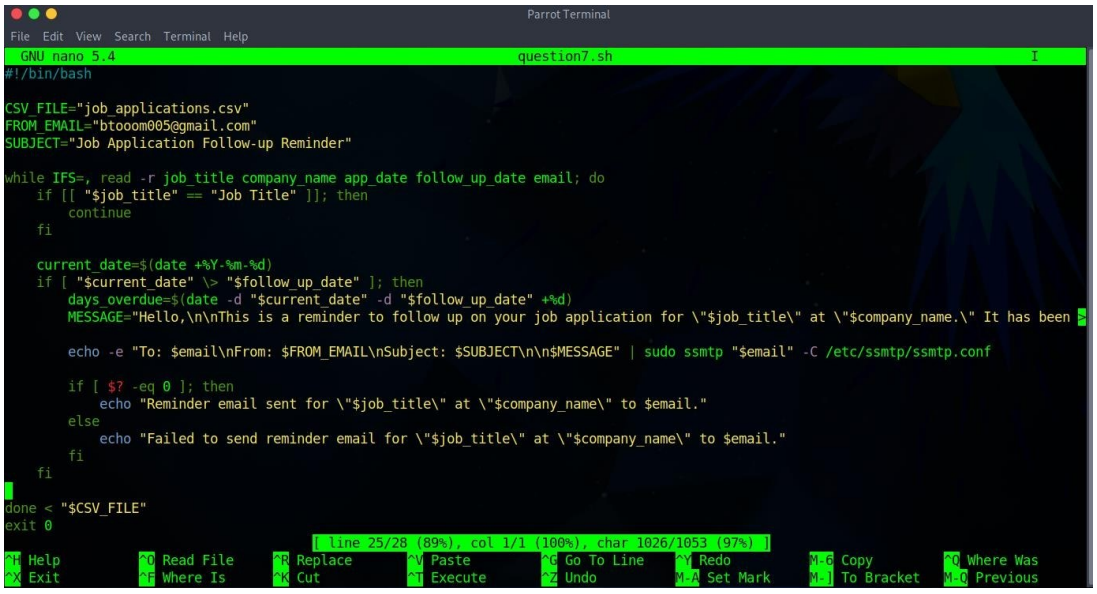
Purchase Order for Low-Stock Items Inbox ×

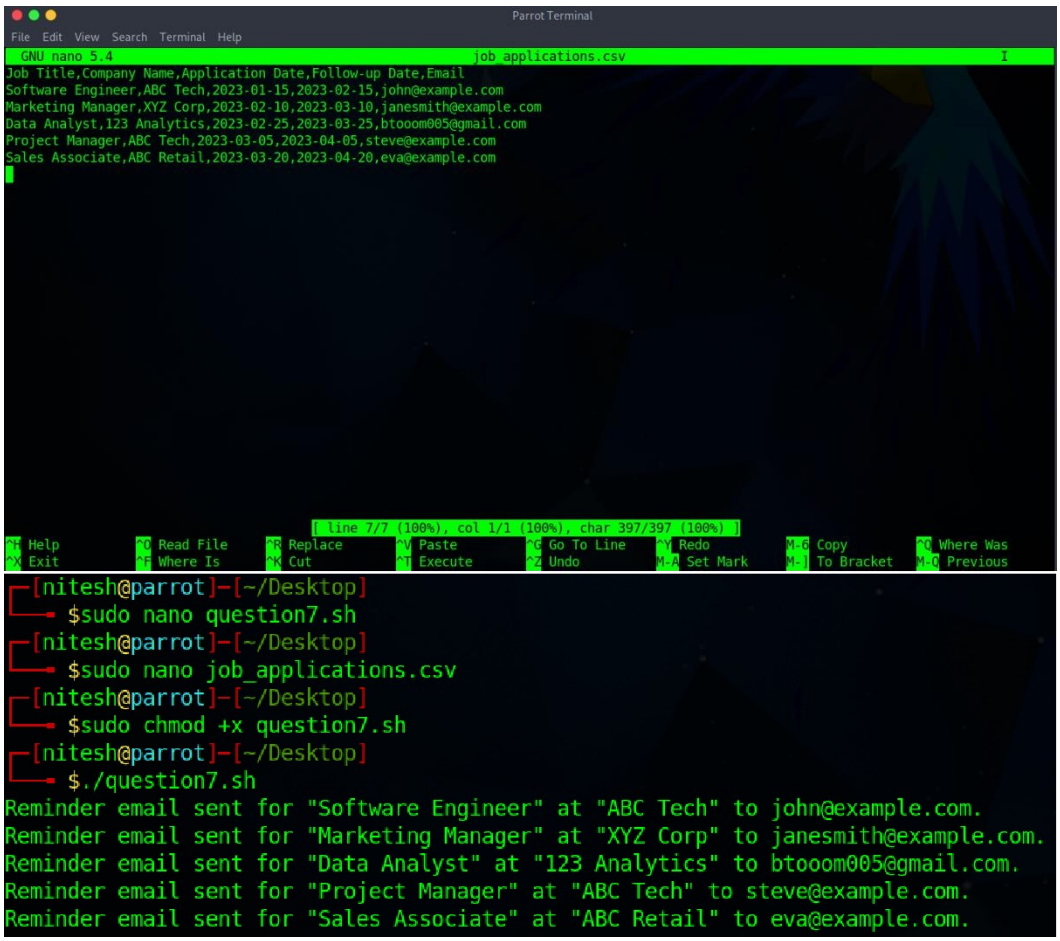
root <btooom005@gmail.com>
to me

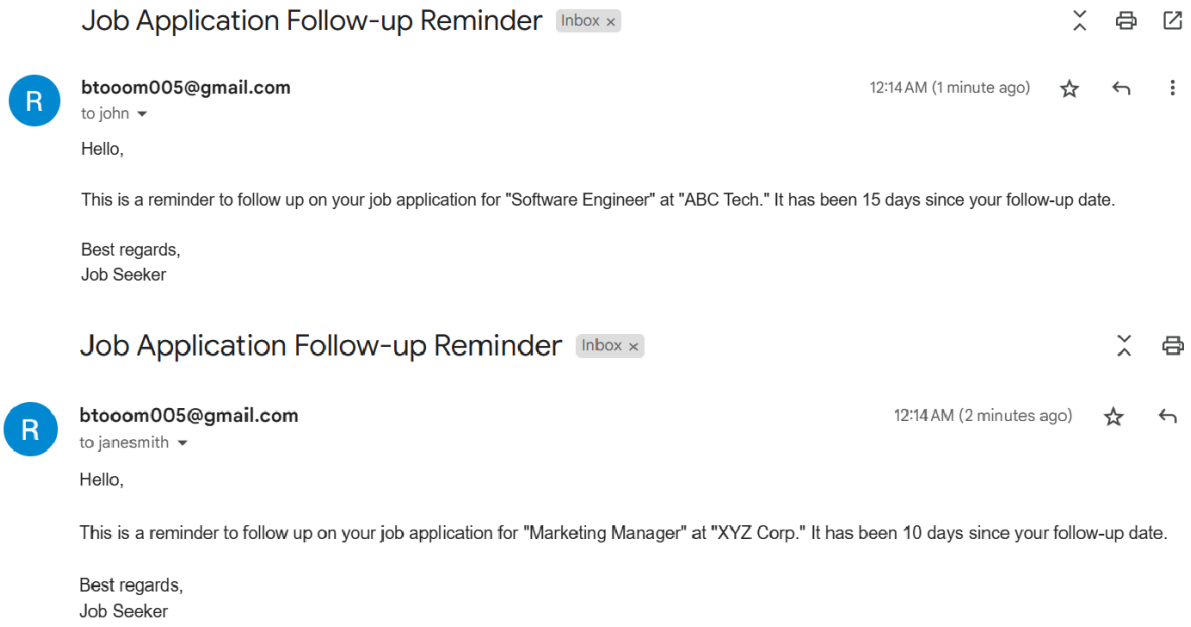
Product ID: 105
Product Name: Widget E
Current Stock: 90
Minimum Stock Level: 150
Quantity to Order: 60

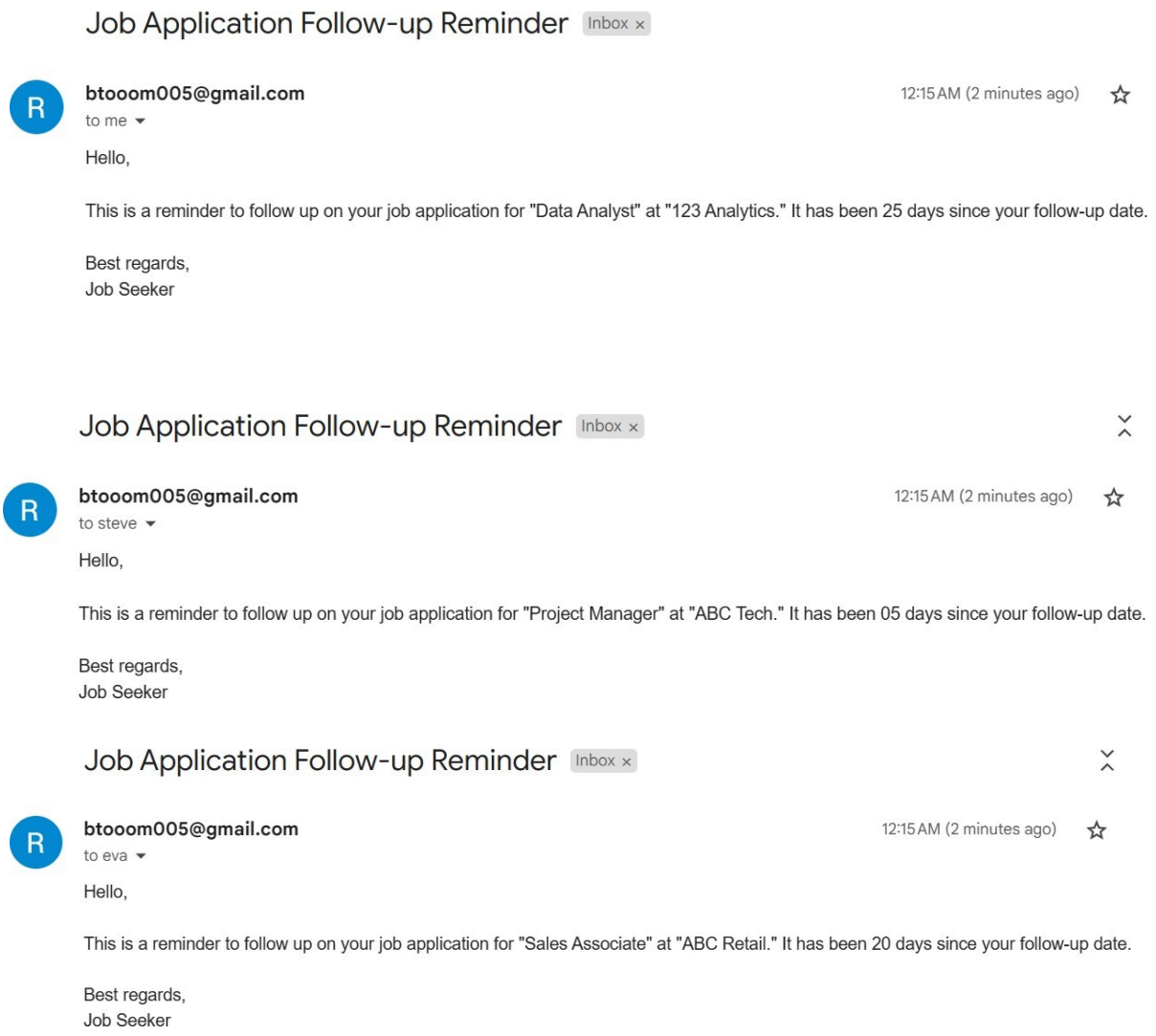
1. Scenario: A job seeker is applying to multiple positions and wants to track application details. They have a CSV file with job application data and want to send follow-up emails after a certain period.

   Question: Develop a shell script that manages job applications, tracks follow-up dates, andsends reminder emails. Implement control structures for application tracking and email scheduling.




Parrot Terminal
File Edit View Search Terminal Help
GNU nano 5.4 job applications.csv
Job Title,Company Name,Application Date,Follow-up Date,Email
Software Engineer,ABC Tech,2023-01-15,2023-02-15,john@example.com
Marketing Manager,XYZ Corp,2023-02-10,2023-03-10,janesmith@example.com
Data Analyst,123 Analytics,2023-02-25,2023-03-25,btooom005@gmail.com
Project Manager,ABC Tech,2023-03-05,2023-04-05,steve@example.com
Sales Associate,ABC Retail,2023-03-20,2023-04-20,eva@example.com
[line 7/7 (100%), col 1/1 (100%), char 397/397 (100%)]
^G Help ^O Read File ^R Replace ^V Paste ^G Go To Line ^Y Redo M-G Copy ^Q Where Was
^X Exit ^F Where Is ^K Cut ^T Execute ^Z Undo M-A Set Mark M-] To Bracket M-Q Previous
[nitesh@parrot]-[~/Desktop]
$sudo nano question7.sh
[nitesh@parrot]-[~/Desktop]
$sudo nano job_applications.csv
[nitesh@parrot]-[~/Desktop]
$sudo chmod +x question7.sh
[nitesh@parrot]-[~/Desktop]
$./question7.sh
Reminder email sent for "Software Engineer" at "ABC Tech" to john@example.com.
Reminder email sent for "Marketing Manager" at "XYZ Corp" to janesmith@example.com.
Reminder email sent for "Data Analyst" at "123 Analytics" to btooom005@gmail.com.
Reminder email sent for "Project Manager" at "ABC Tech" to steve@example.com.
Reminder email sent for "Sales Associate" at "ABC Retail" to eva@example.com.

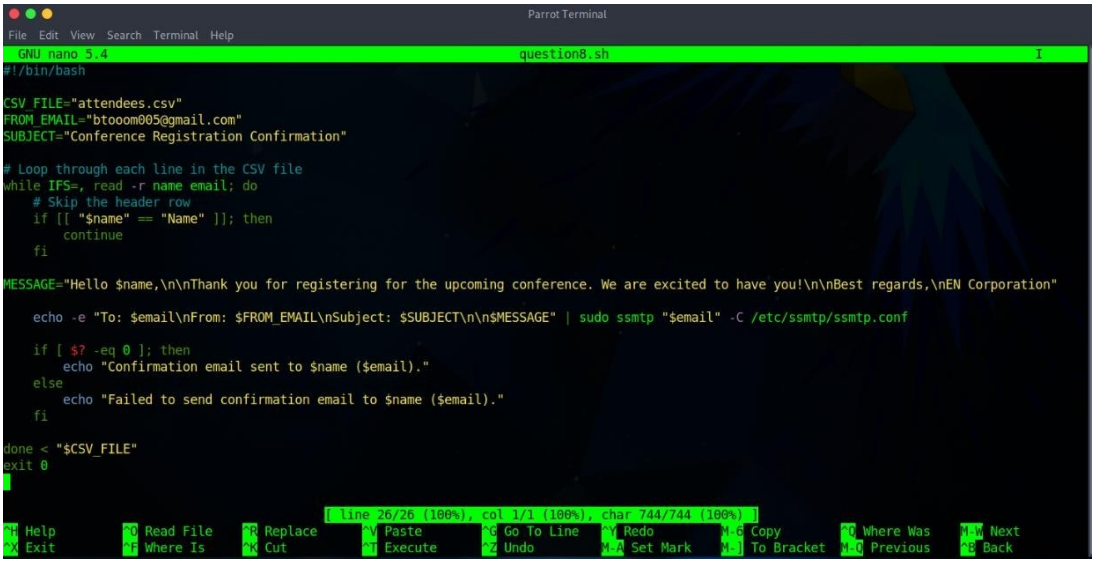

Job Application Follow-up Reminder
Inbox ×
R
btooom005@gmail.com
to john
12:14 AM (1 minute ago)
Hello,
This is a reminder to follow up on your job application for "Software Engineer" at "ABC Tech." It has been 15 days since your follow-up date.
Best regards,
Job Seeker
Job Application Follow-up Reminder
Inbox ×
R
btooom005@gmail.com
to janesmith
12:14 AM (2 minutes ago)
Hello,
This is a reminder to follow up on your job application for "Marketing Manager" at "XYZ Corp." It has been 10 days since your follow-up date.
Best regards,
Job Seeker

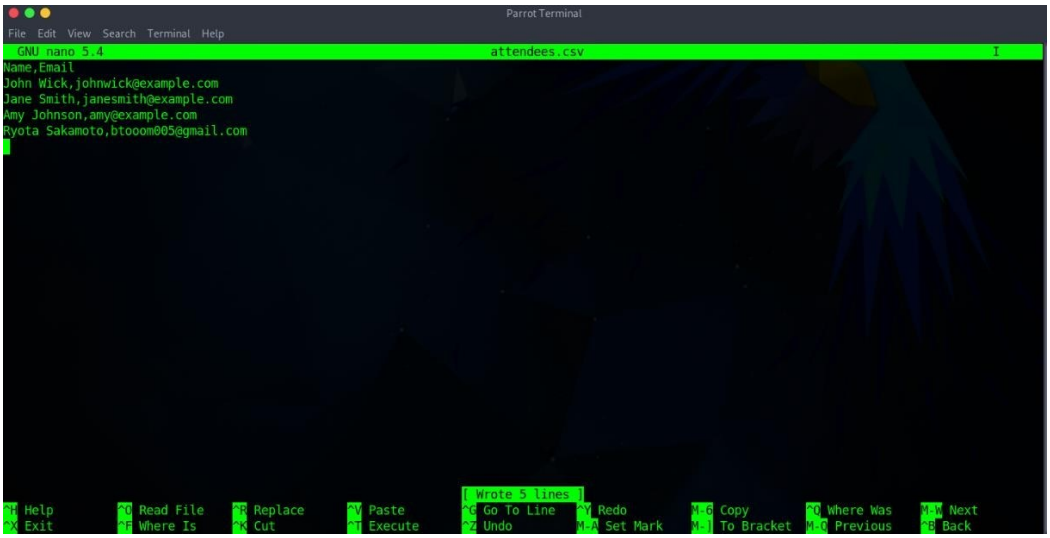


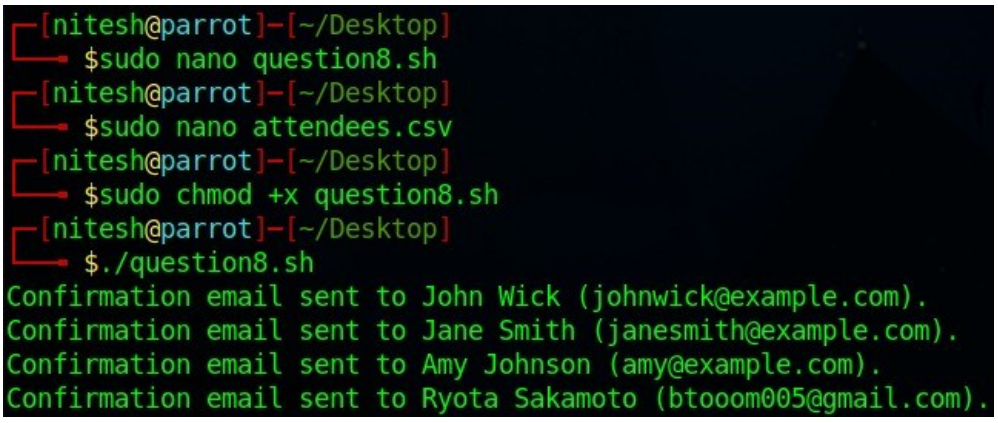
1. Scenario: An event organizer needs to automate registration for an upcoming conference.They have a CSV file with attendee information and want to send confirmation emails to registered participants.

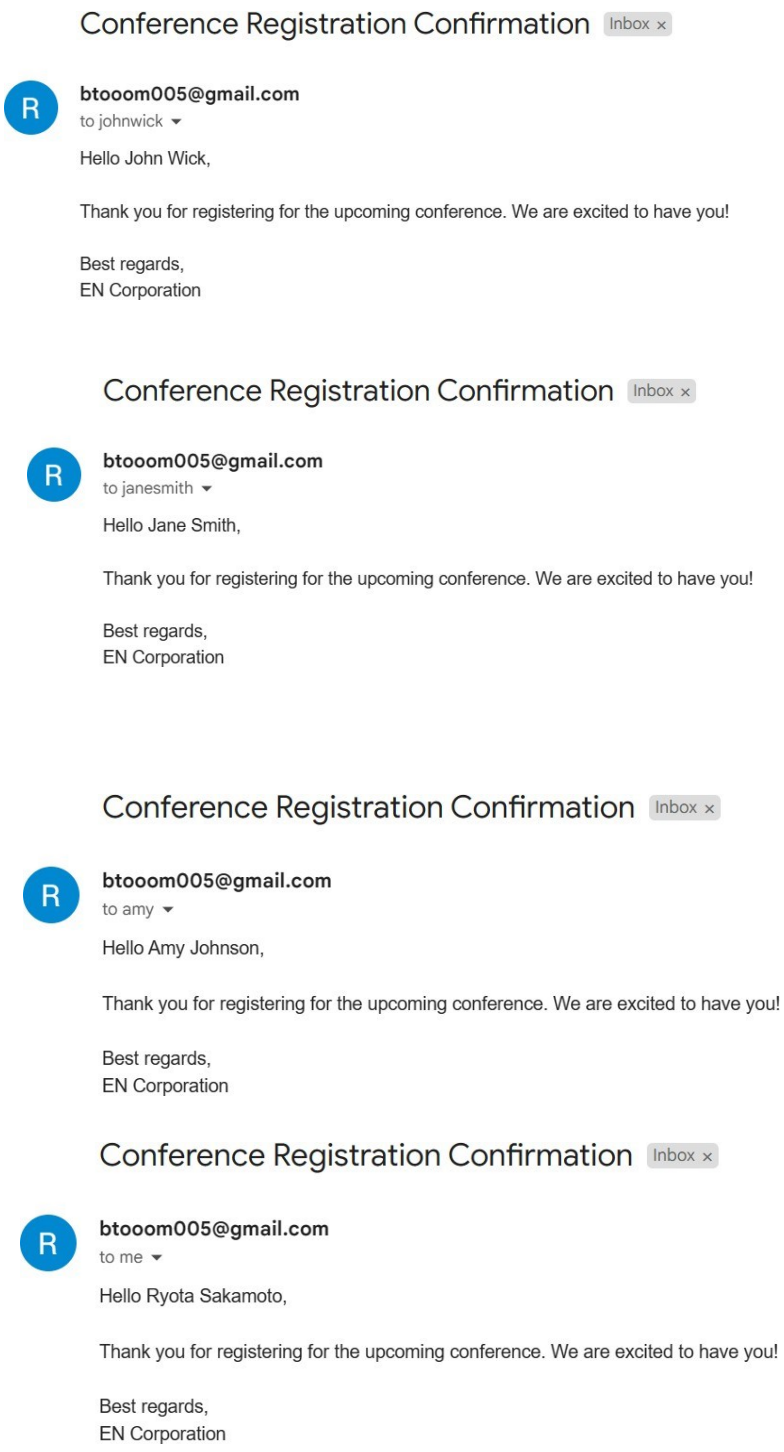
   Question: Create a shell script that processes the attendee CSV file, registers participants,and sends confirmation emails. Use control structures for registration and email sending.

Answer:


Parrot Terminal
File Edit View Search Terminal Help
GNU nano 5.4 question8.sh
#!/bin/bash
CSV_FILE="attendees.csv"
FROM_EMAIL="btooom005@gmail.com"
SUBJECT="Conference Registration Confirmation"
# Loop through each line in the CSV file
while IFS=, read -r name email; do
# Skip the header row
if [["$name" == "Name"]]; then
continue
fi
MESSAGE="Hello $name,\n\nThank you for registering for the upcoming conference. We are excited to have you!\n\nBest regards,\nEN Corporation"
echo -e "To: $email\nFrom: $FROM_EMAIL\nSubject: $SUBJECT\n\n$MESSAGE" | sudo ssmtp "$email" -C /etc/ssmtp/ssmtp.conf
if [$? -eq 0]; then
echo "Confirmation email sent to $name ($email)."
else
echo "Failed to send confirmation email to $name ($email)."
fi
done < "$CSV_FILE"
exit 0
[line 26/26 (100%), col 1/1 (100%), char 744/744 (100%)]
^G Help ^O Read File ^R Replace ^V Paste ^G Go To Line ^Y Redo M-6 Copy ^Q Where Was M-W Next
^X Exit ^F Where Is ^K Cut ^T Execute ^Z Undo M-A Set Mark M-] To Bracket M-Q Previous ^B Back

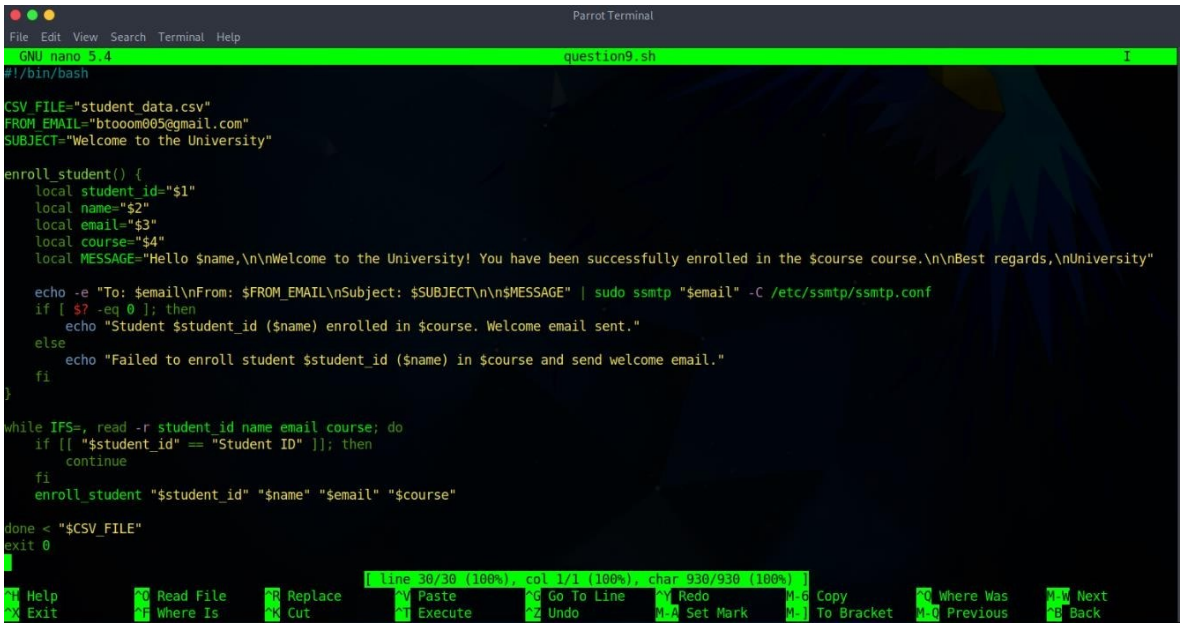

Parrot Terminal
File Edit View Search Terminal Help
GNU nano 5.4 attendees.csv
Name,Email
John Wick,johnwick@example.com
Jane Smith,janesmith@example.com
Amy Johnson,amy@example.com
Ryota Sakamoto,btooom005@gmail.com
[Wrote 5 lines]
^G Help ^O Read File ^R Replace ^V Paste ^G Go To Line ^Y Redo M-6 Copy ^Q Where Was M-W Next
^X Exit ^F Where Is ^K Cut ^T Execute ^Z Undo M-A Set Mark M-] To Bracket M-Q Previous ^B Back

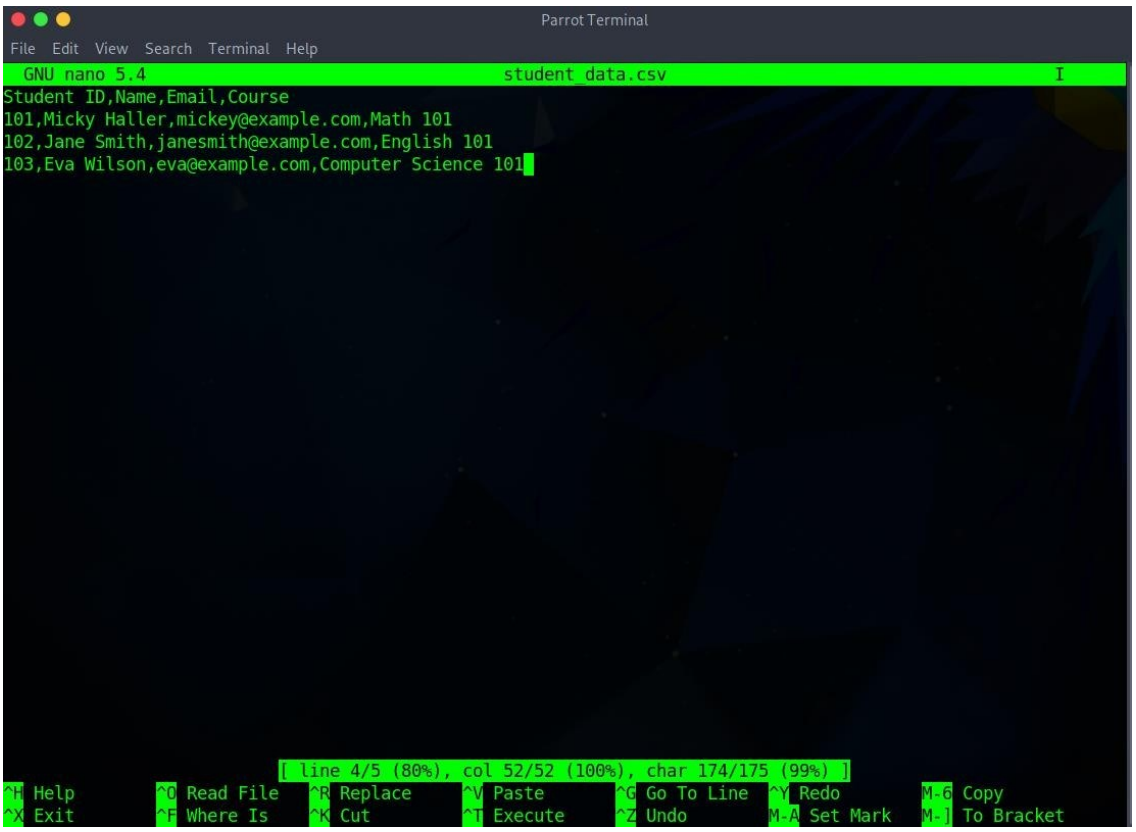

[nitesh@parrot]-[~/Desktop]
$sudo nano question8.sh
[nitesh@parrot]-[~/Desktop]
$sudo nano attendees.csv
[nitesh@parrot]-[~/Desktop]
$sudo chmod +x question8.sh
[nitesh@parrot]-[~/Desktop]
$./question8.sh
Confirmation email sent to John Wick (johnwick@example.com).
Confirmation email sent to Jane Smith (janesmith@example.com).
Confirmation email sent to Amy Johnson (amy@example.com).
Confirmation email sent to Ryota Sakamoto (btooom005@gmail.com).

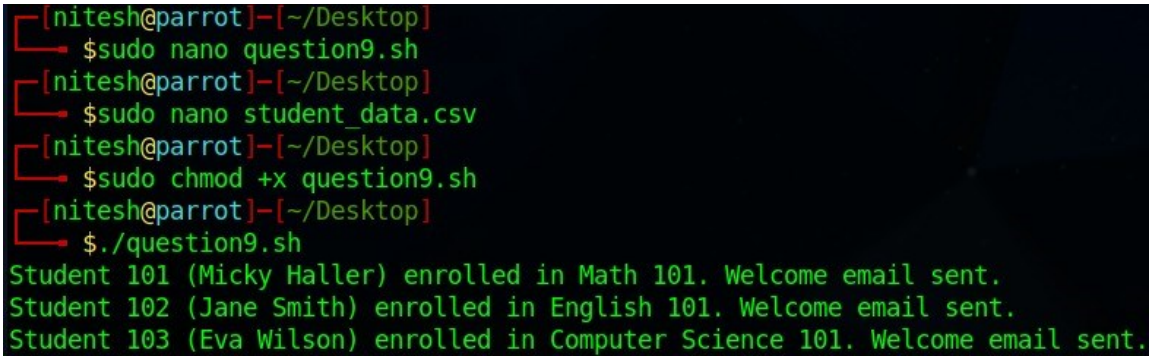


2. Scenario: A university needs to manage student enrollments. They have CSV files with student data and want to enroll students in courses, generate class rosters, and send welcomeemails.

Question: Develop a shell script that processes student data, enrolls students in courses, generates class rosters, and sends welcome emails. Use control structures for enrollment andemail communication.


Parrot Terminal
File Edit View Search Terminal Help
GNU nano 5.4 question9.sh
#!/bin/bash
CSV_FILE="student_data.csv"
FROM_EMAIL="btooom005@gmail.com"
SUBJECT="Welcome to the University"
enroll_student() {
local student_id="$1"
local name="$2"
local email="$3"
local course="$4"
local MESSAGE="Hello $name,\n\nWelcome to the University! You have been successfully enrolled in the $course course.\n\nBest regards,\nUniversity"
echo -e "To: $email\nFrom: $FROM_EMAIL\nSubject: $SUBJECT\n\n$MESSAGE" | sudo ssmtp "$email" -C /etc/ssmtp/ssmtp.conf
if [$? -eq 0]; then
echo "Student $student_id ($name) enrolled in $course. Welcome email sent."
else
echo "Failed to enroll student $student_id ($name) in $course and send welcome email."
fi
}
while IFS=, read -r student_id name email course; do
if [["$student_id" == "Student ID"]]; then
continue
fi
enroll_student "$student_id" "$name" "$email" "$course"
done < "$CSV_FILE"
exit 0
[line 30/30 (100%), col 1/1 (100%), char 930/930 (100%)]
^G Help ^O Read File ^R Replace ^V Paste ^G Go To Line ^Y Redo M-6 Copy ^Q Where Was M-W Next
^X Exit ^F Where Is ^K Cut ^T Execute ^Z Undo M-A Set Mark M-] To Bracket M-Q Previous ^B Back


Parrot Terminal
File Edit View Search Terminal Help
GNU nano 5.4 student_data.csv
Student ID,Name,Email,Course
101,Micky Haller,mickey@example.com,Math 101
102,Jane Smith,janesmith@example.com,English 101
103,Eva Wilson,eva@example.com,Computer Science 101
[line 4/5 (80%), col 52/52 (100%), char 174/175 (99%)]
^G Help ^O Read File ^R Replace ^V Paste ^G Go To Line ^Y Redo M-6 Copy
^X Exit ^F Where Is ^K Cut ^T Execute ^Z Undo M-A Set Mark M-] To Bracket


[nitesh@parrot]-[~/Desktop]
$sudo nano question9.sh
[nitesh@parrot]-[~/Desktop]
$sudo nano student_data.csv
[nitesh@parrot]-[~/Desktop]
$sudo chmod +x question9.sh
[nitesh@parrot]-[~/Desktop]
$./question9.sh
Student 101 (Micky Haller) enrolled in Math 101. Welcome email sent.
Student 102 (Jane Smith) enrolled in English 101. Welcome email sent.
Student 103 (Eva Wilson) enrolled in Computer Science 101. Welcome email sent.

Welcome to the University Inbox ×

R **btooom005@gmail.com**
to mickey

Hello Micky Haller,

Welcome to the University! You have been successfully enrolled in the Math 101 course.

Best regards,
University

Welcome to the University Inbox ×

R **btooom005@gmail.com**
to janesmith

Hello Jane Smith,

Welcome to the University! You have been successfully enrolled in the English 101 course.

Best regards,
University

Welcome to the University Inbox ×

R **btooom005@gmail.com**
to eva

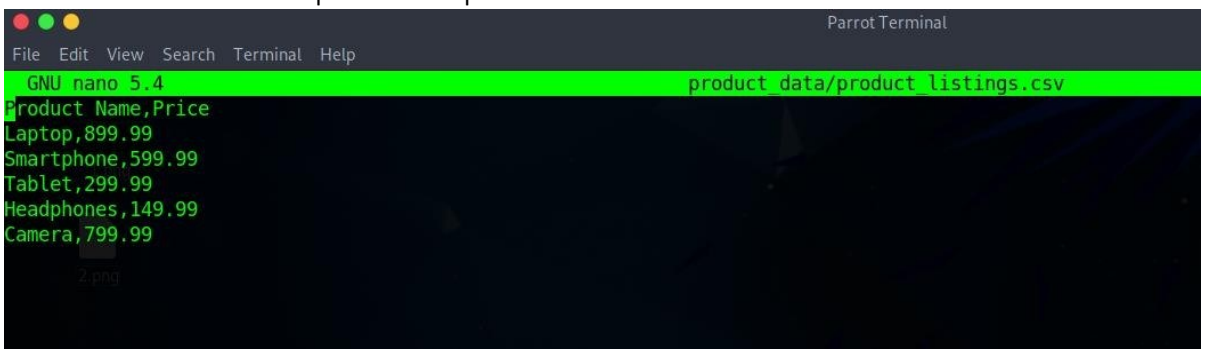
Hello Eva Wilson,

Welcome to the University! You have been successfully enrolled in the Computer Science 101 course.

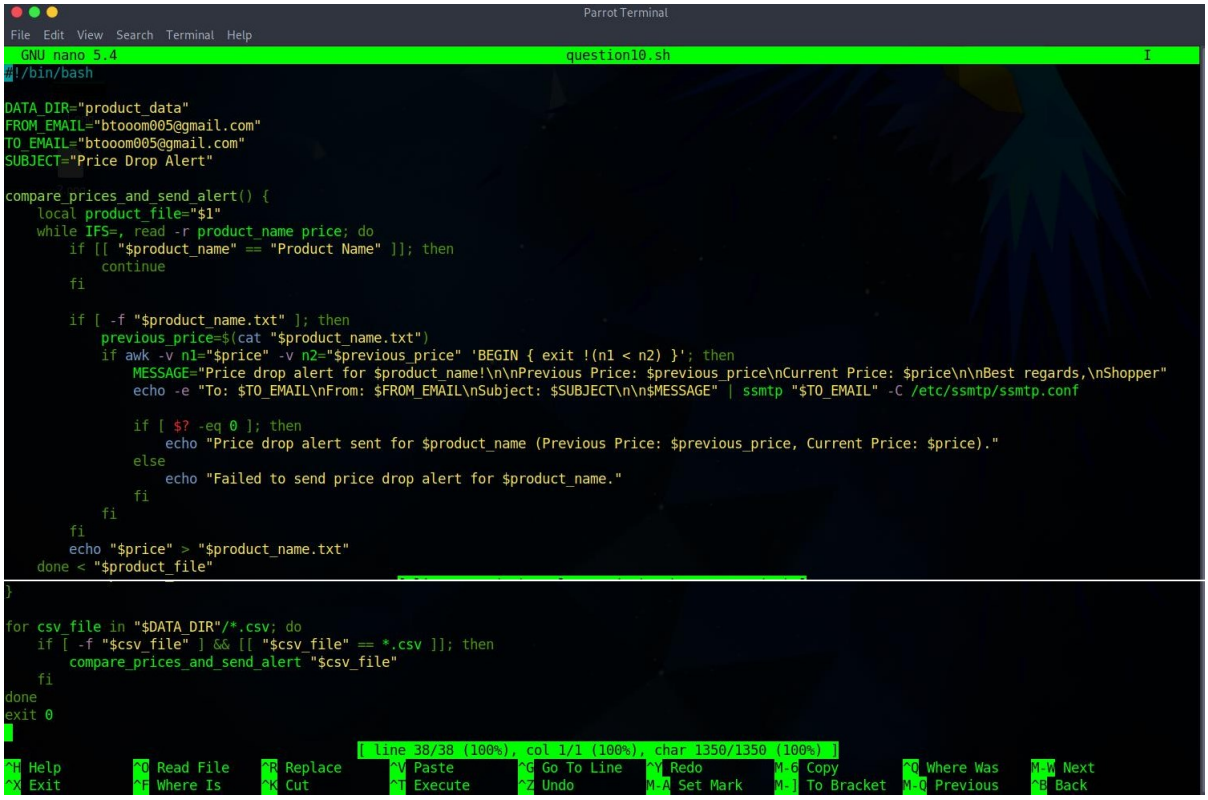
Best regards,
University

3. Scenario: A shopper wants to compare prices of products across multiple online retailers.They have CSV files with product information and prices. They want to find the best deals and receive email notifications for price drops.

Question: Develop a shell script that reads product CSV files, compares prices, identifies thebest deals, and sends email notifications for price drops. Utilize control structures for price comparison and email alerts

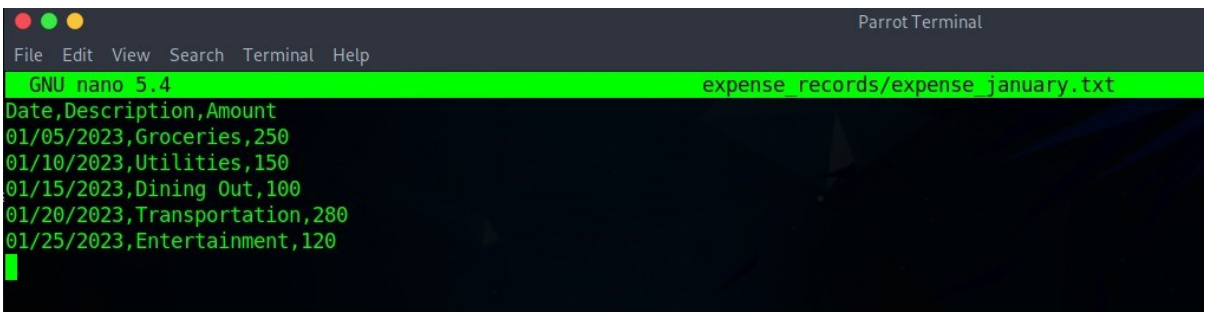


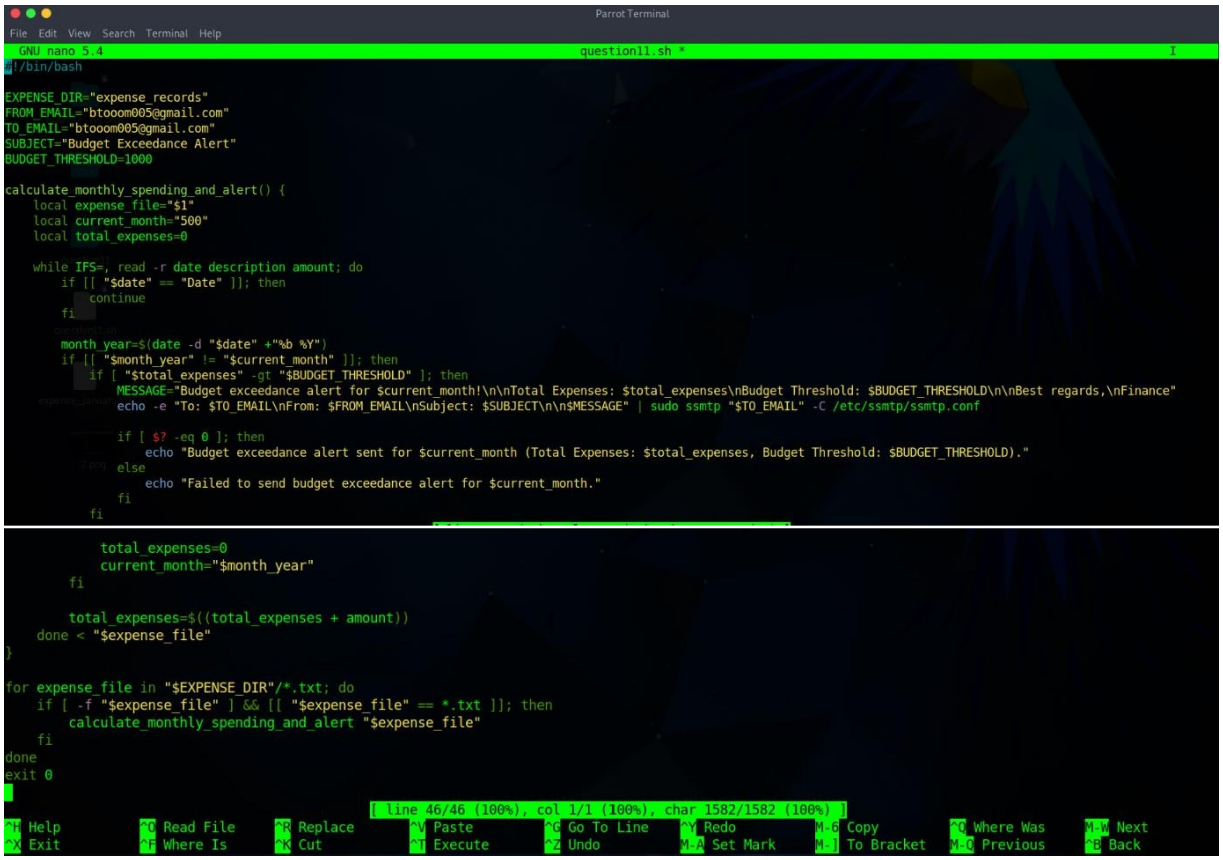
```
┌─[nitesh@parrot]─[~/Desktop]
└──╼ $sudo nano question10.sh
┌─[nitesh@parrot]─[~/Desktop]
└──╼ $sudo nano product_data/product_listings.csv
┌─[nitesh@parrot]─[~/Desktop]
└──╼ $sudo chmod +x question10.sh
┌─[nitesh@parrot]─[~/Desktop]
└──╼ $./question10.sh
```

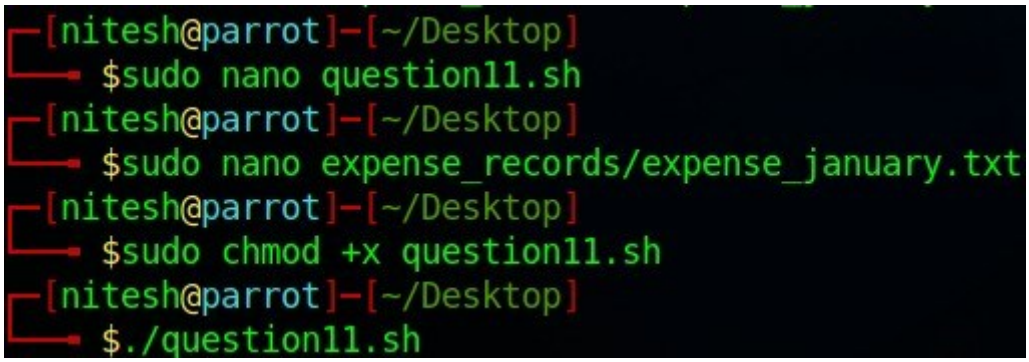


4. Scenario: A financial analyst wants to track personal expenses. They have text files withexpense records and want to generate monthly spending summaries while receiving email alerts for exceeding the budget.

Question: Write a shell script that processes expense text files, calculates monthly spendingsummaries, and sends email alerts for budget exceedance. Implement control structures for expense tracking and email notifications.







5. Scenario: A customer support team receives feedback in text files. They want to analyze customer sentiments, identify recurring issues, and send email notifications for urgent.

Question: Write a shell script that processes customer feedback text files, analyses sentiments, identifies recurring issues, and sends email notifications for urgent cases. Applycontrol structures for sentiment analysis and email al

```
┌─[nitesh@parrot]─[~/Desktop/customer_feedback]
└──╼ $sudo nano question12.sh
┌─[nitesh@parrot]─[~/Desktop/customer_feedback]
└──╼ $sudo nano feedback1.txt
┌─[nitesh@parrot]─[~/Desktop/customer_feedback]
└──╼ $sudo chmod +x question12.sh
┌─[nitesh@parrot]─[~/Desktop/customer_feedback]
└──╼ $./question12.sh
```

Parrot Terminal

File Edit View Search Terminal Help

GNU nano 5.4 question12.sh *

```
#!/bin/bash

FEEDBACK_DIR="///home/nitesh/Desktop/customer_feedback"
FROM_EMAIL="btooom005@gmail.com"
TO_EMAIL="btooom005@gmail.com"
SUBJECT="Urgent Support Needed"

analyze_feedback_and_send_alerts() {
    local feedback_dir="$1"
    for feedback_file in "$feedback_dir"/*.txt; do
        if [ -f "$feedback_file" ] && [[ "$feedback_file" == *.txt ]]; then
            sentiment_score=$(analyze_sentiment "$feedback_file")
            if [ "$sentiment_score" -lt 0 ]; then
                MESSAGE="Urgent support needed for feedback in $feedback_file!\nSentiment Score: $sentiment_score\n\nBest regards,\nCustomer Support"
                echo -e "To: $TO_EMAIL\nFrom: $FROM_EMAIL\nSubject: $SUBJECT\n\n$MESSAGE" | sudo ssmtp "$TO_EMAIL" -C /etc/ssmtp/ssmtp.conf
                if [ $? -eq 0 ]; then
                    echo "Urgent support alert sent for $feedback_file (Sentiment Score: $sentiment_score)."
                else
                    echo "Failed to send urgent support alert for $feedback_file."
                fi
            fi
        fi
    done
}
analyze_sentiment() {
    local feedback_file="$1"
    echo "$((RANDOM % 3 - 1))"
}
analyze_feedback_and_send_alerts "$FEEDBACK_DIR"
exit 0
```

[line 2/31 (6%), col 1/1 (100%), char 12/1211 (0%)]

^H Help ^O Read File ^R Replace ^V Paste ^G Go To Line ^Y Redo M-6 Copy ^Q Where Was M-W Next
^X Exit ^F Where Is ^K Cut ^T Execute ^Z Undo M-A Set Mark M-] To Bracket M-Q Previous ^B Back

Parrot Terminal

File Edit View Search Terminal Help

GNU nano 5.4 feedback1.txt *

```
Customer Feedback - Ticket #12345

Date: 2023-04-15
Customer Name: John Doe
Issue: I am experiencing frequent connection drops with your service.
Sentiment: Negative
Additional Comments:
The service has been unreliable for the past week. I rely on it for my work, and these interruptions are causing significant disruptions to my workflow.
Please address this issue urgently.

---

Date: 2023-04-18
Customer Name: Jane Smith
Issue: The user interface of your software is not user-friendly.
Sentiment: Negative
Additional Comments:
I find it difficult to navigate and perform simple tasks with your software. It's frustrating, and I believe it needs improvement.

---

Date: 2023-04-20
Customer Name: Sarah Johnson
Issue: Great experience with your product!
Sentiment: Positive
Additional Comments:
I just wanted to express my satisfaction with your product. It works seamlessly, and your support team has been helpful when I had questions.

Keep up the good work!
```

[line 30/30 (100%), col 1/1 (100%), char 964/964 (100%)]

^H Help ^O Read File ^R Replace ^V Paste ^G Go To Line ^Y Redo M-6 Copy ^Q Where Was M-W Next
^X Exit ^F Where Is ^K Cut ^T Execute ^Z Undo M-A Set Mark M-] To Bracket M-Q Previous ^B Back

Urgent Support Needed Inbox ×

btooom005@gmail.com 3:34 AM (4 minutes ago)

to me

Urgent support needed for feedback in ///home/nitesh/Desktop/customer_feedback/feedback1.txt!
Sentiment Score: -1

Best regards,
Customer Support

LAB 8: SHELL SCRIPTING- II

AIM: Shell Scripting Exercises

1. Shell Script Program to search whether element is present is in the list or not.

```
┌──(luna㉿Luna)-[~/Desktop/luna]
└─$ nano search.sh

┌──(luna㉿Luna)-[~/Desktop/luna]
└─$ chmod +x search.sh

┌──(luna㉿Luna)-[~/Desktop/luna]
└─$ ./search.sh
Enter the element to search: Phoebe
Element found in the list.
```

```
  GNU nano 7.2                                                search.sh
#!/bin/bash

# Define the list
list=("Phoebe" "Callisto" "Lo" "Titan" "Enceladus")

# Prompt the user to enter the element to search
echo -n "Enter the element to search: "
read element

# Flag to keep track of the element's presence
found=false

# Iterate through the list
for item in "${list[@]}"
do
    # Compare each item with the element
    if [ "$item" == "$element" ]
    then
        found=true
        break
    fi
done
```

```
# Check if element was found or not
if [ "$found" == true ]
then
    echo "Element found in the list."
else
    echo "Element not found in the list."
fi
```

2. Shell Script Program to search whether given file is a directory or not.

```
┌──(luna㉿Luna)-[~/Desktop/luna]
└─$ pwd
/home/luna/Desktop/luna

┌──(luna㉿Luna)-[~/Desktop/luna]
└─$ nano dir_compare.sh

┌──(luna㉿Luna)-[~/Desktop/luna]
└─$ chmod +x dir_compare.sh

┌──(luna㉿Luna)-[~/Desktop/luna]
└─$ ./dir_compare.sh
Enter the file path to search: /home/luna/Desktop/luna
The file is a directory.
```

```
  GNU nano 7.2                                          dir_compare.sh
#!/bin/bash

# Prompt the user to enter the file path to search
echo -n "Enter the file path to search: "
read file_path

# Check if the file exists
if [ -e "$file_path" ]; then
  # Check if the file is a directory
  if [ -d "$file_path" ]; then
    echo "The file is a directory."
  else
    echo "The file is not a directory."
  fi
else
  echo "File not found."
fi
```

3. Shell Script Program to count number of files in a directory.

```
┌──(luna㉿Luna)-[~/Desktop/luna]
└─$ pwd
/home/luna/Desktop/luna

┌──(luna㉿Luna)-[~/Desktop/luna]
└─$ tree
.
├── dir1
├── dir2
├── dir_compare.sh
├── f1.txt
├── file_count.sh
└── search.sh

3 directories, 4 files
```

```
┌──(luna㉿Luna)-[~/Desktop/luna]
└─$ nano file_count.sh

┌──(luna㉿Luna)-[~/Desktop/luna]
└─$ chmod +x file_count.sh

┌──(luna㉿Luna)-[~/Desktop/luna]
└─$ ./file_count.sh
Enter the directory path: /home/luna/Desktop/luna
Number of files in the directory: 4

┌──(luna㉿Luna)-[~/Desktop/luna]
└─$ nano file_count.sh
```

```
  GNU nano 7.2                                          file_count.sh
#!/bin/bash

# Prompt the user to enter the directory path
echo -n "Enter the directory path: "
read directory

# Check if the directory exists
if [ -d "$directory" ]; then
   # Count the number of files in the directory
   count=$(ls -l "$directory" | grep "^-" | wc -l)
   echo "Number of files in the directory: $count"
else
   echo "Directory not found."
fi
```

LAB 9: USER GROUPS

AIM: User Management in Linux Using TMUX

PROCEDURE:

Step-1: Install tmux if not present.

```
┌──(luna㉿Luna)-[~/Desktop/luna]
└─$ sudo apt-get install tmux
[sudo] password for luna:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
tmux is already the newest version (3.3a-4).
0 upgraded, 0 newly installed, 0 to remove and 952 not upgraded.
```

Step-2: Create a new user in you environment.

```
┌──(luna㉿Luna)-[~/Desktop/luna]
└─$ sudo adduser moon
info: Adding user `moon' ...
info: Selecting UID/GID from range 1000 to 59999 ...
info: Adding new group `moon' (1001) ...
info: Adding new user `moon' (1001) with group `moon (1001)' ...
info: Creating home directory `/home/moon' ...
info: Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for moon
Enter the new value, or press ENTER for the default
        Full Name []: Moon Seline
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
info: Adding new user `moon' to supplemental / extra groups `users' ...
info: Adding user `moon' to group `users' ...
```

Step-3*:* Configure the tmux.conf file.

```
┌──(luna㉿Luna)-[~/Desktop/luna]
└─$ sudo nano /home/moon/.tmux.conf
```

```
  GNU nano 7.2                                          /home/moon/.tmux.conf
set -g prefix C-a
bind C-a send-prefix

bind | split-window -h
bind - split-window -v

set -g mouse on
```

Step-4: Configure ownership and permissions for the new user.

```
┌──(luna㉿Luna)-[~/Desktop/luna]
└─$ chown -R moon:moon /home/moon
```

Step-5: Open a new tmux session and create two windows.

Step-6: Login with the new user credentials in the second tmux window.

```
┌──(luna㉿Luna)-[~/Desktop/luna]
└─$ su moon
Password:
┌──(moon㉿Luna)-[/home/luna/Desktop/luna]
└─$ whoami
moon
```

Step-7*:* Switch between the two user windows using (Ctrl+b+0) and (Ctrl+b+1)

```
┌──(moon㉿Luna)-[/home/luna/Desktop/luna]
└─$ whoami
moon

┌──(moon㉿Luna)-[/home/luna/Desktop/luna]
└─$
```

```
┌──(luna㉿Luna)-[~/Desktop/luna]
└─$ whoami
luna

┌──(luna㉿Luna)-[~/Desktop/luna]
└─$
```

LAB 10: NIC BONDING

OBJECTIVE: Configuring NIC Bonding in Linux

PROCEDURE:

1. Installing required package, i.e., ifenslave

```
┌──(luna㉿Luna)-[~/Desktop/luna]
└─$ sudo apt-get install ifenslave
[sudo] password for luna:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ifenslave is already the newest version (2.13).
ifenslave set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 952 not upgraded.
```

2. Editing network configurations using 'sudo nano /etc/network/interfaces'

```
┌──(luna㉿Luna)-[~/Desktop/luna]
└─$ sudo nano /etc/network/interfaces
```

```
  GNU nano 7.2                                  /etc/network/interfaces *
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

auto bond0 inet static
address 10.20.2.15
netmask 255.255.255.0
gateway 10.0.2.255
bond-mode 6
bond-miimon 100
bond-slave none

auto eth1
iface eth1 inet manual
bond-master bond0

auth eth2
iface eth2 inet manual
bond-master bond0
```

3. Loading kernel module

```
┌──(luna㉿Luna)-[~/Desktop/luna]
└─$ sudo nano /etc/modules
```

```
  GNU nano 7.2                                                /etc/modules *
# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.
# Parameters can be specified after the module name.
sudo echo 'bonding' >> /etc/modules
```