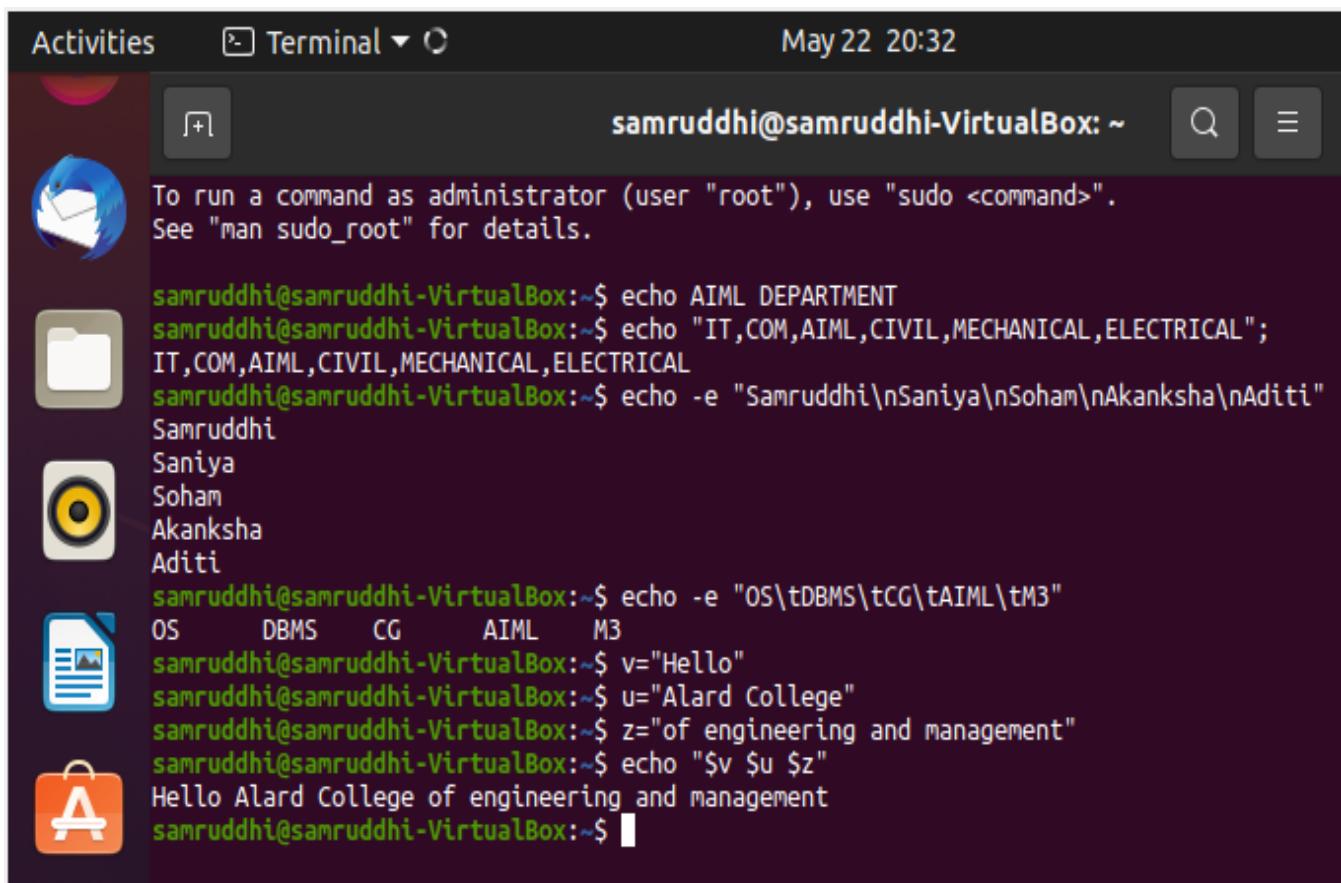


PRACTICAL NO. 1

a. Study of Basic Linux Commands: echo, ls, read, cat, touch, test, loops, arithmetic comparison, conditional loops, grep, sed etc.

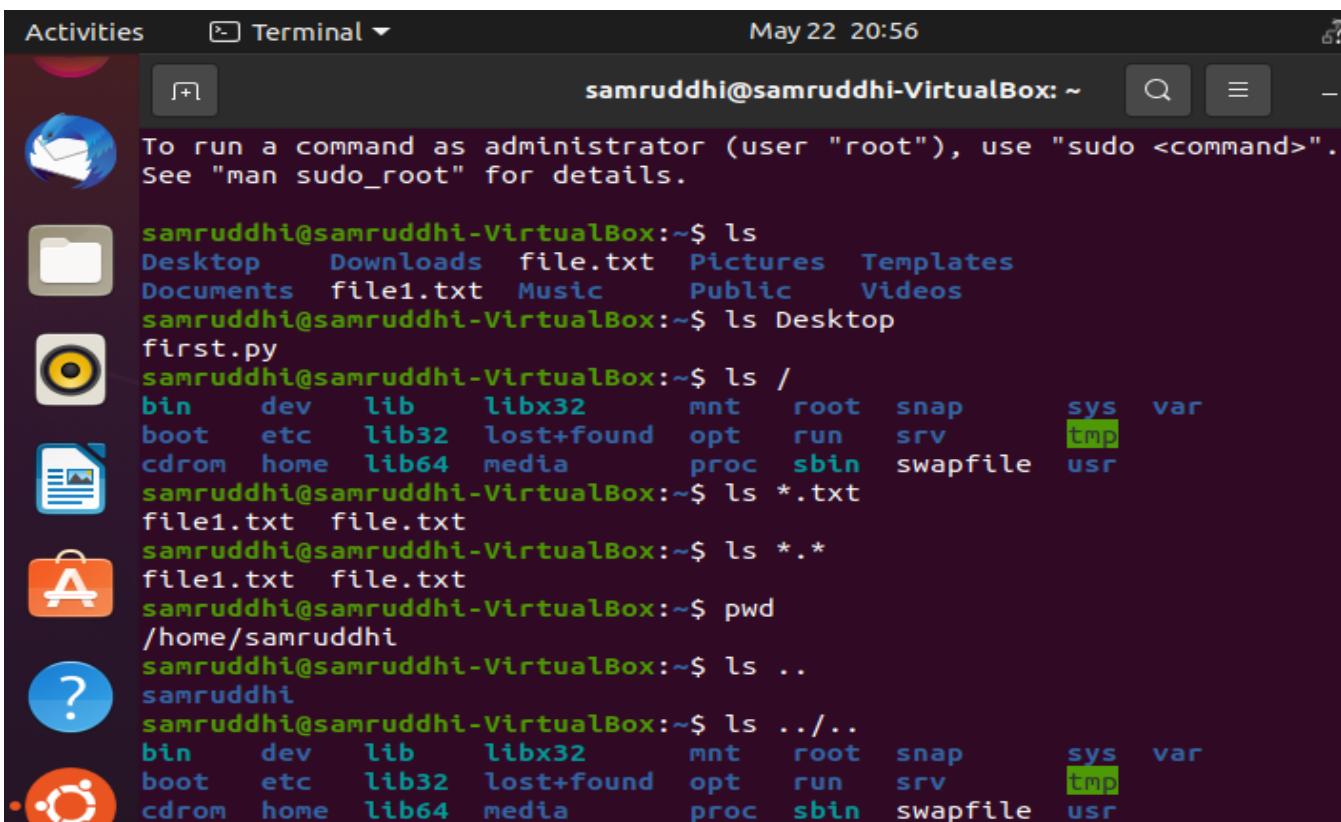
echo command:



The screenshot shows a terminal window in the Unity desktop environment. The terminal title is "Terminal" and the date and time are "May 22 20:32". The user is "samruddhi@samruddhi-VirtualBox". The terminal output shows various examples of the "echo" command:

```
Samruddhi
Saniya
Soham
Akanksha
Aditi
samruddhi@samruddhi-VirtualBox:~$ echo AIML DEPARTMENT
samruddhi@samruddhi-VirtualBox:~$ echo "IT,COM,AIML,CIVIL,MECHANICAL,ELECTRICAL";
IT,COM,AIML,CIVIL,MECHANICAL,ELECTRICAL
samruddhi@samruddhi-VirtualBox:~$ echo -e "Samruddhi\nSaniya\nSoham\nAkanksha\nAditi"
Samruddhi
Saniya
Soham
Akanksha
Aditi
samruddhi@samruddhi-VirtualBox:~$ echo -e "OS\tDBMS\tCG\tAIML\tM3"
OS      DBMS      CG      AIML      M3
samruddhi@samruddhi-VirtualBox:~$ v="Hello"
samruddhi@samruddhi-VirtualBox:~$ u="Alard College"
samruddhi@samruddhi-VirtualBox:~$ z="of engineering and management"
samruddhi@samruddhi-VirtualBox:~$ echo "$v $u $z"
Hello Alard College of engineering and management
samruddhi@samruddhi-VirtualBox:~$
```

ls command:



The screenshot shows a terminal window in the Unity desktop environment. The terminal title is "Terminal" and the date and time are "May 22 20:56". The user is "samruddhi@samruddhi-VirtualBox". The terminal output shows various examples of the "ls" command:

```
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

samruddhi@samruddhi-VirtualBox:~$ ls
Desktop  Downloads  file.txt  Pictures  Templates
Documents  file1.txt  Music  Public  Videos
samruddhi@samruddhi-VirtualBox:~$ ls Desktop
first.py
samruddhi@samruddhi-VirtualBox:~$ ls /
bin  dev  lib  libx32  mnt  root  snap  sys  var
boot  etc  lib32  lost+found  opt  run  srv  tmp
cdrom  home  lib64  media  proc  sbin  swapfile  usr
samruddhi@samruddhi-VirtualBox:~$ ls *.txt
file1.txt  file.txt
samruddhi@samruddhi-VirtualBox:~$ ls *.*
file1.txt  file.txt
samruddhi@samruddhi-VirtualBox:~$ pwd
/home/samruddhi
samruddhi@samruddhi-VirtualBox:~$ ls ..
samruddhi
samruddhi@samruddhi-VirtualBox:~$ ls ../../
bin  dev  lib  libx32  mnt  root  snap  sys  var
boot  etc  lib32  lost+found  opt  run  srv  tmp
cdrom  home  lib64  media  proc  sbin  swapfile  usr
```

```
samruddhi@samruddhi-VirtualBox:~$ ls -l
total 40
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 20:19 Desktop
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Documents
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Downloads
-rw-rw-r-- 1 samruddhi samruddhi 605 May 22 20:45 file1.txt
-rw-rw-r-- 1 samruddhi samruddhi 10 May 22 20:40 file.txt
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Music
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Pictures
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Public
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Templates
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Videos
samruddhi@samruddhi-VirtualBox:~$ ls -a
. .bashrc Documents gnupg Pictures Templates
.. .cache Downloads .local .profile Videos
.bash_history .config file1.txt .mozilla Public
.bash_logout Desktop file.txt Music .ssh
samruddhi@samruddhi-VirtualBox:~$ ls -s
total 40
4 Desktop 4 Downloads 4 file.txt 4 Pictures 4 Templates
4 Documents 4 file1.txt 4 Music 4 Public 4 Videos
```

```
Activities Terminal May 22 20:59
samruddhi@samruddhi-VirtualBox:~$ ls -d */
Desktop/ Downloads/ Pictures/ Templates/
Documents/ Music/ Public/ Videos/
samruddhi@samruddhi-VirtualBox:~$ ls -al
total 88
drwxr-xr-x 16 samruddhi samruddhi 4096 May 22 20:45 .
drwxr-xr-x 3 root root 4096 Feb 24 19:55 ..
-rw----- 1 samruddhi samruddhi 1084 May 22 20:47 .bash_history
-rw-r--r-- 1 samruddhi samruddhi 220 Feb 24 19:55 .bash_logout
-rw-r--r-- 1 samruddhi samruddhi 3771 Feb 24 19:55 .bashrc
drwx----- 14 samruddhi samruddhi 4096 Feb 24 20:24 .cache
drwx----- 15 samruddhi samruddhi 4096 Apr 12 14:24 .config
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 20:19 Desktop
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Documents
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Downloads
-rw-rw-r-- 1 samruddhi samruddhi 605 May 22 20:45 file1.txt
-rw-rw-r-- 1 samruddhi samruddhi 10 May 22 20:40 file.txt
drwx----- 3 samruddhi samruddhi 4096 May 22 19:49 gnupg
drwx----- 3 samruddhi samruddhi 4096 Feb 24 19:59 local
drwx----- 5 samruddhi samruddhi 4096 Feb 24 20:13 mozilla
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Music
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Pictures
-rw-r--r-- 1 samruddhi samruddhi 807 Feb 24 19:55 .profile
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Public
drwx----- 2 samruddhi samruddhi 4096 Feb 24 20:17 ssh
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Templates
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Videos
samruddhi@samruddhi-VirtualBox:~$
```

Activities Terminal May 22 21:02

```
samruddhi@samruddhi-VirtualBox:~$ ls -ls
total 40
4 drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 20:19 Desktop
4 drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Documents
4 drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Downloads
4 -rw-rw-r-- 1 samruddhi samruddhi 605 May 22 20:45 file1.txt
4 -rw-rw-r-- 1 samruddhi samruddhi 10 May 22 20:40 file.txt
4 drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Music
4 drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Pictures
4 drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Public
4 drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Templates
4 drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Videos
samruddhi@samruddhi-VirtualBox:~$ ls -l>file3.txt
samruddhi@samruddhi-VirtualBox:~$ man ls
samruddhi@samruddhi-VirtualBox:~$
```

Activities Terminal May 22 21:03

```
samruddhi@samruddhi-VirtualBox:~$ ls(1) User Commands
NAME
    ls - list directory contents
SYNOPSIS
    ls [OPTION]... [FILE]...
DESCRIPTION
    List information about the FILEs (the current directory by default).
    Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

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

    -a, --all
        do not ignore entries starting with .

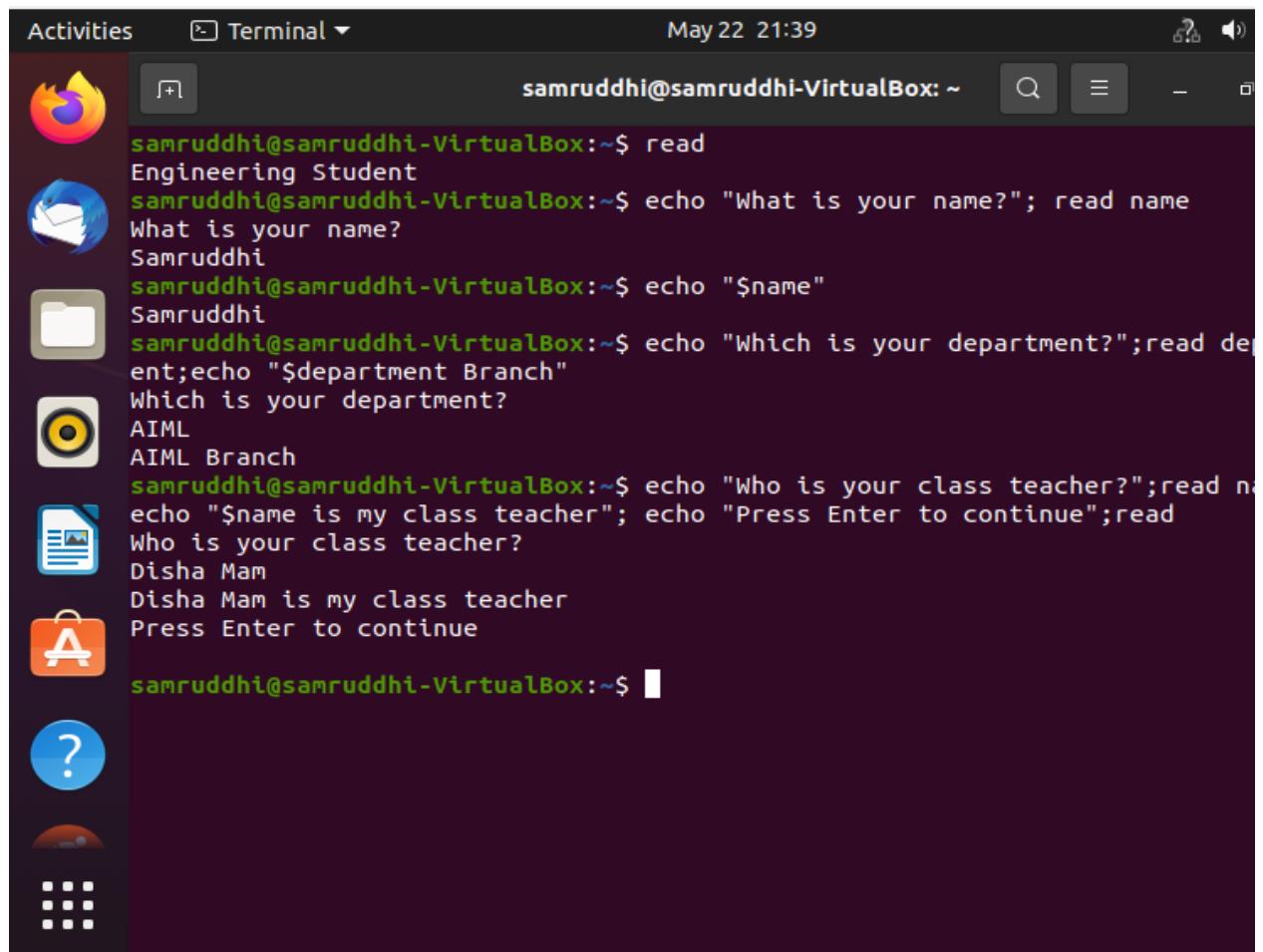
    -A, --almost-all
        do not list implied . and ..

    --author
        with -l, print the author of each file

    -b, --escape
        print C-style escapes for nongraphic characters
samruddhi@samruddhi-VirtualBox:~$
```

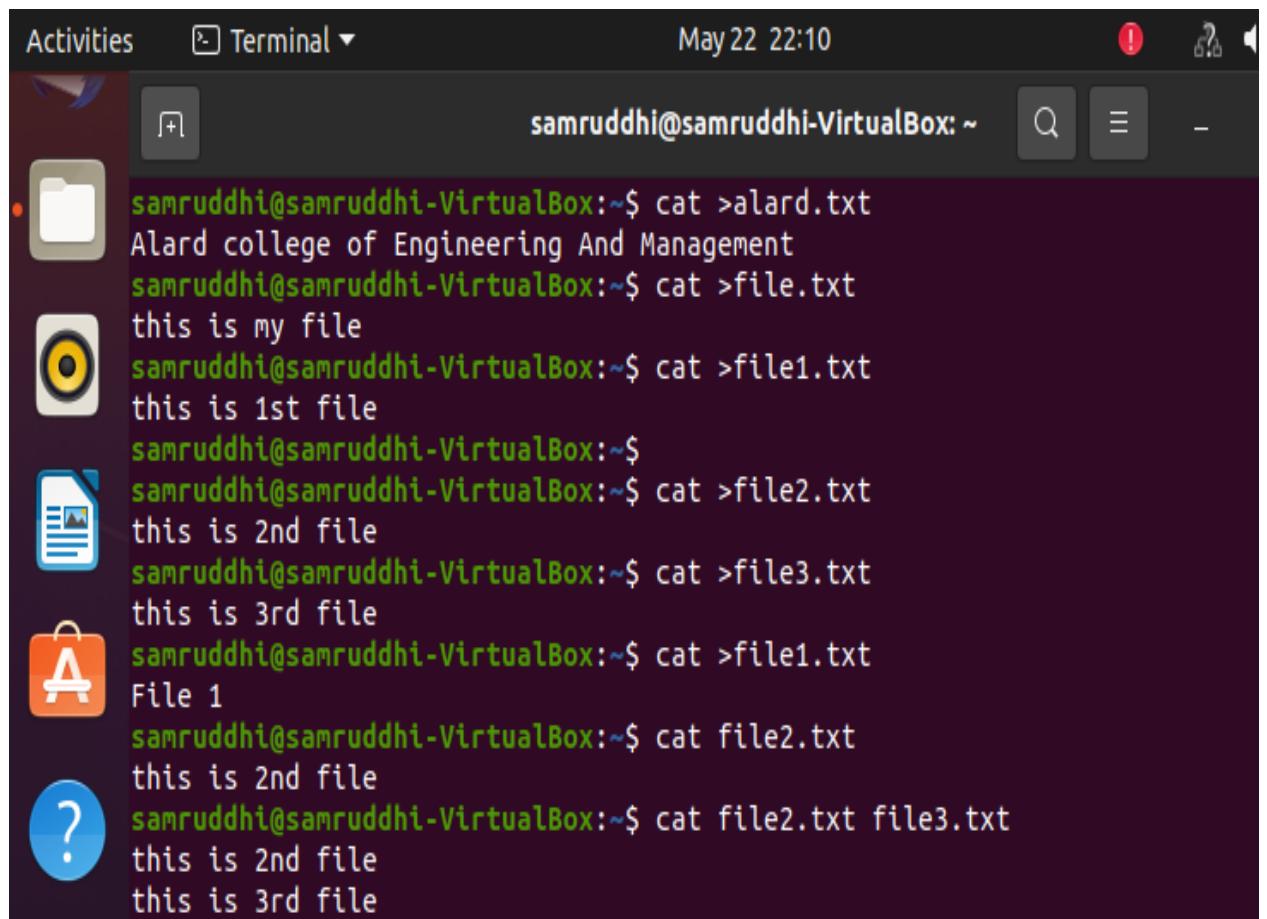
Manual page ls(1) line 1 (press h for help or q to quit)

read command:



```
Activities Terminal May 22 21:39
samruddhi@samruddhi-VirtualBox:~$ read
Engineering Student
samruddhi@samruddhi-VirtualBox:~$ echo "What is your name?"; read name
What is your name?
Samruddhi
samruddhi@samruddhi-VirtualBox:~$ echo "$name"
Samruddhi
samruddhi@samruddhi-VirtualBox:~$ echo "Which is your department?"; read department;echo "$department Branch"
Which is your department?
AIML
AIML Branch
samruddhi@samruddhi-VirtualBox:~$ echo "Who is your class teacher?";read name
echo "$name is my class teacher"; echo "Press Enter to continue";read
Who is your class teacher?
Disha Mam
Disha Mam is my class teacher
Press Enter to continue
samruddhi@samruddhi-VirtualBox:~$
```

cat command:



```
Activities Terminal May 22 22:10
samruddhi@samruddhi-VirtualBox:~$ cat >alard.txt
Alard college of Engineering And Management
samruddhi@samruddhi-VirtualBox:~$ cat >file.txt
this is my file
samruddhi@samruddhi-VirtualBox:~$ cat >file1.txt
this is 1st file
samruddhi@samruddhi-VirtualBox:~$
samruddhi@samruddhi-VirtualBox:~$ cat >file2.txt
this is 2nd file
samruddhi@samruddhi-VirtualBox:~$ cat >file3.txt
this is 3rd file
samruddhi@samruddhi-VirtualBox:~$ cat >file1.txt
File 1
samruddhi@samruddhi-VirtualBox:~$ cat file2.txt
this is 2nd file
samruddhi@samruddhi-VirtualBox:~$ cat file2.txt file3.txt
this is 2nd file
this is 3rd file
```

Activities Terminal May 22 22:17

```
samruddhi@samruddhi-VirtualBox:~$ cat >file.txt
Samruddhi

Akanksha
Soham

Saniya
Aditi
samruddhi@samruddhi-VirtualBox:~$ cat -n file.txt
 1 Samruddhi
 2
 3 Akanksha
 4 Soham
 5
 6
 7 Saniya
 8
 9 Aditi
```

Activities Terminal May 22 22:17

```
samruddhi@samruddhi-VirtualBox:~$ cat -b file.txt
 1 Samruddhi

 2 Akanksha
 3 Soham

 4 Saniya
 5 Aditi
samruddhi@samruddhi-VirtualBox:~$ cat -s file.txt
Samruddhi
Akanksha
Soham
Saniya
Aditi
samruddhi@samruddhi-VirtualBox:~$ cat file.txt
Samruddhi
Akanksha
Soham
Saniya
Aditi
```

Activities Terminal May 22 22:19 samruddhi@samruddhi-VirtualBox: ~

- Akanksha Soham
- Saniya
- Aditi

```
cat file.txt
Samruddhi
```

- Akanksha Soham
- Saniya
- Aditi

```
cat file2.txt
this is 2nd file
```

```
cat file1.txt
File 1
this is 2nd file
```

```
cat file2.txt>>file1.txt
cat file1.txt
File 1
this is 2nd file
this is 2nd file
```

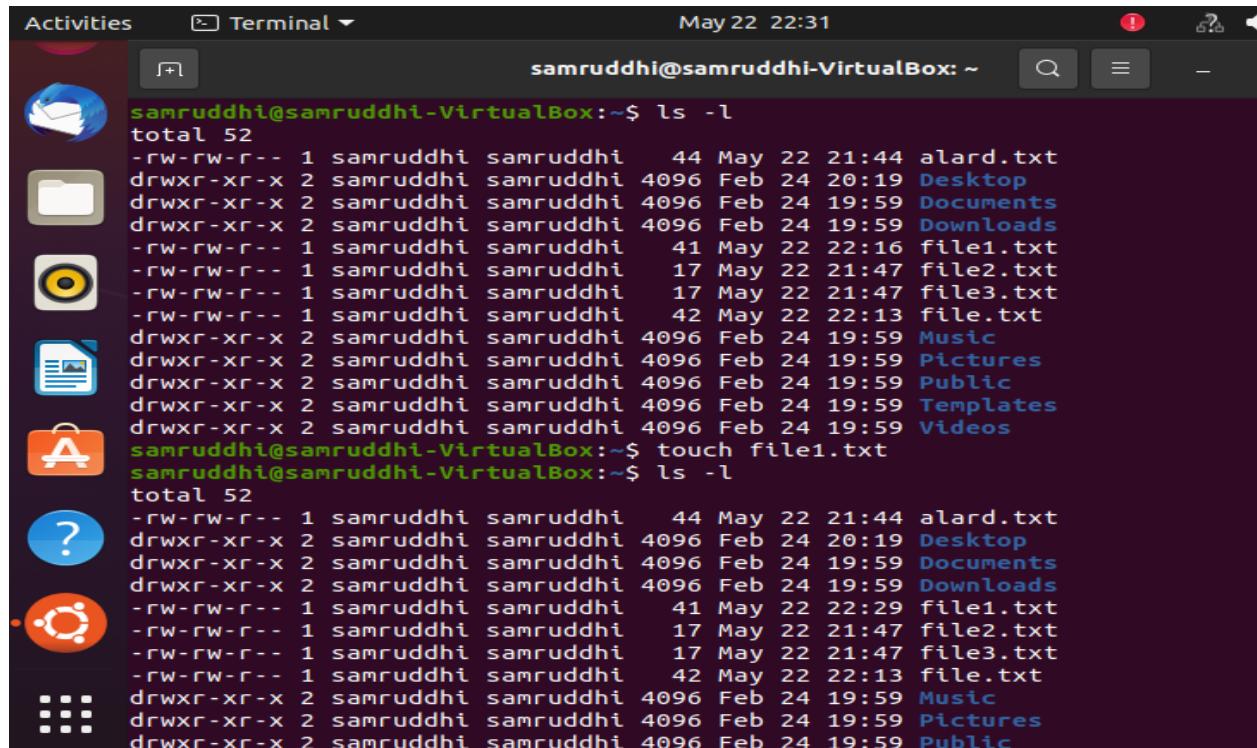
```
man cat
```

Activities Terminal May 22 22:19 samruddhi@samruddhi-VirtualBox: ~

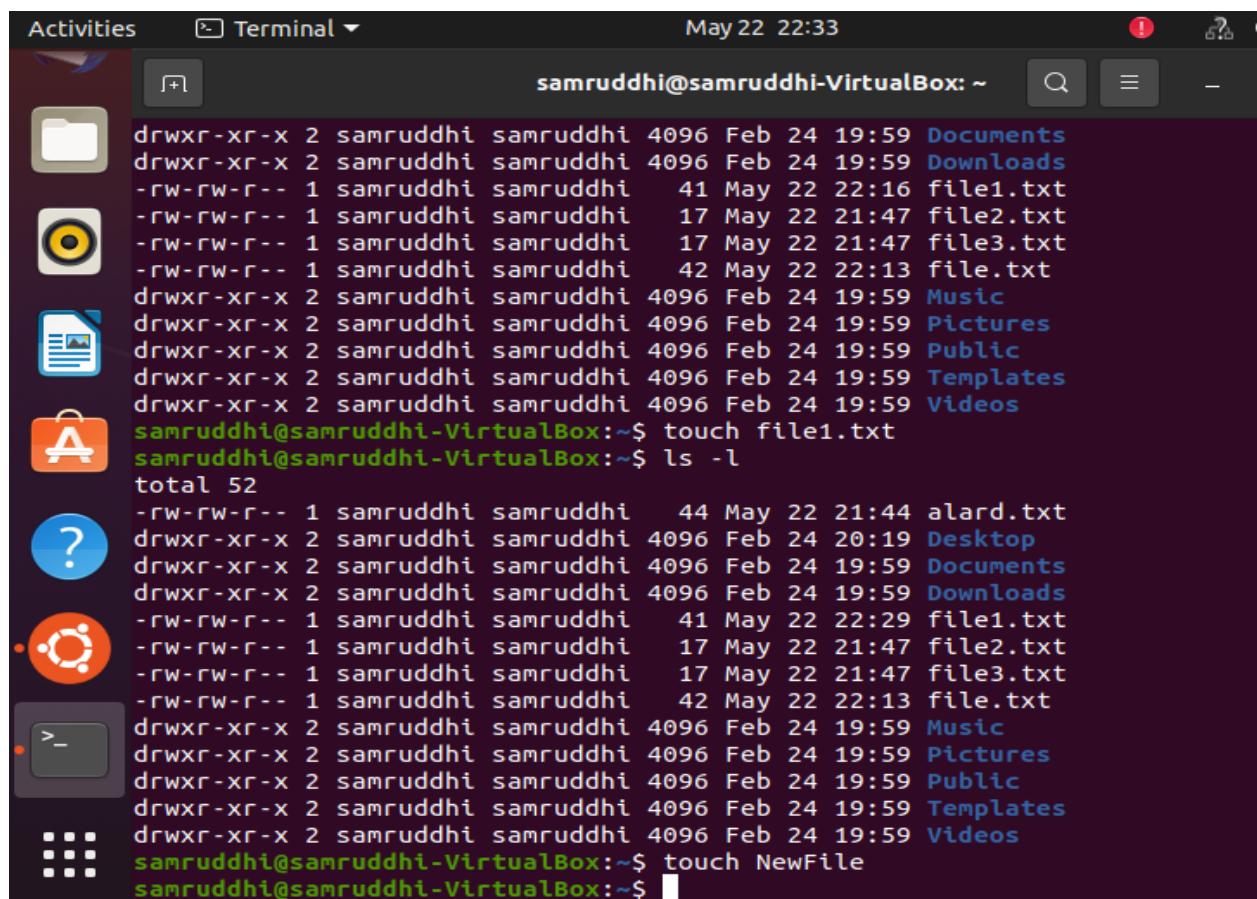
- CAT(1) User Commands
- NAME cat - concatenate files and print on the standard output
- SYNOPSIS cat [OPTION]... [FILE]...
- DESCRIPTION Concatenate FILE(s) to standard output.
- With no FILE, or when FILE is -, read standard input.
- A, --show-all equivalent to -VET
- b, --number-nonblank number nonempty 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

Manual page cat(1) line 1 (press h for help or q to quit)

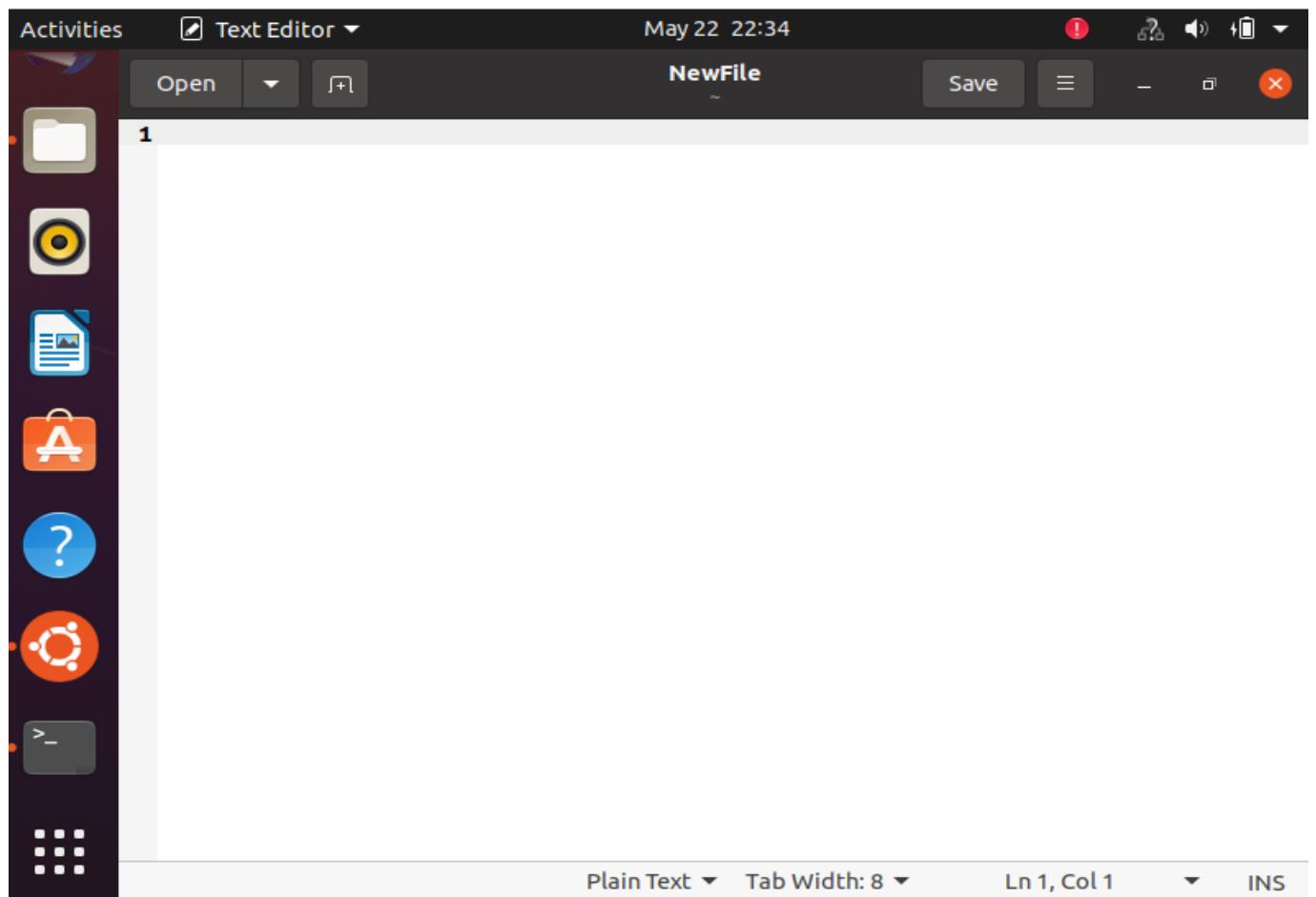
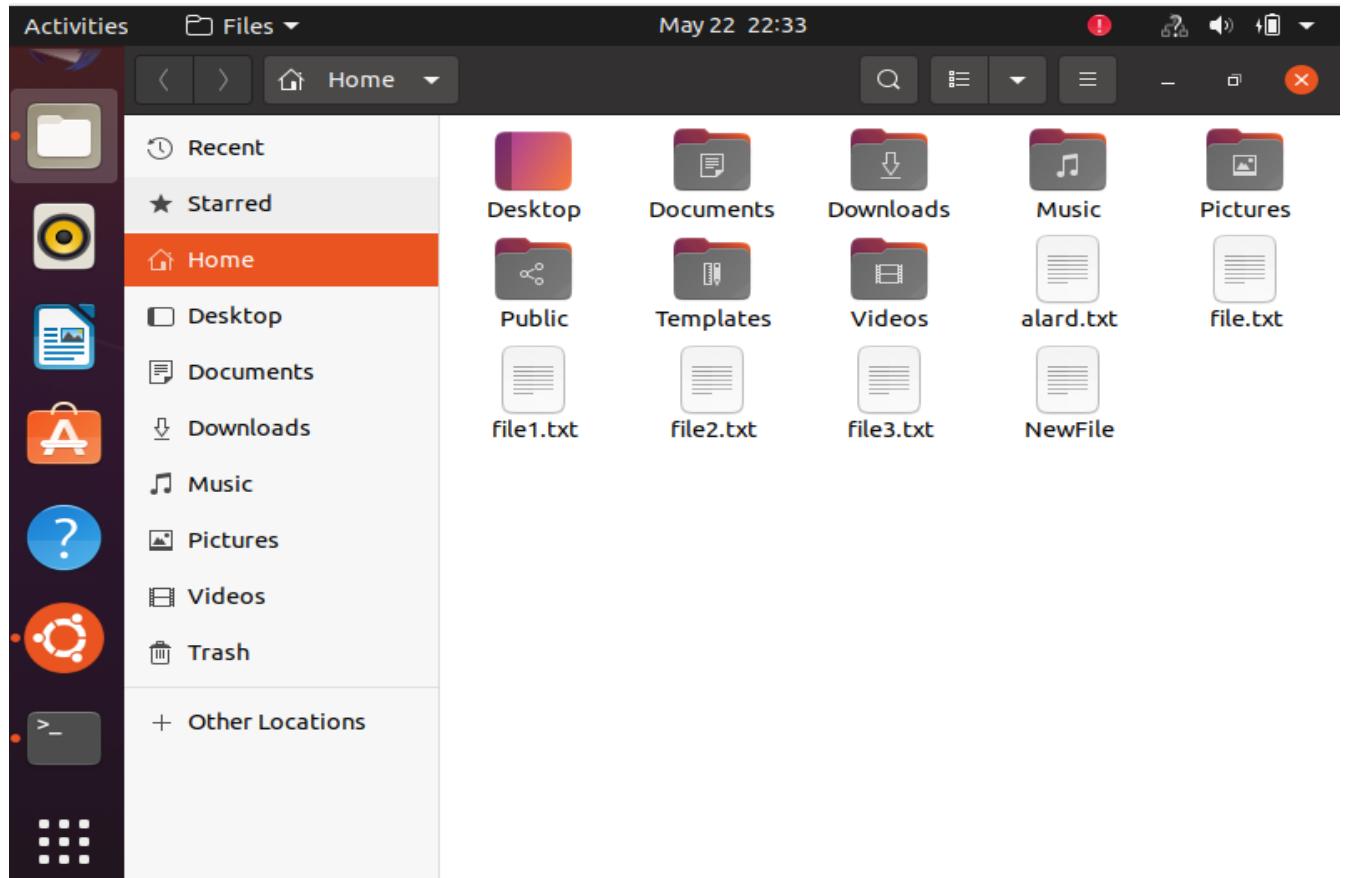
touch command:



```
Activities Terminal May 22 22:31 samruddhi@samruddhi-VirtualBox: ~
samruddhi@samruddhi-VirtualBox:~$ ls -l
total 52
-rw-rw-r-- 1 samruddhi samruddhi 44 May 22 21:44 alard.txt
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 20:19 Desktop
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Documents
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Downloads
-rw-rw-r-- 1 samruddhi samruddhi 41 May 22 22:16 file1.txt
-rw-rw-r-- 1 samruddhi samruddhi 17 May 22 21:47 file2.txt
-rw-rw-r-- 1 samruddhi samruddhi 42 May 22 22:13 file.txt
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Music
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Pictures
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Public
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Templates
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Videos
samruddhi@samruddhi-VirtualBox:~$ touch file1.txt
samruddhi@samruddhi-VirtualBox:~$ ls -l
total 52
-rw-rw-r-- 1 samruddhi samruddhi 44 May 22 21:44 alard.txt
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 20:19 Desktop
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Documents
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Downloads
-rw-rw-r-- 1 samruddhi samruddhi 41 May 22 22:29 file1.txt
-rw-rw-r-- 1 samruddhi samruddhi 17 May 22 21:47 file2.txt
-rw-rw-r-- 1 samruddhi samruddhi 17 May 22 21:47 file3.txt
-rw-rw-r-- 1 samruddhi samruddhi 42 May 22 22:13 file.txt
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Music
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Pictures
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Public
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Templates
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Videos
```



```
Activities Terminal May 22 22:33 samruddhi@samruddhi-VirtualBox: ~
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Documents
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Downloads
-rw-rw-r-- 1 samruddhi samruddhi 41 May 22 22:16 file1.txt
-rw-rw-r-- 1 samruddhi samruddhi 17 May 22 21:47 file2.txt
-rw-rw-r-- 1 samruddhi samruddhi 17 May 22 21:47 file3.txt
-rw-rw-r-- 1 samruddhi samruddhi 42 May 22 22:13 file.txt
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Music
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Pictures
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Public
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Templates
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Videos
samruddhi@samruddhi-VirtualBox:~$ touch file1.txt
samruddhi@samruddhi-VirtualBox:~$ ls -l
total 52
-rw-rw-r-- 1 samruddhi samruddhi 44 May 22 21:44 alard.txt
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 20:19 Desktop
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Documents
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Downloads
-rw-rw-r-- 1 samruddhi samruddhi 41 May 22 22:29 file1.txt
-rw-rw-r-- 1 samruddhi samruddhi 17 May 22 21:47 file2.txt
-rw-rw-r-- 1 samruddhi samruddhi 17 May 22 21:47 file3.txt
-rw-rw-r-- 1 samruddhi samruddhi 42 May 22 22:13 file.txt
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Music
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Pictures
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Public
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Templates
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Videos
samruddhi@samruddhi-VirtualBox:~$ touch NewFile
samruddhi@samruddhi-VirtualBox:~$
```



Activities Terminal May 22 22:35

```
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Downloads
-rw-rw-r-- 1 samruddhi samruddhi   41 May 22 22:16 file1.txt
-rw-rw-r-- 1 samruddhi samruddhi   17 May 22 21:47 file2.txt
-rw-rw-r-- 1 samruddhi samruddhi   17 May 22 21:47 file3.txt
-rw-rw-r-- 1 samruddhi samruddhi   42 May 22 22:13 file.txt
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Music
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Pictures
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Public
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Templates
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Videos
samruddhi@samruddhi-VirtualBox:~$ touch file1.txt
samruddhi@samruddhi-VirtualBox:~$ ls -l
total 52
-rw-rw-r-- 1 samruddhi samruddhi   44 May 22 21:44 alard.txt
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 20:19 Desktop
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Documents
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Downloads
-rw-rw-r-- 1 samruddhi samruddhi   41 May 22 22:29 file1.txt
-rw-rw-r-- 1 samruddhi samruddhi   17 May 22 21:47 file2.txt
-rw-rw-r-- 1 samruddhi samruddhi   17 May 22 21:47 file3.txt
-rw-rw-r-- 1 samruddhi samruddhi   42 May 22 22:13 file.txt
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Music
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Pictures
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Public
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Templates
drwxr-xr-x 2 samruddhi samruddhi 4096 Feb 24 19:59 Videos
samruddhi@samruddhi-VirtualBox:~$ touch NewFile
samruddhi@samruddhi-VirtualBox:~$ man touch
samruddhi@samruddhi-VirtualBox:~$
```

test command:

Activities Terminal May 22 22:44

```
samruddhi@samruddhi-VirtualBox:~$ test 1 -eq 1
samruddhi@samruddhi-VirtualBox:~$ echo $?
0
samruddhi@samruddhi-VirtualBox:~$ test 1 -eq 2
samruddhi@samruddhi-VirtualBox:~$ echo $?
1
samruddhi@samruddhi-VirtualBox:~$ test "string"="string"
samruddhi@samruddhi-VirtualBox:~$ echo $?
0
samruddhi@samruddhi-VirtualBox:~$
```

loops command:

1) for loop

Example 1

A screenshot of a Kali Linux desktop environment in Oracle VM VirtualBox. The terminal window shows the following script:

```
GNU nano 7.2
#!/bin/bash
for (( i=1; i<=5; i++ ))
do
    echo "Number: $i"
done
```

The terminal window title is "File System".

A screenshot of a Kali Linux terminal window in Oracle VM VirtualBox. The session starts with:

```
(kali㉿samruddhi)-[~]
$ sudo su
```

Then, the user enters the password for kali. After becoming root:

```
[root㉿samruddhi)-[/home/kali]
# touch forloop3.sh
```

```
[root㉿samruddhi)-[/home/kali]
# nano forloop3.sh
```

```
[root㉿samruddhi)-[/home/kali]
# chmod +x forloop3.sh
```

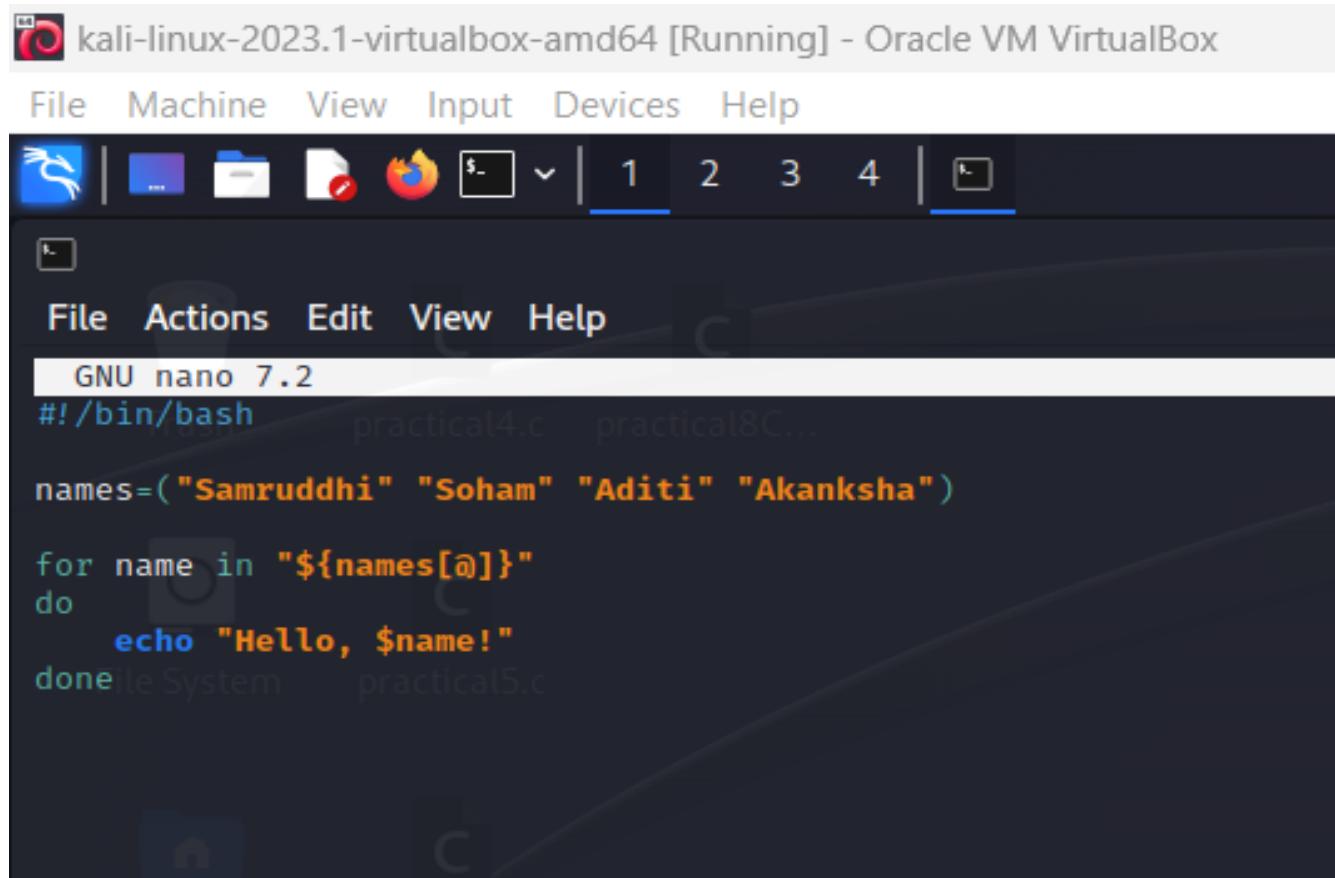
```
[root㉿samruddhi)-[/home/kali]
# ./forloop3.sh
```

The script output is:

```
Number: 1
Number: 2
Number: 3
Number: 4
Number: 5
```

```
[root㉿samruddhi)-[/home/kali]
#
```

Example 2



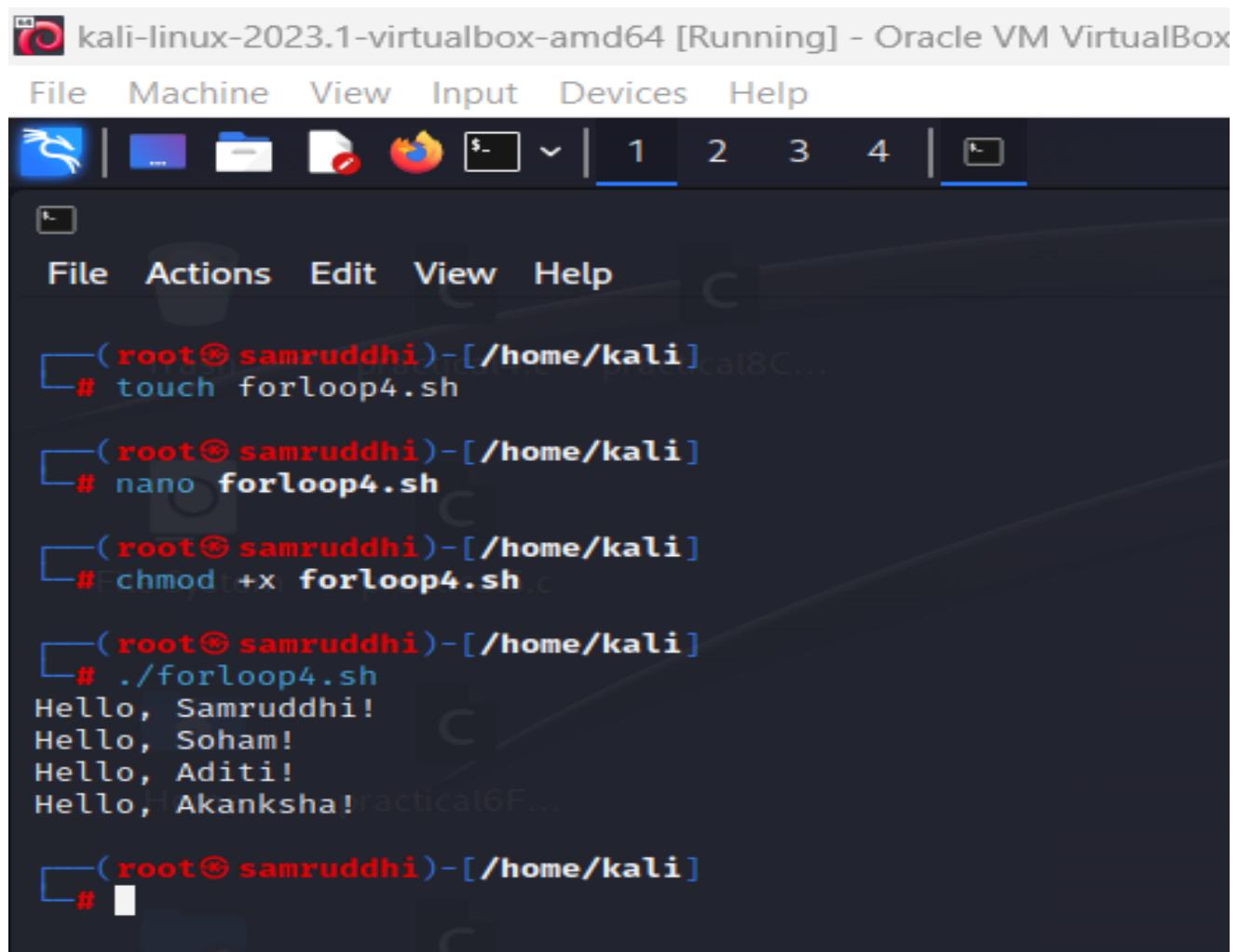
kali-linux-2023.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

File Actions Edit View Help

GNU nano 7.2

```
#!/bin/bash      practical4.c  practical8C...  
  
names=("Samruddhi" "Soham" "Aditi" "Akanksha")  
  
for name in "${names[@]}"  
do  
    echo "Hello, $name!"  
done
```



kali-linux-2023.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

File Actions Edit View Help

```
[root@samruddhi]~[/home/kali]# touch forloop4.sh  
[root@samruddhi]~[/home/kali]# nano forloop4.sh  
[root@samruddhi]~[/home/kali]# chmod +x forloop4.sh  
[root@samruddhi]~[/home/kali]# ./forloop4.sh  
Hello, Samruddhi!  
Hello, Soham!  
Hello, Aditi!  
Hello, Akanksha!
```

[root@samruddhi]~[/home/kali]#

Example 3

A screenshot of a Kali Linux desktop environment within Oracle VM VirtualBox. The window title is "kali-linux-2023.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox". The desktop interface includes a menu bar with File, Machine, View, Input, Devices, Help, and a toolbar with icons for desktop, file manager, terminal, and browser. A taskbar at the bottom shows four windows labeled 1, 2, 3, and 4. The main window contains a terminal session titled "GNU nano 7.2" with the following code:

```
#!/bin/bash practical4.c practical8C...
for letter in {A..Z}
do
    echo $letter
done
```

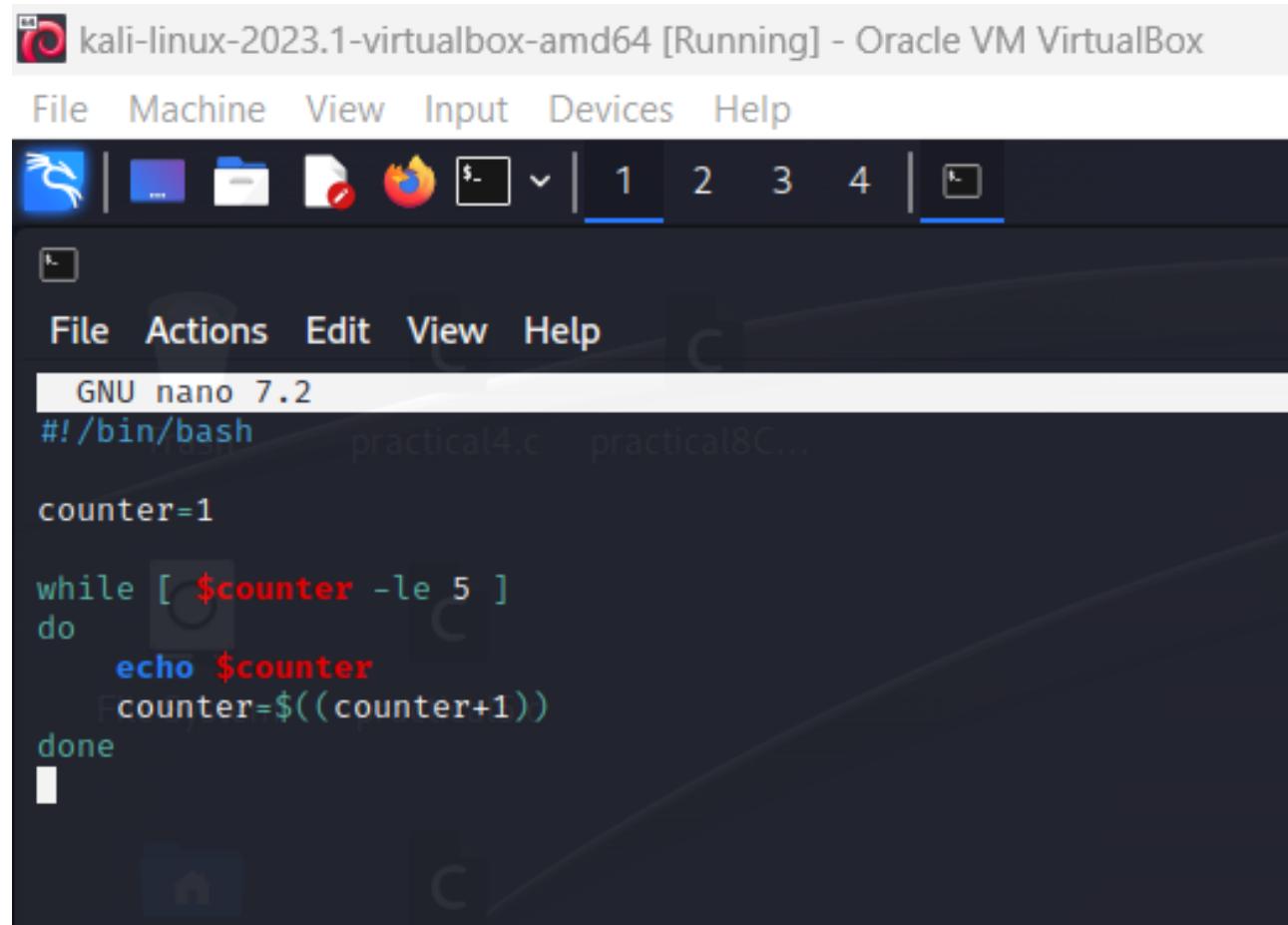
A screenshot of a Kali Linux desktop environment within Oracle VM VirtualBox. The window title is "kali-linux-2023.1-virtualbox-amd64 [Running]". The desktop interface includes a menu bar with File, Machine, View, Input, Devices, Help, and a toolbar with icons for desktop, file manager, terminal, and browser. A taskbar at the bottom shows two windows labeled 1 and 2. The main window contains a terminal session titled "root@samruddhi:[/home/kali]" with the following commands:

```
[root@samruddhi ~]# touch forloop5.sh
[root@samruddhi ~]# nano forloop5.sh
[root@samruddhi ~]# chmod +x forloop5.sh
[root@samruddhi ~]# ./forloop5.sh
```

The desktop background shows a file tree with folders for "Home", "practical6F...", "practical6L...", "practical6...", and "practical7.c". A vertical sidebar on the left lists letters from A to Z.

2) while loop

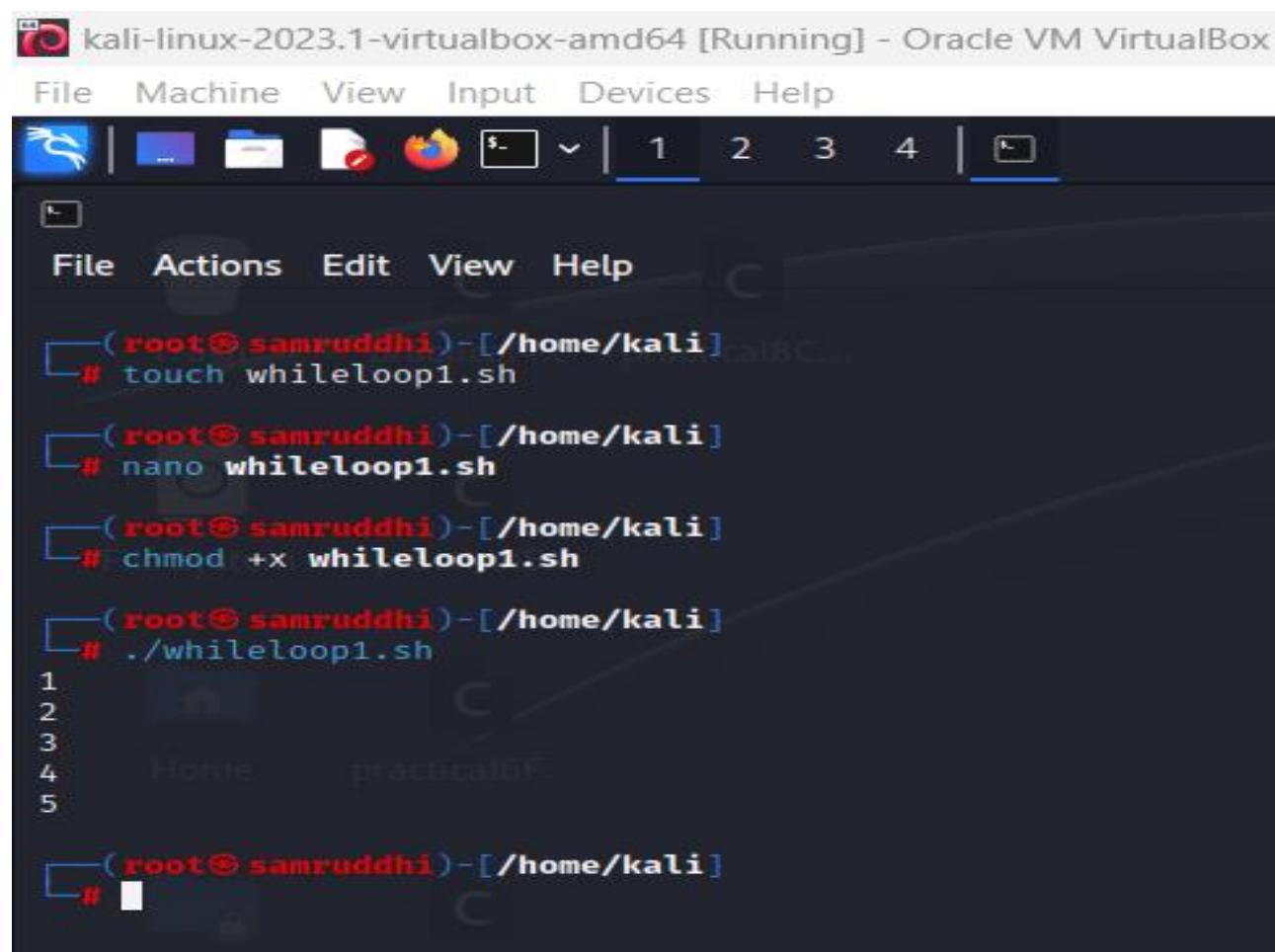
Example 1



A screenshot of a Kali Linux desktop environment. In the foreground, a terminal window titled "GNU nano 7.2" is open, displaying a shell script. The script starts with "#!/bin/bash", initializes a variable "counter=1", and then enters a "while" loop with the condition "\$counter -le 5". Inside the loop, it prints the value of "counter" using "echo \$counter" and increments it by 1 using "counter=\$((counter+1))". The loop concludes with a "done" statement. The terminal window has a dark background with light-colored text. The desktop interface above the terminal includes a menu bar with "File", "Machine", "View", "Input", "Devices", and "Help". A dock at the bottom contains icons for various applications like a file manager, terminal, and browser.

```
GNU nano 7.2
#!/bin/bash
counter=1

while [ $counter -le 5 ]
do
    echo $counter
    counter=$((counter+1))
done
```



A screenshot of a Kali Linux terminal window. The terminal prompt shows the user is root on a machine named "samruddhi" in the "/home/kali" directory. The user has run several commands to create a script, give it execute permissions, and then execute it. The output of the script is shown as a numbered list from 1 to 5. The terminal has a dark background with light-colored text. The desktop interface above the terminal is visible, showing a menu bar and application icons.

```
[root@samruddhi]# touch whileloop1.sh
[root@samruddhi]# nano whileloop1.sh
[root@samruddhi]# chmod +x whileloop1.sh
[root@samruddhi]# ./whileloop1.sh
1
2
3
4
5
```

Example 2

```
GNU nano 7.2
#!/bin/bash

while true
do
    read -p "Enter a number (0 to quit): " num
    if [ $num -eq 0 ]
    then
        break
    fi
    echo "Squared value: $((num * num))"
done
```

```
(root@samruddhi)-[/home/kali]
# touch whileloop2.sh

(root@samruddhi)-[/home/kali]
# nano whileloop2.sh

(root@samruddhi)-[/home/kali]
# chmod +x whileloop2.sh

(root@samruddhi)-[/home/kali]
# ./whileloop2.sh
Enter a number (0 to quit): 1
Squared value: 1
Enter a number (0 to quit): 3
Squared value: 9
Enter a number (0 to quit): 4
Squared value: 16
Enter a number (0 to quit):
```

3) until loop

Example 1

The screenshot shows a Kali Linux desktop environment running in Oracle VM VirtualBox. A terminal window is open in the foreground, displaying a script in nano editor. The script uses an until loop to increment a counter variable until it reaches 5.

```
GNU nano 7.2
#!/bin/bash
counter=1

until [ $counter -gt 5 ]
do
    echo $counter
    counter=$((counter+1))
done
```

The screenshot shows a Kali Linux desktop environment running in Oracle VM VirtualBox. A terminal window is open in the foreground, showing the creation and execution of a shell script named untilloop1.sh. The script uses an until loop to increment a counter variable until it reaches 5.

```
(root@samruddhi)-[~/kali]
# touch untilloop1.sh

(root@samruddhi)-[~/kali]
# nano untilloop1.sh

(root@samruddhi)-[~/kali]
# chmod +x untilloop1.sh

(root@samruddhi)-[~/kali]
# ./untilloop1.sh
1
2
3
4
5

(root@samruddhi)-[~/kali]
```

Example 2

The screenshot shows a Kali Linux terminal window titled "kali-linux-2023.1-virtualbox-amd64 [Running]". The terminal interface includes a menu bar with File, Machine, View, Input, Devices, and Help. Below the menu is a toolbar with icons for Home, File, Copy, Paste, and others. The main area displays a file named "untilloop3.sh" being edited in the nano text editor. The script content is as follows:

```
GNU nano 7.2
#!/bin/bash

sum=0
limit=20
num=1

until [ $sum -ge $limit ]
do
    sum=$((sum+num))
    num=$((num+1))
done

echo "Sum: $sum"
```

At the bottom of the terminal window, there are tabs for "practical4.c", "practical5.c", and "practical6F...".

The screenshot shows a Kali Linux terminal window titled "kali-linux-2023.1-virtualbox-amd64 [Running] - Oracle". The terminal interface is similar to the previous one, with a menu bar, toolbar, and nano text editor displaying the same "untilloop3.sh" script. The script's purpose is to calculate the sum of integers from 1 to 20. The terminal output shows the command being run and the resulting sum:

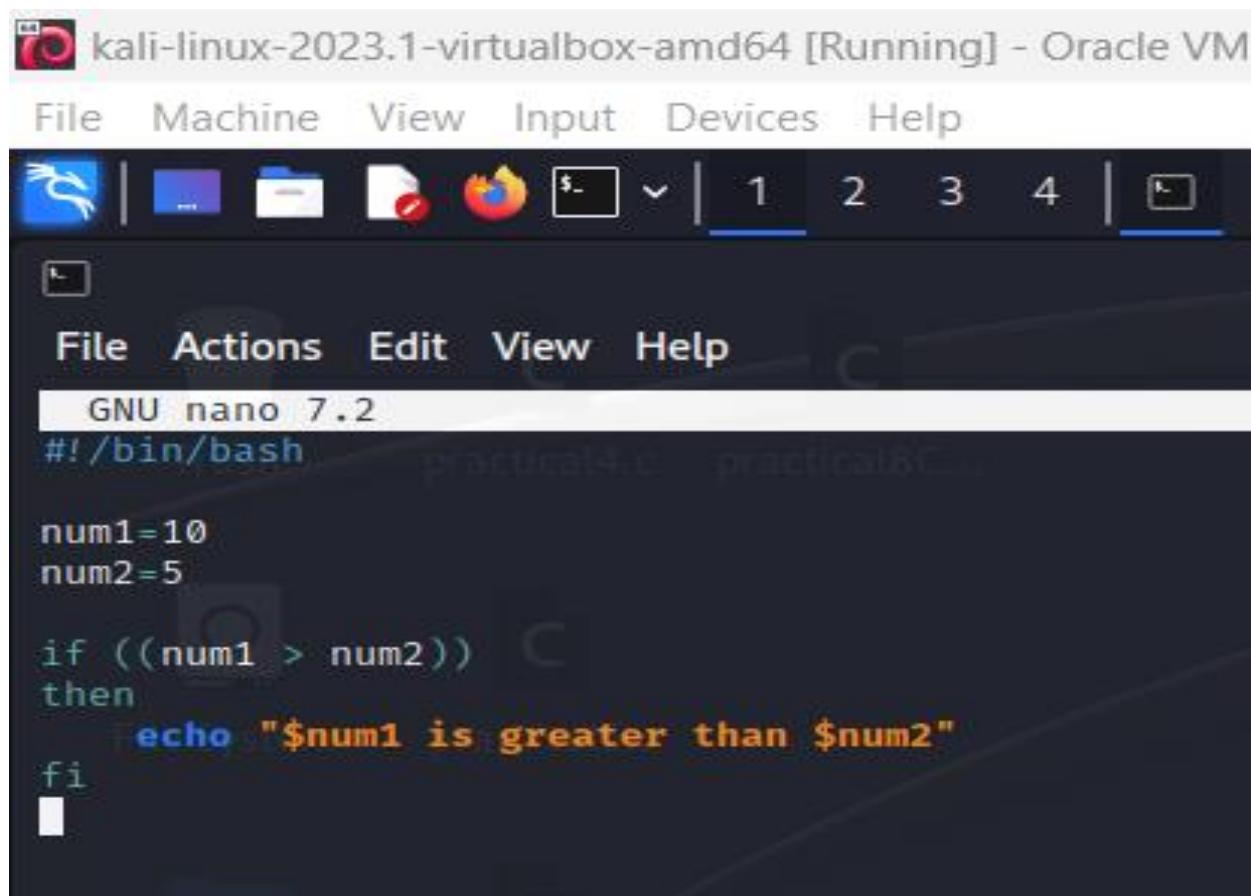
```
[root@samruddhi]# ./untilloop3.sh
Sum: 21
```

At the bottom of the terminal window, there are tabs for "practical8C...", "practical9A...", and "practical6F...".

arithmetic comparison:

1) > (Greater Than) Operator

Example

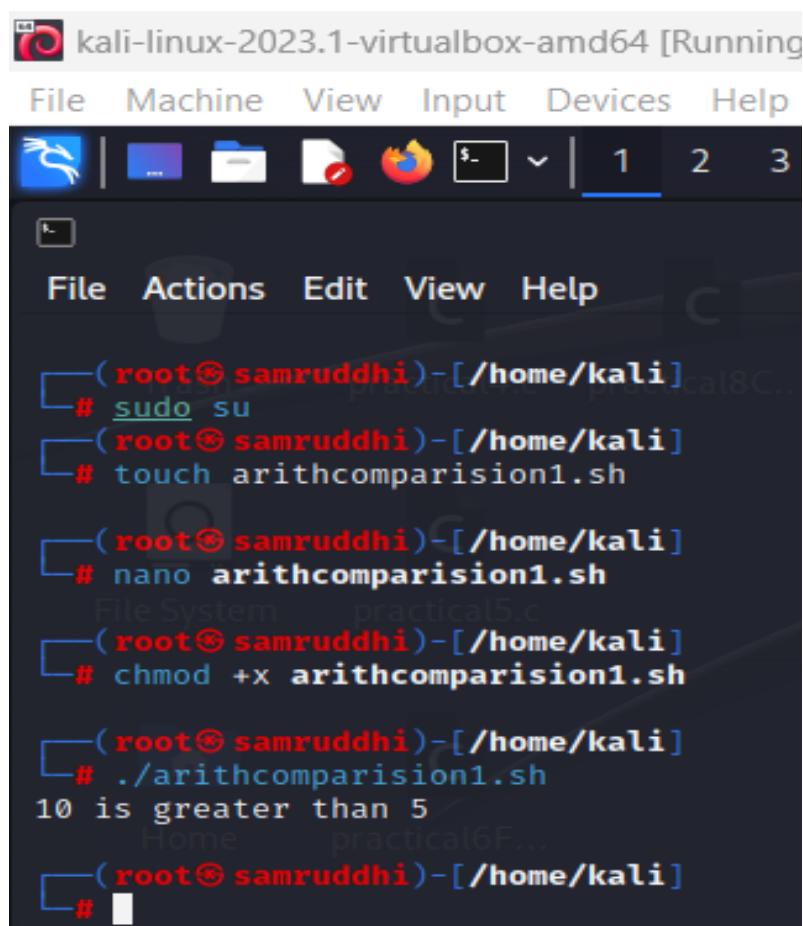


A screenshot of a Kali Linux desktop environment within Oracle VM VirtualBox. The terminal window title is "kali-linux-2023.1-virtualbox-amd64 [Running] - Oracle VM". The terminal shows a nano editor session with the following script content:

```
GNU nano 7.2
#!/bin/bash

num1=10
num2=5

if ((num1 > num2))
then
    echo "$num1 is greater than $num2"
fi
```



A screenshot of a Kali Linux desktop environment within Oracle VM VirtualBox. The terminal window title is "kali-linux-2023.1-virtualbox-amd64 [Running]". The terminal shows a root shell session with the following command history and output:

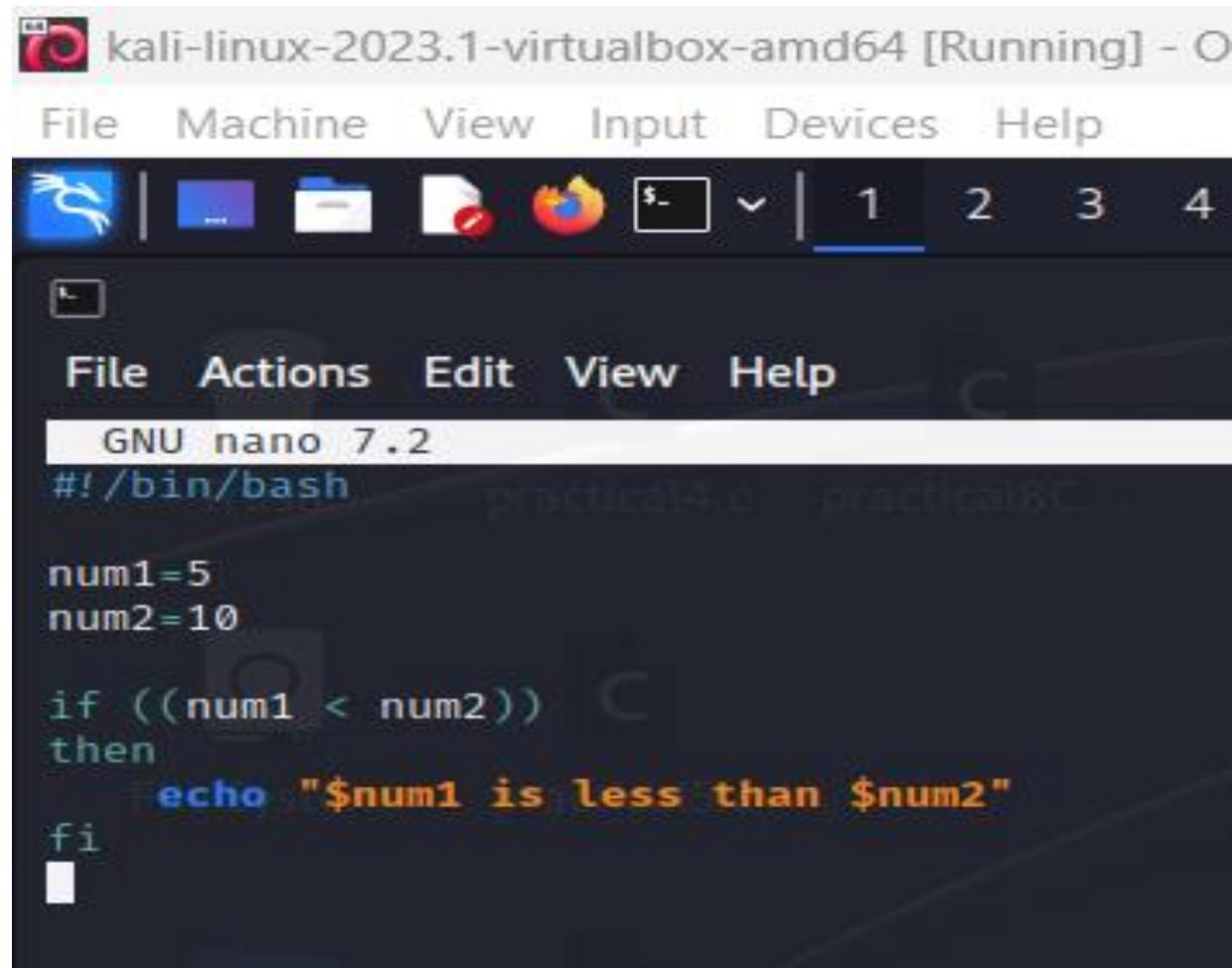
```
(root@samruddhi)-[~/home/kali]
# sudo su
(root@samruddhi)-[~/home/kali]
# touch arithcomparision1.sh

(root@samruddhi)-[~/home/kali]
# nano arithcomparision1.sh
File System practical5.c
(root@samruddhi)-[~/home/kali]
# chmod +x arithcomparision1.sh

(root@samruddhi)-[~/home/kali]
# ./arithcomparision1.sh
10 is greater than 5
Home practical6F...
(root@samruddhi)-[~/home/kali]
#
```

2) < (Less Than) Operator

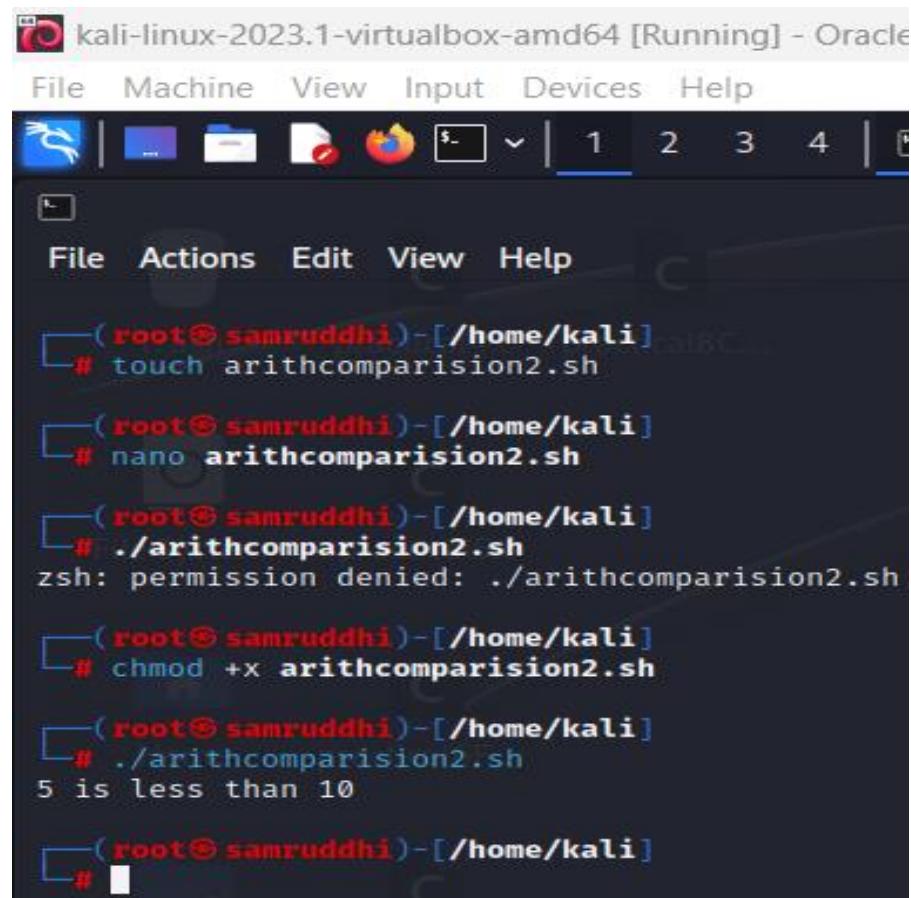
Example



```
File Machine View Input Devices Help
File Actions Edit View Help
GNU nano 7.2
#!/bin/bash

num1=5
num2=10

if ((num1 < num2))
then
    echo "$num1 is less than $num2"
fi
```



```
(root@samruddhi)-[~/kali]
# touch arithcomparision2.sh

(root@samruddhi)-[~/kali]
# nano arithcomparision2.sh

(root@samruddhi)-[~/kali]
# ./arithcomparision2.sh
zsh: permission denied: ./arithcomparision2.sh

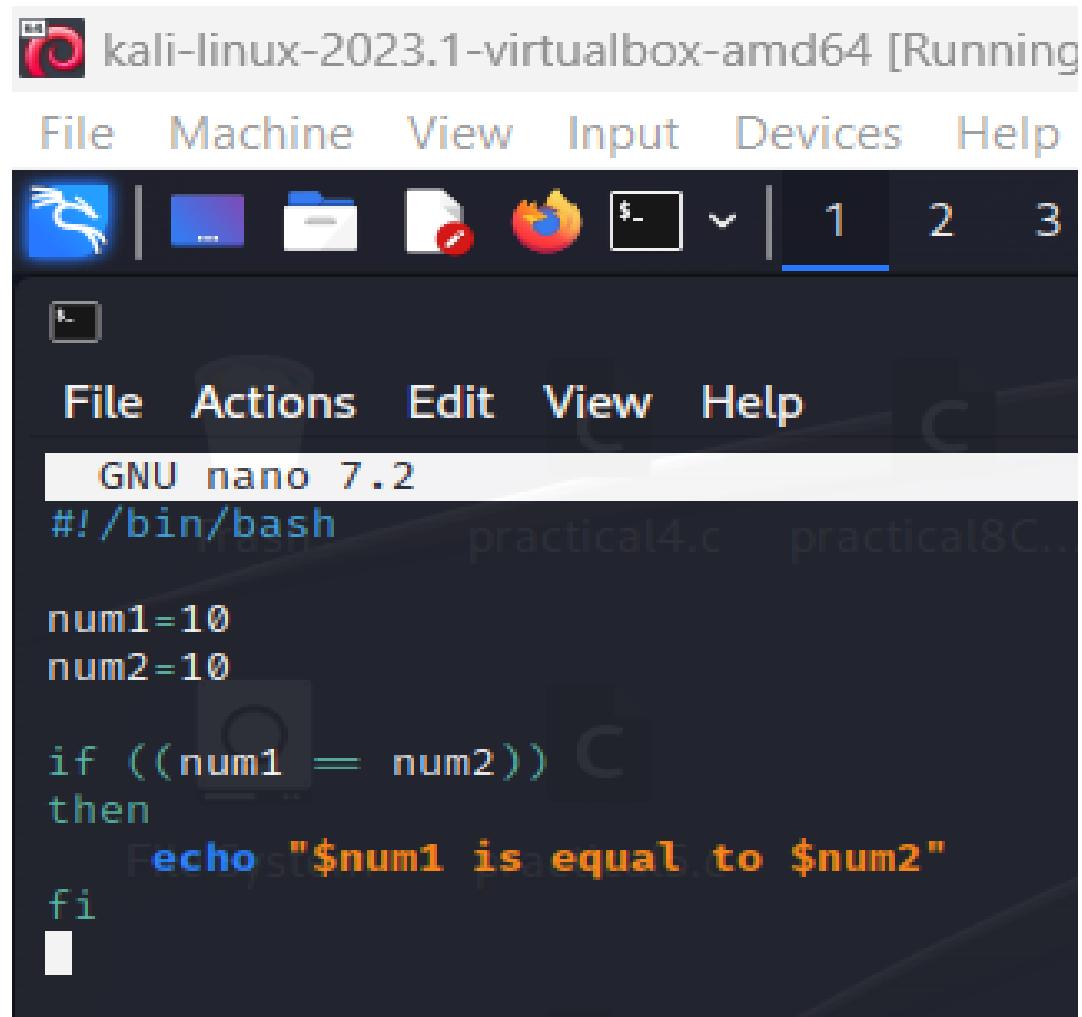
(root@samruddhi)-[~/kali]
# chmod +x arithcomparision2.sh

(root@samruddhi)-[~/kali]
# ./arithcomparision2.sh
5 is less than 10

(root@samruddhi)-[~/kali]
```

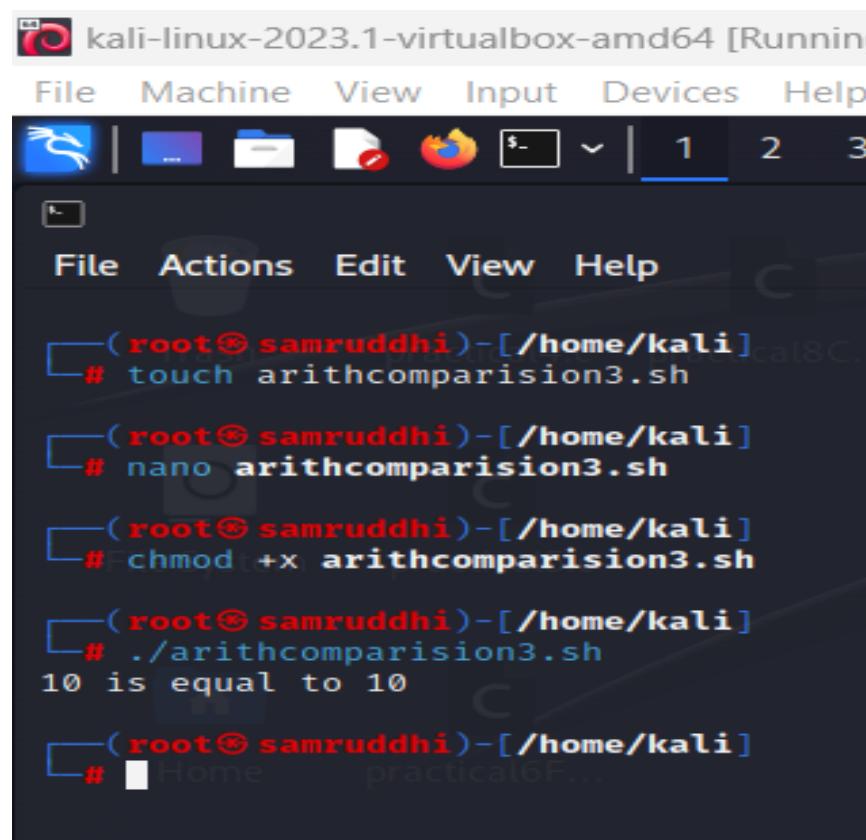
3) **==** (Equal To Operator)

Example



```
File Machine View Input Devices Help
File Actions Edit View Help
GNU nano 7.2
#!/bin/bash
practical4.c practical8C...
num1=10
num2=10

if ((num1 == num2))
then
    echo "$num1 is equal to $num2"
fi
```



```
(root@samruddhi)-[~/home/kali]
# touch arithcomparision3.sh

(root@samruddhi)-[~/home/kali]
# nano arithcomparision3.sh

(root@samruddhi)-[~/home/kali]
# chmod +x arithcomparision3.sh

(root@samruddhi)-[~/home/kali]
# ./arithcomparision3.sh
10 is equal to 10

(root@samruddhi)-[~/home/kali]
# Home practical6F...
```

4) != (Not Equal To) Operator

Example

```
GNU nano 7.2
#!/bin/bash
practical4.c practical8C...
num1=5
num2=10

if ((num1 != num2))
then
    echo "$num1 is not equal to $num2"
fi
```

```
(root@samruddhi)-[~/home/kali]
# touch arithcomparision4.sh

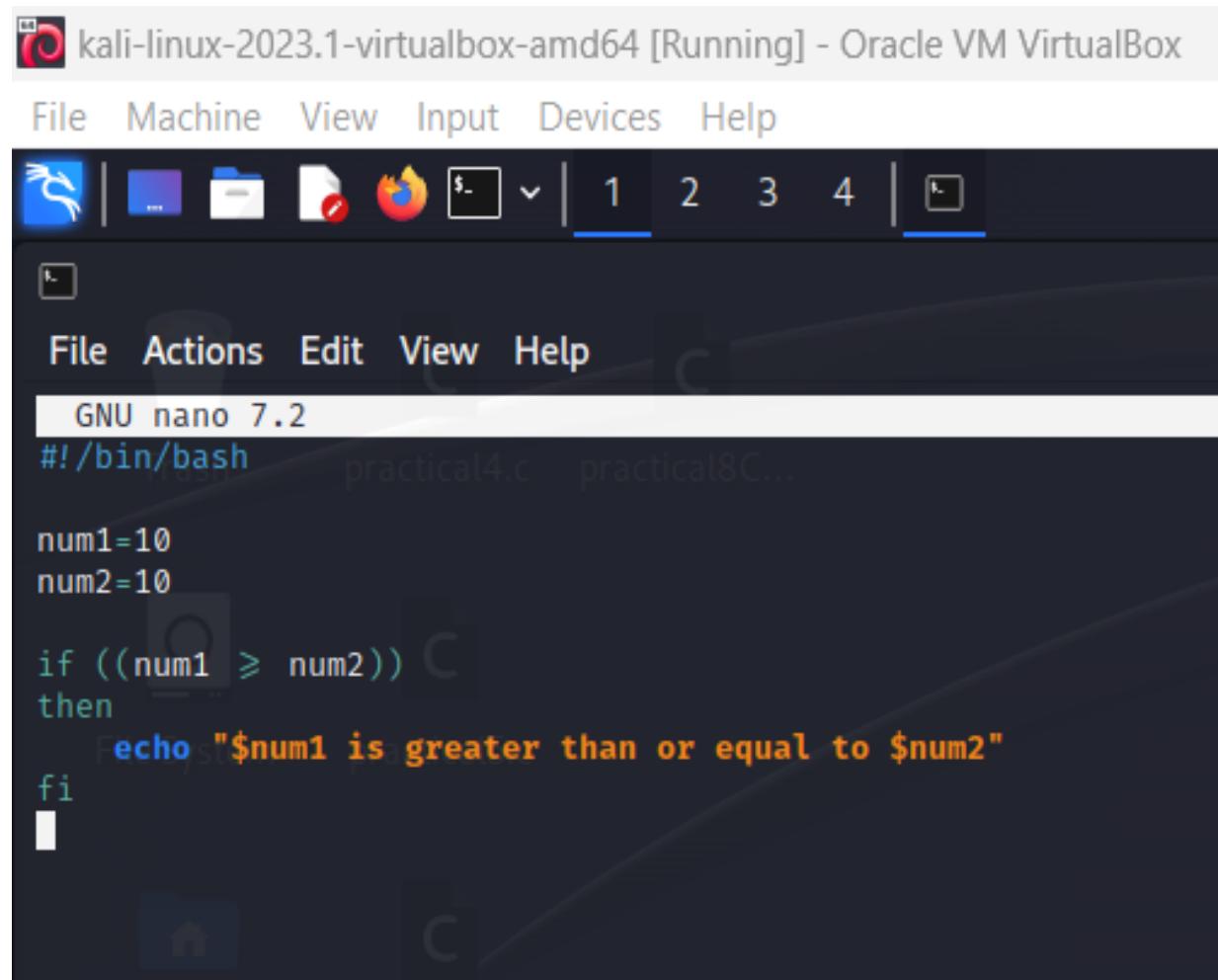
(root@samruddhi)-[~/home/kali]
# nano arithcomparision4.sh

(root@samruddhi)-[~/home/kali]
# chmod +x arithcomparision4.sh

(root@samruddhi)-[~/home/kali]
# ./arithcomparision4.sh
5 is not equal to 10
```

5) >= (Greater Than Equal) Operator

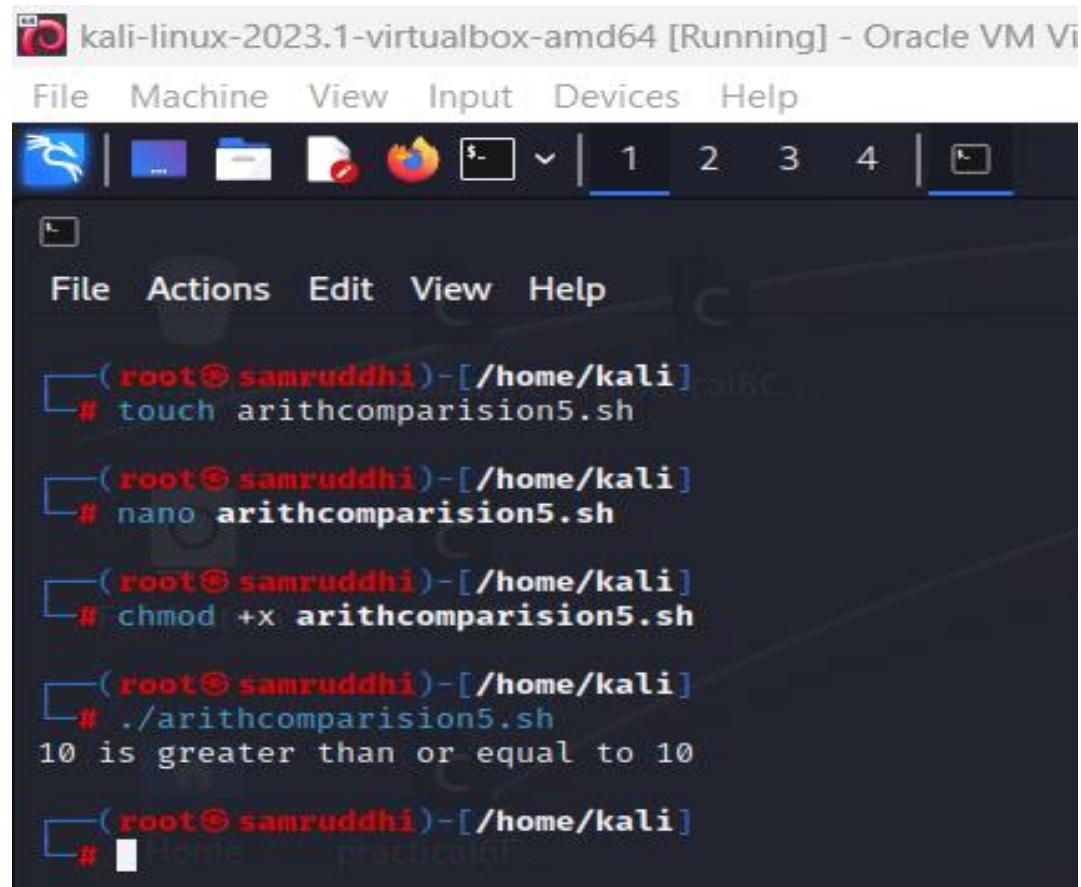
Example



A screenshot of a Kali Linux desktop environment in Oracle VM VirtualBox. The terminal window shows the following content:

```
GNU nano 7.2
#!/bin/bash      practical4.c  practical8C...
num1=10
num2=10

if ((num1 >= num2))
then
    echo "$num1 is greater than or equal to $num2"
fi
```

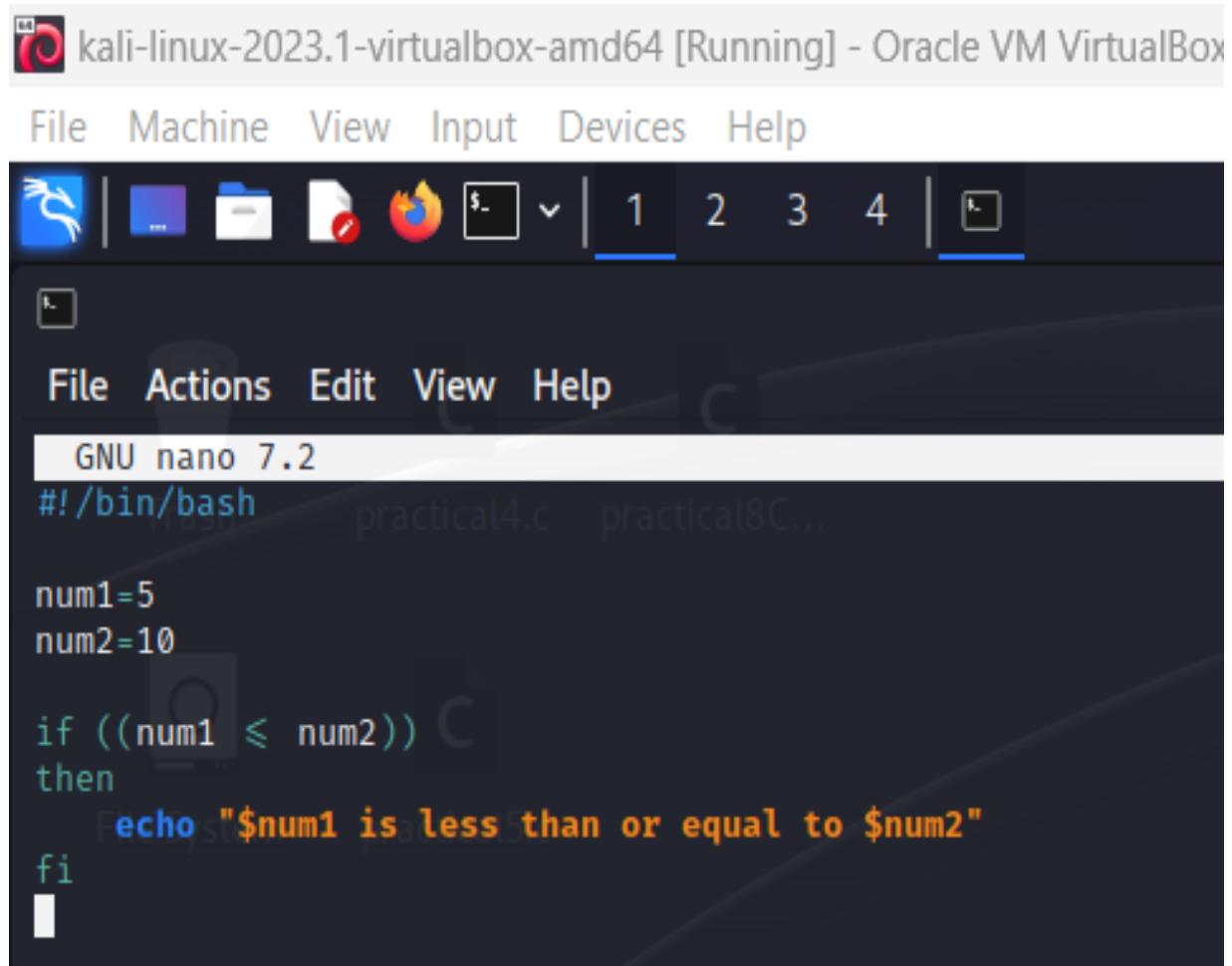


A screenshot of a Kali Linux desktop environment in Oracle VM VirtualBox. The terminal window shows the following root-level commands and their output:

```
[root@samruddhi]# touch arithcomparision5.sh
[root@samruddhi]# nano arithcomparision5.sh
[root@samruddhi]# chmod +x arithcomparision5.sh
[root@samruddhi]# ./arithcomparision5.sh
10 is greater than or equal to 10
[root@samruddhi]#
```

6) (Less Than Equal) Operator

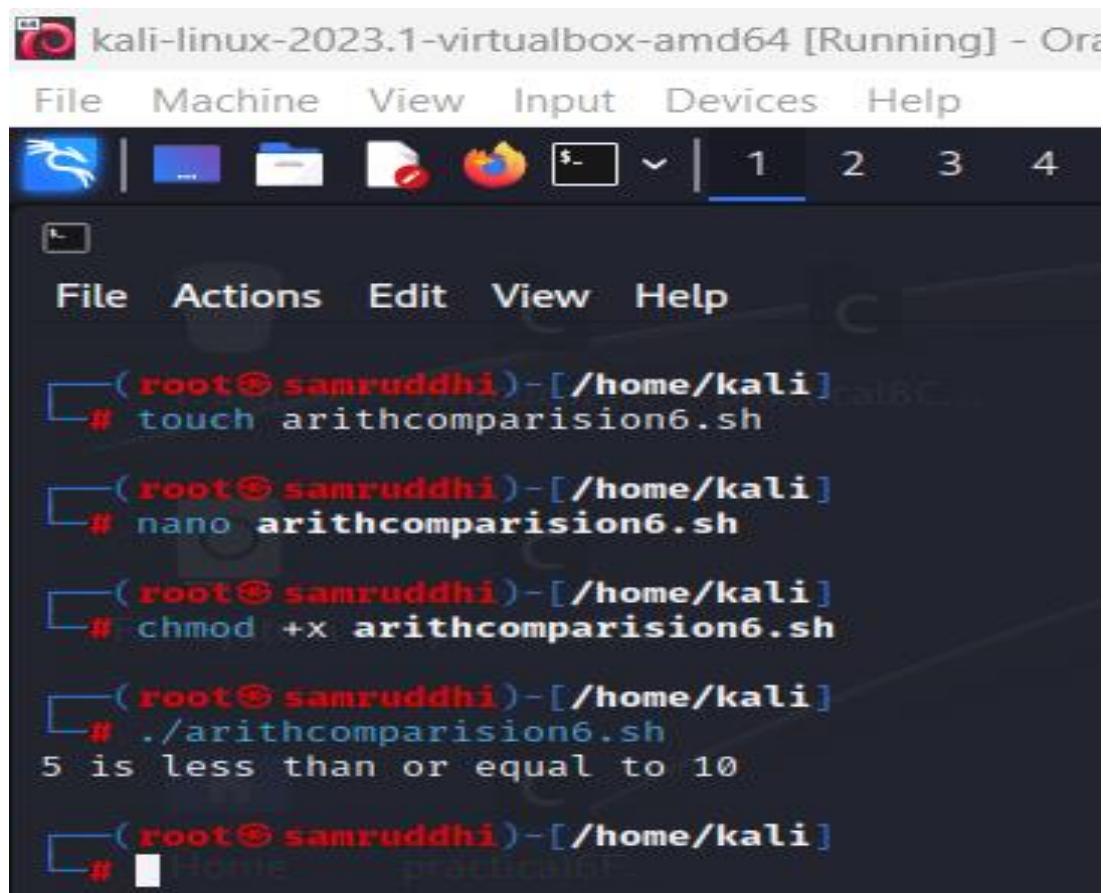
Example



The screenshot shows a Kali Linux 2023.1 virtual machine running in Oracle VM VirtualBox. A terminal window is open, showing the command-line interface. The terminal title is "File Machine View Input Devices Help". Below the title bar is a menu bar with "File Actions Edit View Help". The main area of the terminal shows the following content:

```
GNU nano 7.2
#!/bin/bash
practical4.c practical8C...
num1=5
num2=10

if ((num1 <= num2))
then
    echo "$num1 is less than or equal to $num2"
fi
```

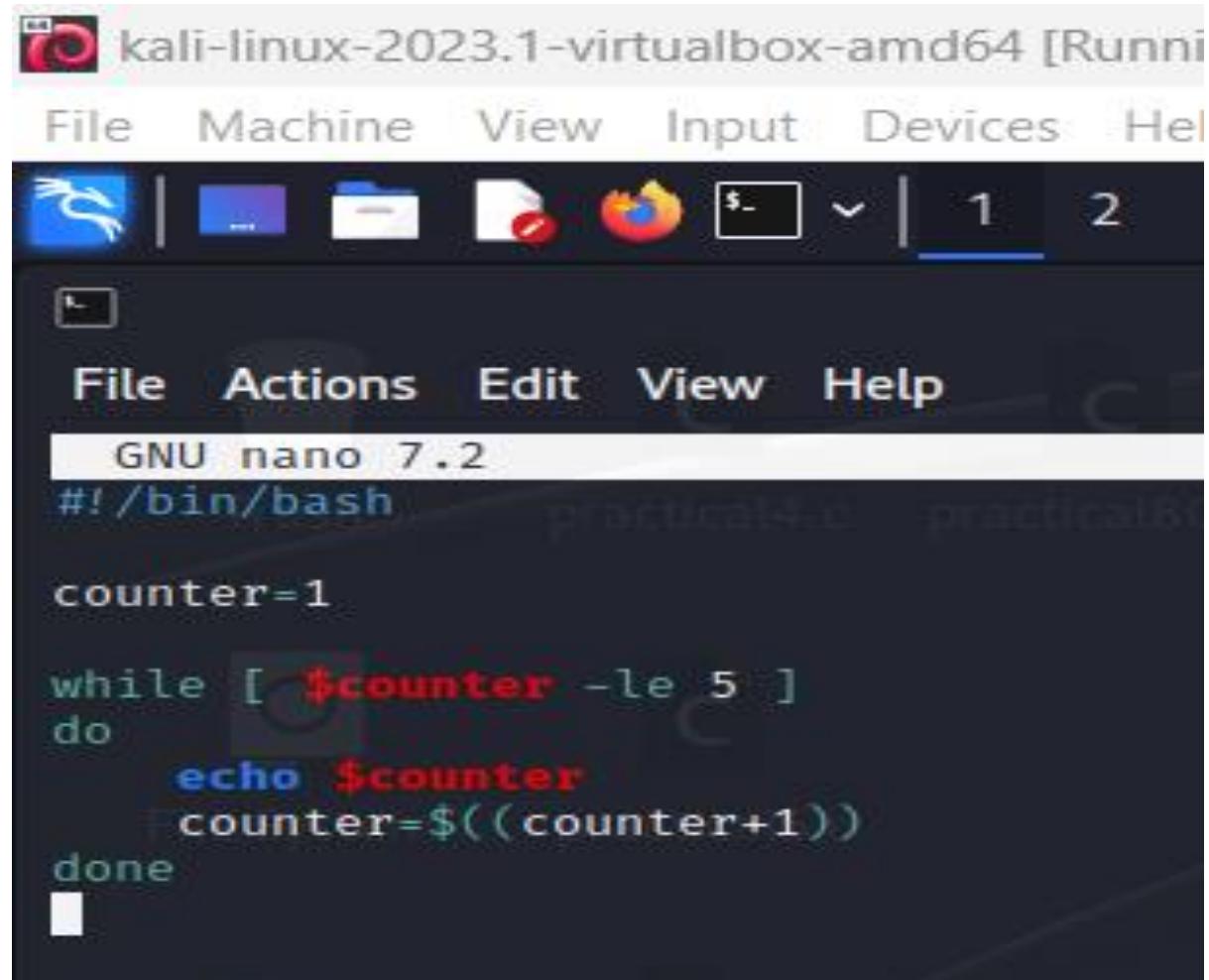


The screenshot shows a Kali Linux 2023.1 virtual machine running in Oracle VM VirtualBox. A terminal window is open, showing a root shell. The terminal title is "File Machine View Input Devices Help". Below the title bar is a menu bar with "File Actions Edit View Help". The main area of the terminal shows the following content:

```
[root@samruddhi]# touch arithcomparision6.sh
[root@samruddhi]# nano arithcomparision6.sh
[root@samruddhi]# chmod +x arithcomparision6.sh
[root@samruddhi]# ./arithcomparision6.sh
5 is less than or equal to 10
[root@samruddhi]#
```

conditional loops:

Example 1

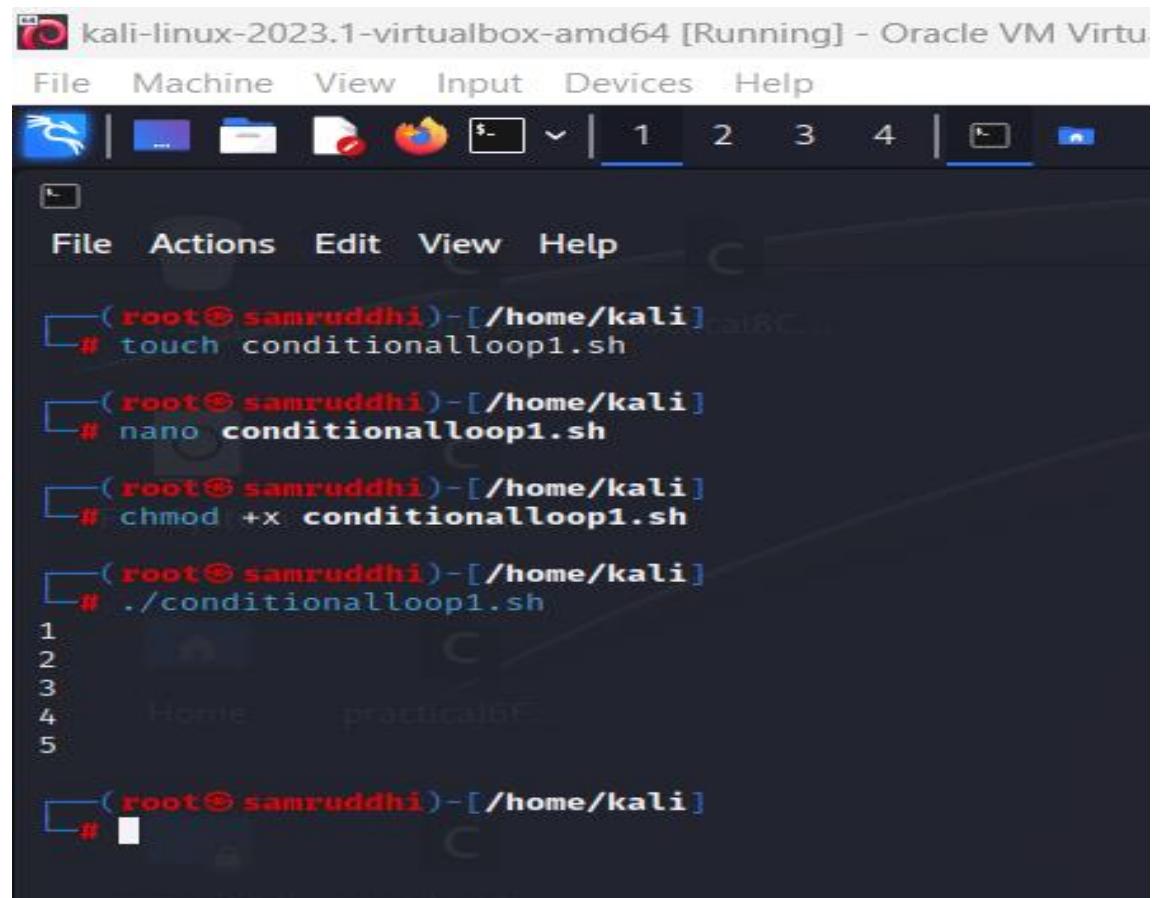


A screenshot of a Kali Linux terminal window titled "kali-linux-2023.1-virtualbox-amd64 [Running]". The window shows a menu bar with File, Machine, View, Input, Devices, Help, and a toolbar with icons for terminal, file manager, browser, and others. A nano editor window is open with the following script content:

```
GNU nano 7.2
#!/bin/bash

counter=1

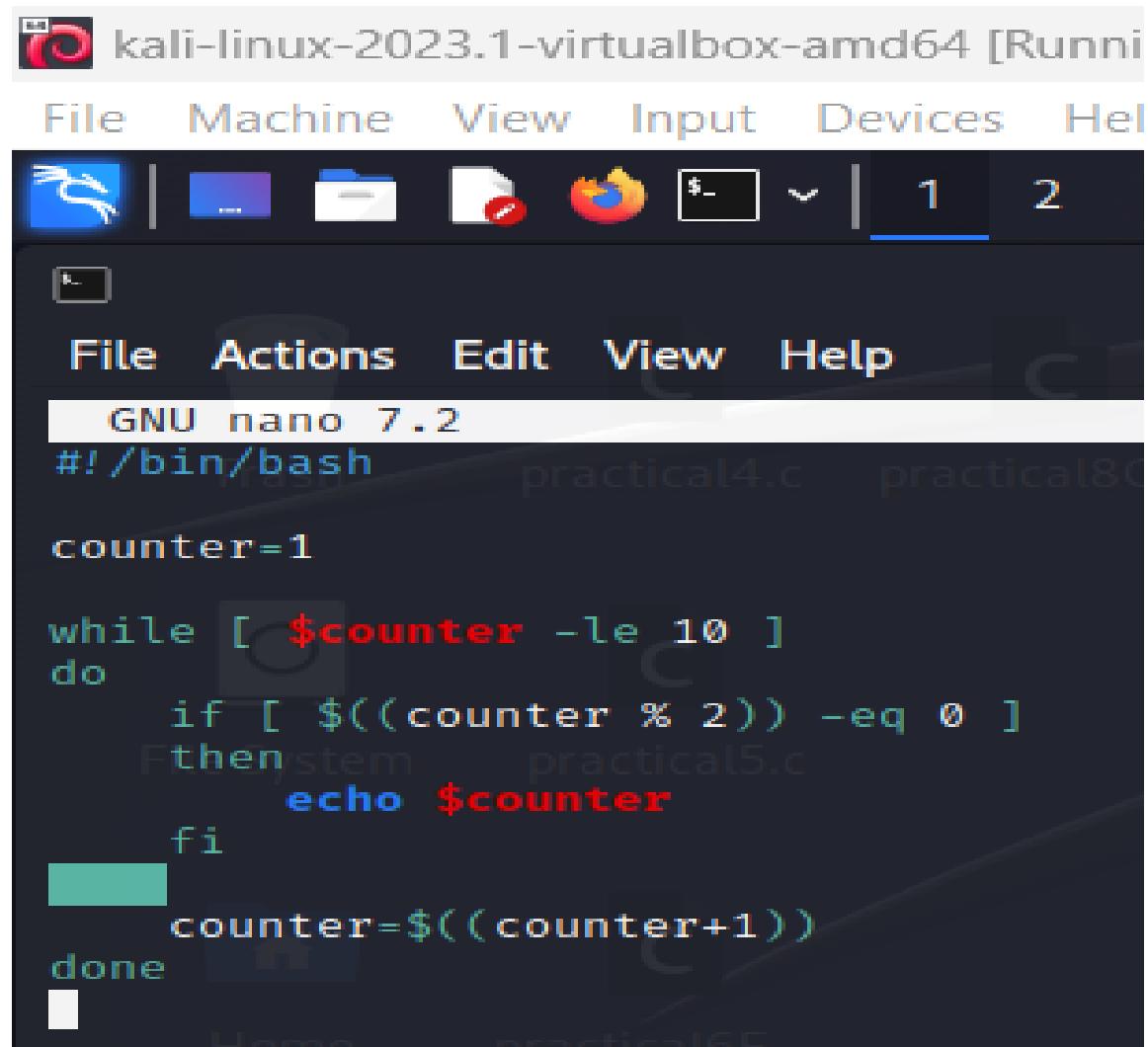
while [ $counter -le 5 ]
do
    echo $counter
    counter=$((counter+1))
done
```



A screenshot of a Kali Linux terminal window titled "kali-linux-2023.1-virtualbox-amd64 [Running] - Oracle VM Virtu". The window shows a menu bar with File, Machine, View, Input, Devices, Help, and a toolbar with icons for terminal, file manager, browser, and others. The terminal session shows the following steps:

```
[root@samruddhi ~]# touch conditionalloop1.sh
[root@samruddhi ~]# nano conditionalloop1.sh
[root@samruddhi ~]# chmod +x conditionalloop1.sh
[root@samruddhi ~]# ./conditionalloop1.sh
1
2
3
4
5
[root@samruddhi ~]
```

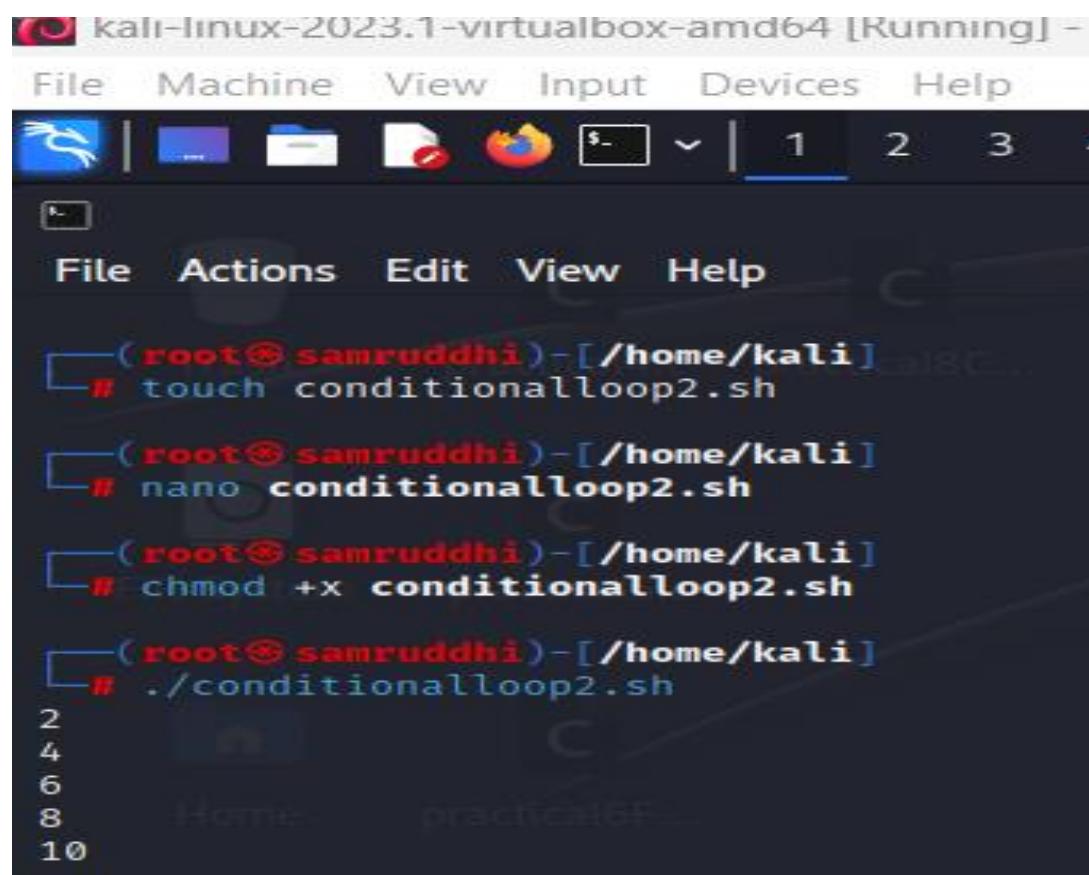
Example 2



A screenshot of a Kali Linux terminal window titled "kali-linux-2023.1-virtualbox-amd64 [Running]". The terminal shows a nano editor session with the following script content:

```
GNU nano 7.2
#!/bin/bash
counter=1

while [ $counter -le 10 ]
do
    if [ $((counter % 2)) -eq 0 ]
    Then
        echo $counter
    fi
    counter=$((counter+1))
done
```



A screenshot of a Kali Linux terminal window titled "kali-linux-2023.1-virtualbox-amd64 [Running]". The terminal shows the execution of the shell script "conditionalloop2.sh" with the following output:

```
[root@samruddhi]# touch conditionalloop2.sh
[root@samruddhi]# nano conditionalloop2.sh
[root@samruddhi]# chmod +x conditionalloop2.sh
[root@samruddhi]# ./conditionalloop2.sh
2
4
6
8
10
```

Example 3

The screenshot shows a Kali Linux terminal window titled "kali-linux-2023.1-virtualbox-amd64 [Running] - Oracle VM". The terminal interface includes a menu bar with File, Machine, View, Input, Devices, and Help. Below the menu is a toolbar with icons for Home, File Manager, Terminal, and a browser. The main window displays a terminal session where a script is being edited in nano. The script contains a while loop that reads user input and checks if it's greater than zero. If valid, it prints a message and breaks the loop; if invalid, it prints an error message and loops back.

```
GNU nano 7.2
#!/bin/bash
while true
do
    read -p "Enter a positive number: " number
    if [ $number -gt 0 ]
    then
        echo "Valid input. Exiting loop."
        break
    else
        echo "Invalid input. Please try again."
    fi
done
```

The screenshot shows a Kali Linux terminal window titled "kali-linux-2023.1-virtualbox-amd64 [Running] - Oracle VM". The terminal interface is similar to the previous one. This time, the terminal session shows the execution of the script. It starts by creating a file named "conditionalloop3.sh", then opens it in nano. After saving and exiting nano, the script is made executable with chmod +x. Finally, the script is run with ./conditionalloop3.sh. The user enters -8, which is invalid, so the script asks for a valid input. The user then enters 7, which is valid, and the script exits the loop and prints a success message.

```
[root@samruddhi ~]# touch conditionalloop3.sh
[root@samruddhi ~]# nano conditionalloop3.sh
[root@samruddhi ~]# chmod +x conditionalloop3.sh
[root@samruddhi ~]# ./conditionalloop3.sh
Enter a positive number: -8
Invalid input. Please try again.
Enter a positive number: 7
Valid input. Exiting loop.
```

grep command:

```
Activities Terminal May 23 00:26 samruddhi@samruddhi-VirtualBox:~$ cat students.txt Samruddhi Saniya Soham Akanksha Aditi Akanksha samruddhi@samruddhi-VirtualBox:~$ grep -l "Samruddhi" students samruddhi@samruddhi-VirtualBox:~$ grep -n "Samruddhi" students.txt 1:Samruddhi samruddhi@samruddhi-VirtualBox:~$ grep -l "Samruddhi" students.txt students.txt samruddhi@samruddhi-VirtualBox:~$ grep -c "Akanksha" students.txt 2 samruddhi@samruddhi-VirtualBox:~$ grep "Soham" students.txt Soham samruddhi@samruddhi-VirtualBox:~$ grep S students.txt Samruddhi Saniya Soham samruddhi@samruddhi-VirtualBox:~$ grep Sa students.txt Samruddhi Saniya samruddhi@samruddhi-VirtualBox:~$ grep i students.txt Samruddhi Saniya Aditi samruddhi@samruddhi-VirtualBox:~$
```

sed command:

```
Activities Terminal May 23 10:47 samruddhi@samruddhi-VirtualBox:~$ cat studentdata.txt Samruddhi 18 Pune IT Aditi 26 Usmanabad Com Soham 35 Baramati AIML samruddhi@samruddhi-VirtualBox:~$ sed 's/Pawar/Soham' studentdata.txt sed: -e expression #1, char 13: unterminated `s' command samruddhi@samruddhi-VirtualBox:~$ sed '2p' studentdata.txt Samruddhi 18 Pune IT Aditi 26 Usmanabad Com Aditi 26 Usmanabad Com Soham 35 Baramati AIML samruddhi@samruddhi-VirtualBox:~$ sed -n '2p' studentdata.txt Aditi 26 Usmanabad Com samruddhi@samruddhi-VirtualBox:~$ sed -n '$p' studentdata.txt Soham 35 Baramati AIML samruddhi@samruddhi-VirtualBox:~$ sed -n '1,2p' studentdata.txt Samruddhi 18 Pune IT Aditi 26 Usmanabad Com samruddhi@samruddhi-VirtualBox:~$ sed -n '1,2!p' studentdata.txt Soham 35 Baramati AIML samruddhi@samruddhi-VirtualBox:~$ sed 's/[Ss]oham/Pawar/' studentdata.txt Samruddhi 18 Pune IT Aditi 26 Usmanabad Com Pawar 35 Baramati AIML samruddhi@samruddhi-VirtualBox:~$ sed -e 's/Com/Computer/g' -e 's/IT/Info Technology/g' studentdata.txt Samruddhi 18 Pune Info Technology Aditi 26 Usmanabad Computer Soham 35 Baramati AIML
```

PRACTICAL NO. 2

Process control system calls: The demonstration of FORK, EXECVE and WAIT system calls along with zombie and orphan states.

a. Implement the C program in which main program accepts the integers to be sorted. Main program uses the FORK system call to create a new process called a child process. Parent process sorts the integers using sorting algorithm and waits for child process using WAIT system call to sort the integers using any sorting algorithm. Also demonstrate zombie and orphan states.

CODE:

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/wait.h>

#define MAX 30

void merge_sort(int[],int,int);
void merge(int[],int,int,int);

int main()
{
    pid_t pid;
    int a[10],b[10],n,i;
    printf("Enter No of Elements You want to enter:");
    scanf("%d",&n);
    printf("Enter Elements:");
    for(i=0;i<n;i++)
    {
        printf("\n Enter Number %d:",i+1);
        scanf("%d",&a[i]);
        b[i]=a[i];
    }
    merge_sort(a,0,n-1);
```

```
printf("\n sorted data using merge sort in parent:");
for(i=0;i<n;i++)
{
    printf("\t%d",a[i]);
}
}

void merge_sort(int a[],int i,int j)
{
    int k;
    if(i<j)
    {
        k=(i+j)/2;
        merge_sort(a,i,k);
        merge_sort(a,k+1,j);
        merge(a,i,k,j);
    }
}

void merge(int a[],int l,int m,int u)
{
    int c[MAX];
    int i,j,k;
    i=l;
    j=m+1;
    k=0;
    while(i<=m && j<=u)
    {
        if(a[i]<a[j])
        {
            c[k]=a[i];
            k++;
            i++;
        }
    }
}
```

```
    }
else
{
    c[k]=a[j];
    k++;
    j++;
}
while(i<=m)
{
    c[k]=a[i];
    i++;
    k++;
}
while(j<=u)
{
    c[k]=a[j];
    k++;
    j++;
}
for(i=l,j=0;i<=u;i++,j++)
    a[i]=c[j];
}
```

OUTPUT:

The screenshot shows a terminal window titled "kali-linux-2023.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox". The terminal interface includes a menu bar with File, Machine, View, Input, Devices, and Help, and a toolbar with icons for file operations. The main window displays a command-line session:

```
(kali㉿samruddhi)~$ sudo su
[sudo] password for kali:
(root㉿samruddhi) [/home/kali]
# cd Desktop

(root㉿samruddhi) [/home/kali/Desktop]
# gcc practical2.c
[root@samruddhi ~]# ./a.out
Enter No of Elements You want to enter:5
Enter Elements:
Enter Number 1:7
Enter Number 2:5
Enter Number 3:9
Enter Number 4:2
Enter Number 5:0
sorted data using merge sort in parent: 0 2 5 7 9
[root@samruddhi ~]#
```

PRACTICAL NO. 3

Implement the C program for CPU Scheduling Algorithms: Shortest Job First (Pre-emptive) and Round Robin with different arrival time.

SHORTEST JOB FIRST (PRE-EMPTIVE)

CODE:

```
#include <stdio.h>

int main()
{
    int a[10],b[10],x[10],i,j,smallest,count=0,time,n;

    double avg=0,tt=0,end;

    printf("Enter the number of Processes:\n");
    scanf("%d",&n);

    printf("Enter arrival time\n");
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);

    printf("Enter burst time\n");
    for(i=0;i<n;i++)
        scanf("%d",&b[i]);
    for(i=0;i<n;i++)
        x[i]=b[i];
    b[9]=9999;
    for(time=0;count!=n;time++)
    {
        smallest=9;
        for(i=0;i<n;i++)
        {
            if(a[i]<=time && b[i]<b[smallest] && b[i]>0 )
                smallest=i;
        }
        b[smallest]--;
        count++;
        tt+=b[smallest];
        if(count==n)
            end=tt;
    }
    avg=tt/n;
    printf("Average waiting time = %f",avg);
    printf("Turnaround time = %f",tt);
    printf("Completion time = %f",end);
}
```

```
if(b[smallest]==0)
{
    count++;
    end=time+1;
    avg=avg+end-a[smallest]-x[smallest];
    tt= tt+end-a[smallest];
}
printf("\n\nAverage waiting time = %lf\n",avg/n);
printf("Average Turnaround time = %lf",tt/n);
return 0;
}
```

OUTPUT:

```
kali-linux-2023.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
File Actions Edit View Help
[(kali㉿samruddhi)-[~]
$ sudo su
[sudo] password for kali:
[(root㉿samruddhi)-[/home/kali]
# cd Desktop

[(root㉿samruddhi)-[/home/kali/Desktop]
# gcc practical3SJF.c
File System practical5.c
[(root㉿samruddhi)-[/home/kali/Desktop]
# ./a.out
Enter the number of Processes:
4
Enter arrival time
5
6 Home practical6F...
7
8
Enter burst time
4
5
8
9 samruddhi practical6L...

Average waiting time = 6.000000
Average Turnaround time = 12.500000

[(root㉿samruddhi)-[/home/kali/Desktop]
# █
```

ROUND ROBIN

CODE:

```
#include <stdio.h>

int main()
{
    int cnt, j, n, t, remain, flag = 0, tq;
    float wt = 0, tat = 0, at[10], bt[10], rt[10];
    printf("Enter Total Process:\t");
    scanf("%d", &n);
    remain = n;
    for (cnt = 0; cnt < n; cnt++)
    {
        printf("\nEnter Arrival Time and Burst Time for Process Number %d: ", cnt + 1);
        scanf("%f", &at[cnt]);
        scanf("%f", &bt[cnt]);
        rt[cnt] = bt[cnt];
    }
    printf("Enter Time Quantum:\t");
    scanf("%d", &tq);
    printf("\n\nProcess\tTurnaround Time|Waiting Time\n\n");
    for (t = 0, cnt = 0; remain != 0;)
    {
        if (rt[cnt] <= tq && rt[cnt] > 0)
        {
            t += rt[cnt];
            rt[cnt] = 0;
            flag = 1;
        }
        else if (rt[cnt] > 0)
        {
            remain--;
        }
    }
}
```

```
rt[cnt] -= tq;
t += tq;
}

if(rt[cnt] == 0 && flag == 1)
{
    remain--;
    printf("P[%d]\t%d\t%d\n", cnt + 1, t - at[cnt], t - at[cnt] - bt[cnt]);
    wt += t - at[cnt] - bt[cnt];
    tat += t - at[cnt];
    flag = 0;
}
if(cnt == n - 1)
    cnt = 0;
else if(at[cnt + 1] <= t)
    cnt++;
else
    cnt = 0;
}
printf("\nAverage Waiting Time= %.2f\n", wt / n);
printf("Average Turnaround Time = %.2f", tat / n);
return 0;
}
```

OUTPUT:

```
kali-linux-2023.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
File Actions Edit View Help
(kali㉿samruddhi)~
$ sudo su
[sudo] password for kali:
(root㉿samruddhi) [/home/kali]
# cd Desktop

(root㉿samruddhi) [/home/kali/Desktop]
# gcc practical3RR.c
[root㉿samruddhi) [/home/kali/Desktop]
# ./a.out
Enter Total Process:      5

Enter Arrival Time and Burst Time for Process Number 1: 1 5

Enter Arrival Time and Burst Time for Process Number 2: 7 2

Enter Arrival Time and Burst Time for Process Number 3: 9 4

Enter Arrival Time and Burst Time for Process Number 4: 6 0

Enter Arrival Time and Burst Time for Process Number 5: 3 1
Enter Time Quantum:        4

Process | Turnaround Time | Waiting Time
P[1]    |           1          |       1137906064
a.out
```

PRACTICAL NO. 4

a. Thread synchronization using counting semaphores. Application to demonstrate: producer-consumer problem with counting semaphores and mutex.

CODE:

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <semaphore.h>
#include <unistd.h>

sem_t mutex, full, empty;
int *buffer, front = 0, rear = 0, count = 1;
int sizeOfBuffer;
int stopProducing = 0;
int stopConsuming = 0;

void *producer(void *p);
void *consumer(void *p);

int main(int argc, char *argv[]) {
    int choice;
    int gProduce = 3;
    int gConsume = 3;
    int i;

    printf("Enter Size of Buffer:\n");
    scanf("%d", &sizeOfBuffer);

    buffer = (int *)malloc(sizeOfBuffer * sizeof(int));
```

```
sem_init(&mutex, 0, 1);
sem_init(&full, 0, 0);
sem_init(&empty, 0, sizeOfBuffer);

pthread_t tid_p[gProduce];
pthread_t tid_c[gConsume];

while (1) {
    for (i = 0; i < gProduce; i++) {
        pthread_create(&tid_p[i], NULL, producer, (void *)(intptr_t)i);
    }

    for (i = 0; i < gConsume; i++) {
        pthread_create(&tid_c[i], NULL, consumer, (void *)(intptr_t)i);
    }

    printf("\nDo you want to continue? (1/0):\n");
    scanf("%d", &choice);

    if (choice == 0)
        break;
}

stopProducing = 1;
stopConsuming = 1;

for (i = 0; i < gProduce; i++) {
    pthread_join(tid_p[i], NULL);
}

for (i = 0; i < gConsume; i++) {
```

```
pthread_join(tid_c[i], NULL);
}

free(buffer);
sem_destroy(&mutex);
sem_destroy(&full);
sem_destroy(&empty);

return 0;
}

void *producer(void *p) {
    int t = (int)(intptr_t)p;

    while (!stopProducing) {
        sem_wait(&empty);
        sem_wait(&mutex);

        buffer[rear] = count;
        printf("\nProducer %d produced element: %d\n", t, buffer[rear]);
        rear = (rear + 1) % sizeOfBuffer;
        count++;

        sem_post(&mutex);
        sem_post(&full);

        printf("\n*****\n");
        for (int i = 0; i < sizeOfBuffer; i++)
            printf("\t%d", buffer[i]);
        printf("\n*****\n");
    }
}
```

```
    sleep(1);

}

pthread_exit(0);
}

void *consumer(void *p) {
    int t = (int)(intptr_t)p;

    while (!stopConsuming) {
        sem_wait(&full);
        sem_wait(&mutex);

        printf("\nConsumer %d consumed element: %d\n", t, buffer[front]);
        buffer[front] = 0;
        front = (front + 1) % sizeOfBuffer;

        sem_post(&mutex);
        sem_post(&empty);

        printf("\n*****\n");
        for (int i = 0; i < sizeOfBuffer; i++)
            printf("\t%d", buffer[i]);
        printf("\n*****\n");

        sleep(1);
    }

    pthread_exit(0);
}
```

OUTPUT:

```
kali-linux-2023.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
File Actions Edit View Help
(kali㉿samruddhi) [~]
$ sudo su
[sudo] password for kali:
(root㉿samruddhi) [/home/kali]
# cd Desktop

(root㉿samruddhi) [/home/kali/Desktop]
# gcc practical4.c
[root@samruddhi ~]# ./a.out
Enter Size of Buffer:
5

Do you want to continue? (1/0):
Producer 2 produced element: 1
*****
1 0 0 0
*****
Consumer 2 consumed element: 1
*****
0 2 0 0
*****
Producer 0 produced element: 2
*****
0 2 0 0
*****
Consumer 1 consumed element: 2
*****
0 0 0 0
*****
Producer 1 produced element: 3
*****
0 0 3 0
*****
Consumer 0 consumed element: 3
*****
0 0 0 0
*****
```



News &
more



Search

PRACTICAL NO. 5

Implement the C program for Deadlock Avoidance Algorithm: Bankers Algorithm.

BANKERS ALGORITHM

CODE:

```
#include <stdio.h>

#define MAX_PROCESSES 10
#define MAX_RESOURCES 10

int main()
{
    int num_processes, num_resources;
    int available[MAX_RESOURCES];
    int max_claim[MAX_PROCESSES][MAX_RESOURCES];
    int allocation[MAX_PROCESSES][MAX_RESOURCES];
    int need[MAX_PROCESSES][MAX_RESOURCES];
    int work[MAX_RESOURCES];
    int finish[MAX_PROCESSES];
    printf("Enter the number of processes: ");
    scanf("%d", &num_processes);
    printf("Enter the number of resources: ");
    scanf("%d", &num_resources);
    printf("Enter the available resources: ");
    for (int i = 0; i < num_resources; i++)
    {
        scanf("%d", &available[i]);
    }
    printf("Enter the maximum claim of each process: ");
    for (int i = 0; i < num_processes; i++)
    {
        printf("For process P%d: ", i);
        for (int j = 0; j < num_resources; j++)
        {
            scanf("%d", &max_claim[i][j]);
        }
    }
    // Initialize allocation and need matrices
    for (int i = 0; i < num_processes; i++)
    {
        for (int j = 0; j < num_resources; j++)
        {
            allocation[i][j] = 0;
            need[i][j] = max_claim[i][j];
        }
    }
    // Initialize work matrix
    for (int i = 0; i < num_resources; i++)
    {
        work[i] = available[i];
    }
    // Initialize finish matrix
    for (int i = 0; i < num_processes; i++)
    {
        finish[i] = 0;
    }
    // Main loop
    while (1)
    {
        // Check if all processes are finished
        int all_finished = 1;
        for (int i = 0; i < num_processes; i++)
        {
            if (finish[i] == 0)
            {
                all_finished = 0;
                break;
            }
        }
        if (all_finished)
        {
            break;
        }
        // Find a process that can be allocated
        int i;
        for (i = 0; i < num_processes; i++)
        {
            if (finish[i] == 0)
            {
                break;
            }
        }
        if (i == num_processes)
        {
            break;
        }
        // Allocate resources to process i
        for (int j = 0; j < num_resources; j++)
        {
            work[j] -= allocation[i][j];
            need[i][j] -= allocation[i][j];
        }
        // Update finish matrix
        finish[i] = 1;
        // If all resources are available, update available matrix
        if (work[0] == available[0] && work[1] == available[1] && work[2] == available[2])
        {
            for (int j = 0; j < num_resources; j++)
            {
                available[j] += allocation[i][j];
            }
        }
    }
    // Print final results
    printf("Final state:\n");
    printf("Available resources: ");
    for (int i = 0; i < num_resources; i++)
    {
        printf("%d ", available[i]);
    }
    printf("\n");
    printf("Allocation matrix:\n");
    for (int i = 0; i < num_processes; i++)
    {
        for (int j = 0; j < num_resources; j++)
        {
            printf("%d ", allocation[i][j]);
        }
        printf("\n");
    }
    printf("Need matrix:\n");
    for (int i = 0; i < num_processes; i++)
    {
        for (int j = 0; j < num_resources; j++)
        {
            printf("%d ", need[i][j]);
        }
        printf("\n");
    }
}
```

```

    {
        scanf("%d", &max_claim[i][j]);
    }
}

printf("Enter the allocated resources for each process: ");
for (int i = 0; i < num_processes; i++)
{
    printf("For process P%d: ", i);
    for (int j = 0; j < num_resources; j++)
    {
        scanf("%d", &allocation[i][j]);
        need[i][j] = max_claim[i][j] - allocation[i][j];
    }
    finish[i] = 0;
}

for (int i = 0; i < num_resources; i++)
{
    work[i] = available[i];
}

int count = 0;

int safe_sequence[MAX_PROCESSES];

while (count < num_processes)
{
    int found = 0;
    for (int i = 0; i < num_processes; i++)
    {
        if (finish[i] == 0)
        {
            int j;
            for (j = 0; j < num_resources; j++)
            {

```

```

    if (need[i][j] > work[j])
    {
        break;
    }
}

if (j == num_resources)
{
    for (int k = 0; k < num_resources; k++)
    {
        work[k] += allocation[i][k];
    }

    safe_sequence[count++] = i;
    finish[i] = 1;
    found = 1;
}

}

if (found == 0)
{
    break;
}

}

if (count < num_processes)
{
    printf("\nSystem is in an unsafe state. Deadlock detected.\n");
}

else
{
    printf("\nSystem is in a safe state.\nSafe sequence: ");
    for (int i = 0; i < num_processes; i++)
    {

```

```
    printf("P%d ", safe_sequence[i]);  
}  
printf("\n");  
}  
return 0;  
}
```

OUTPUT:

```
kali-linux-2023.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
File Actions Edit View Help
(kali㉿samruddhi)-[~]
$ sudo su
[sudo] password for kali:
(root㉿samruddhi)-[/home/kali]
# cd Desktop

(root㉿samruddhi)-[/home/kali/Desktop]
# gcc practical5.c

(root㉿samruddhi)-[/home/kali/Desktop]
# ./a.out
Enter the number of processes: 3
Enter the number of resources: 4
Enter the available resources: 3
4
5
6
Enter the maximum claim of each process: For process P0: 2
3
4
5
For process P1: 7
5
4
3
For process P2: 2
3
1
5
Enter the allocated resources for each process: For process P0: 2
3
4
5
For process P1: 5
2
3
4
For process P2: 5
6
7
8

System is in a safe state.
Safe sequence: P0 P1 P2

(root㉿samruddhi)-[/home/kali/Desktop]
#
```

PRACTICAL NO. 6

Implement the C program for Page Replacement Algorithms: FCFS, LRU, and Optimal for frame size as minimum three.

1) FCFS

CODE:

```
#include<stdio.h>

int main()
{
    int incomingStream[] = {4,7,6,1,7,6,1,2,7,2};
    int pageFaults = 0;
    int frames = 3;
    int m, n, s, pages;
    pages = sizeof(incomingStream)/sizeof(incomingStream[0]);
    printf(" Incoming \t Frame 1 \t Frame 2 \t Frame 3 ");
    int temp[ frames ];
    for(m = 0; m < frames; m++)
    {
        temp[m] = -1;
    }
    for(m = 0; m < pages; m++)
    {
        s = 0;
        for(n = 0; n < frames; n++)
        {
            if(incomingStream[m] == temp[n])
            {
                s++;
                pageFaults--;
            }
        }
    }
}
```

```
pageFaults++;

if((pageFaults <= frames) && (s == 0))

{

    temp[m] = incomingStream[m];

}

else if(s == 0)

{

    temp[(pageFaults - 1) % frames] = incomingStream[m];

}

printf("\n");

printf("%d\t\t\t",incomingStream[m]);

for(n = 0; n < frames; n++)

{

    if(temp[n] != -1)

        printf(" %d\t\t\t", temp[n]);

    else

        printf(" - \t\t\t");

}

printf("\nTotal Page Faults:\t%d\n", pageFaults);

return 0;

}
```

OUTPUT:

The screenshot shows a terminal window titled "kali-linux-2023.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox". The terminal displays a sequence of commands and their output related to a memory management experiment.

```
(kali㉿samruddhi)~$ sudo su
[sudo] password for kali:
(root㉿samruddhi) [/home/kali]
# cd Desktop

(root㉿samruddhi) [/home/kali/Desktop]
# gcc practical6FCFS.c

(root㉿samruddhi) [/home/kali/Desktop]
# ./a.out

Incoming      Frame 1      Frame 2      Frame 3
4             4             -             -
7             4             7             -
6             4             7             6
1             1             7             6
7             1             7             6
6             1             7             6
1             1             7             6
2             1             2             6
7             1             2             7
2             1             2             7

Total Page Faults: 6

#
```

2) LRU

CODE:

```
#include<stdio.h>

int main()
{
    int q[20],p[50],c=0,c1,d,f,i,j,k=0,n,r,t,b[20],c2[20];
    printf("Enter no of pages:");
    scanf("%d",&n);
    printf("Enter the reference string:");
    for(i=0;i<n;i++)
        scanf("%d",&p[i]);
    printf("Enter no of frames:");
    scanf("%d",&f);
    q[k]=p[k];
    printf("\n\t%d\n",q[k]);
    c++;
    k++;
    for(i=1;i<n;i++)
    {
        c1=0;
        for(j=0;j<f;j++)
        {
            if(p[i]!=q[j])
                c1++;
        }
        if(c1==f)
        {
            c++;
            if(k<f)
            {
                q[k]=p[i];
            }
        }
    }
}
```

```

k++;

for(j=0;j<k;j++)
printf("\t%od",q[j]);
printf("\n");

}

else

{

for(r=0;r<f;r++)
{

c2[r]=0;

for(j=i-1;j<n;j--)
{

if(q[r]!=p[j])
c2[r]++;
else
break;
}
}

for(r=0;r<f;r++)
b[r]=c2[r];
for(r=0;r<f;r++)
{

for(j=r;j<f;j++)
{

if(b[r]<b[j])
{
t=b[r];
b[r]=b[j];
b[j]=t;
}
}
}
}

```

```
        }

        for(r=0;r<f;r++)
        {
            if(c2[r]==b[0])
                q[r]=p[i];
            printf("\t%od",q[r]);
        }

        printf("\n");
    }

}

printf("\nThe no of page faults is %d",c);
}
```

OUTPUT:

kali-linux-2023.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

File Actions Edit View Help

```
(kali㉿samruddhi)-[~]
$ sudo su
[sudo] password for kali:
(root㉿samruddhi)-[/home/kali]
# cd Desktop

(root㉿samruddhi)-[/home/kali/Desktop]
# gcc practical6LRU.c
[root㉿samruddhi)-[/home/kali/Desktop]
# ./a.out
Enter no of pages:4
Enter the reference string:2
3
4
5
Enter no of frames:6

2
2      3
2      3      4
2      3      4      5

The no of page faults is 4

[root㉿samruddhi)-[/home/kali/Desktop]
#
```

3) OPTIMAL

CODE:

```
#include<stdio.h>

int main()
{
    int no_of_frames, no_of_pages, frames[10], pages[30], temp[10];
    int flag1, flag2, flag3, i, j, k, pos, max, faults = 0;
    printf("Enter number of frames: ");
    scanf("%d", &no_of_frames);
    printf("Enter number of pages: ");
    scanf("%d", &no_of_pages);
    printf("Enter page reference string: ");
    for(i = 0; i < no_of_pages; ++i)
    {
        scanf("%d", &pages[i]);
    }
    for(i = 0; i < no_of_frames; ++i)
    {
        frames[i] = -1;
    }
    for(i = 0; i < no_of_pages; ++i)
    {
        flag1 = flag2 = 0;
        for(j = 0; j < no_of_frames; ++j)
        {
            if(frames[j] == pages[i])
            {
                flag1 = flag2 = 1;
                break;
            }
        }
    }
```

```

if(flag1 == 0)
{
    for(j = 0; j < no_of_frames; ++j)
    {
        if(frames[j] == -1)
        {
            faults++;
            frames[j] = pages[i];
            flag2 = 1;
            break;
        }
    }
    if(flag2 == 0)
    {
        flag3 = 0;
        for(j = 0; j < no_of_frames; ++j)
        {
            temp[j] = -1;
        }
        for(k = i + 1; k < no_of_pages; ++k)
        {
            for(j = 0; j < no_of_frames; ++j)
            {
                if(frames[j] == pages[k])
                {
                    temp[j] = k;
                    break;
                }
            }
        }
        for(j = 0; j < no_of_frames; ++j)
    }
}

```

```

    {
        if(temp[j] == -1)
        {
            pos = j;
            flag3 = 1;
            break;
        }
    }

    if(flag3 == 0)
    {
        max = temp[0];
        pos = 0;
        for(j = 1; j < no_of_frames; ++j)
        {
            if(temp[j] > max)
            {
                max = temp[j];
                pos = j;
            }
        }
        frames[pos] = pages[i];
        faults++;
    }
}

printf("\n");
for(j = 0; j < no_of_frames; ++j)
{
    printf("%d\t", frames[j]);
}
printf("\n\nTotal Page Faults = %d", faults);
return 0;
}

```

OUTPUT:

The screenshot shows a terminal window in Oracle VM VirtualBox. The terminal title is "kali-linux-2023.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox". The terminal window has a dark blue background with white text. It displays the following command-line session:

```
(kali㉿samruddhi)-[~]
$ sudo su
[sudo] password for kali:
(root㉿samruddhi)-[/home/kali]
# cd Desktop

(root㉿samruddhi)-[/home/kali/Desktop]
# gcc practical6OPTIMAL.c
File System practical5.c
[root@samruddhi ~]# ./a.out
Enter number of frames: 3
Enter number of pages: 4
Enter page reference string: 1
2
3
4

1      -1      -1
1      2      -1
1      2      3
4      2      3

Total Page Faults = 4

[root@samruddhi ~]#
```

PRACTICAL NO. 7

Inter process communication in Linux using following

a. **FIFOs:** Full duplex communication between two independent processes. First process accepts sentences and writes on one pipe to be read by second process and second process counts number of characters, number of words and number of lines in accepted sentences, writes this output in a text file and writes the contents of the file on second pipe to be read by first process and displays on standard output.

FIFO

CODE:

```
#include<stdio.h>
#include<unistd.h>
#include<string.h>
#include<stdlib.h>

int main()
{
    int fd1[2], nbytes=1,fd2[2],a=0;
    pid_t pid;
    char string[80];
    char readbuffer[80];
    char ch='a',ch1='\n';
    FILE *fp;
    pipe(fd1);
    pipe(fd2);
    if((pid = fork()) == -1)
    {
        perror("fork");
        exit(1);
    }
    if(pid == 0)
    {
        close(fd1[1]);
```

```

read(fd1[0], readbuffer,sizeof(readbuffer));

printf("\nFilename '%s' is being read by Child Process through Pipe
1...\n",readbuffer);

fp=fopen(readbuffer,"r");

close(fd1[0]);

close(fd2[0]);

printf("\nContents of %s are being sent to Parent Process through Pipe
2...\n",readbuffer);

while(a!=-1)

{

    a=fscanf(fp,"%c",&ch);

    write(fd2[1], &ch,sizeof(ch));

}

close(fd2[1]);

exit(0);

}

else

{

    close(fd1[0]);

    printf("IN PARENT PROCESS\n" );

    printf("\nEnter name of file:");

    scanf("%s",string);

    printf("Filename is being sent by Parent Process to Child Process through Pipe
1...\n");

    write(fd1[1],string, (strlen(string)+1));

    close(fd1[1]);

    close(fd2[1]);

    printf("\nContents of %s are being received by Parent Process through Pipe
2...\n\n",string);

    printf("IN PARENT PROCESS\n" );

    printf("\nReceived Message:\n");

    while(nbytes!=0)

```

```
{  
    printf("%c",ch1);  
    nbytes = read(fd2[0], &ch1, sizeof(ch1));  
}  
close(fd2[0]);  
}  
return(0);  
}
```

OUTPUT:

```
kali-linux-2023.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
File Actions Edit View Help
(kali㉿ samruddhi)-[~]
$ sudo su
[sudo] password for kali:
(root㉿ samruddhi)-[/home/kali]
# cd Desktop

(root㉿ samruddhi)-[/home/kali/Desktop]
# gcc practical7.c

(root㉿ samruddhi)-[/home/kali/Desktop]
# ./a.out
IN PARENT PROCESS

Enter name of file:samruddhi
Filename is being sent by Parent Process to Child Process through Pipe 1 ...

Contents of samruddhi are being received by Parent Process through Pipe 2 ...

IN PARENT PROCESS

Received Message:

Filename 'samruddhi' is being read by Child Process through Pipe 1 ...

Contents of samruddhi are being sent to Parent Process through Pipe 2 ...
a

(root㉿ samruddhi)-[/home/kali/Desktop]
#
```

PRACTICAL NO. 8

Implement the C program for Disk Scheduling Algorithms: SSTF, SCAN, C-Look considering the initial head position moving away from the spindle.

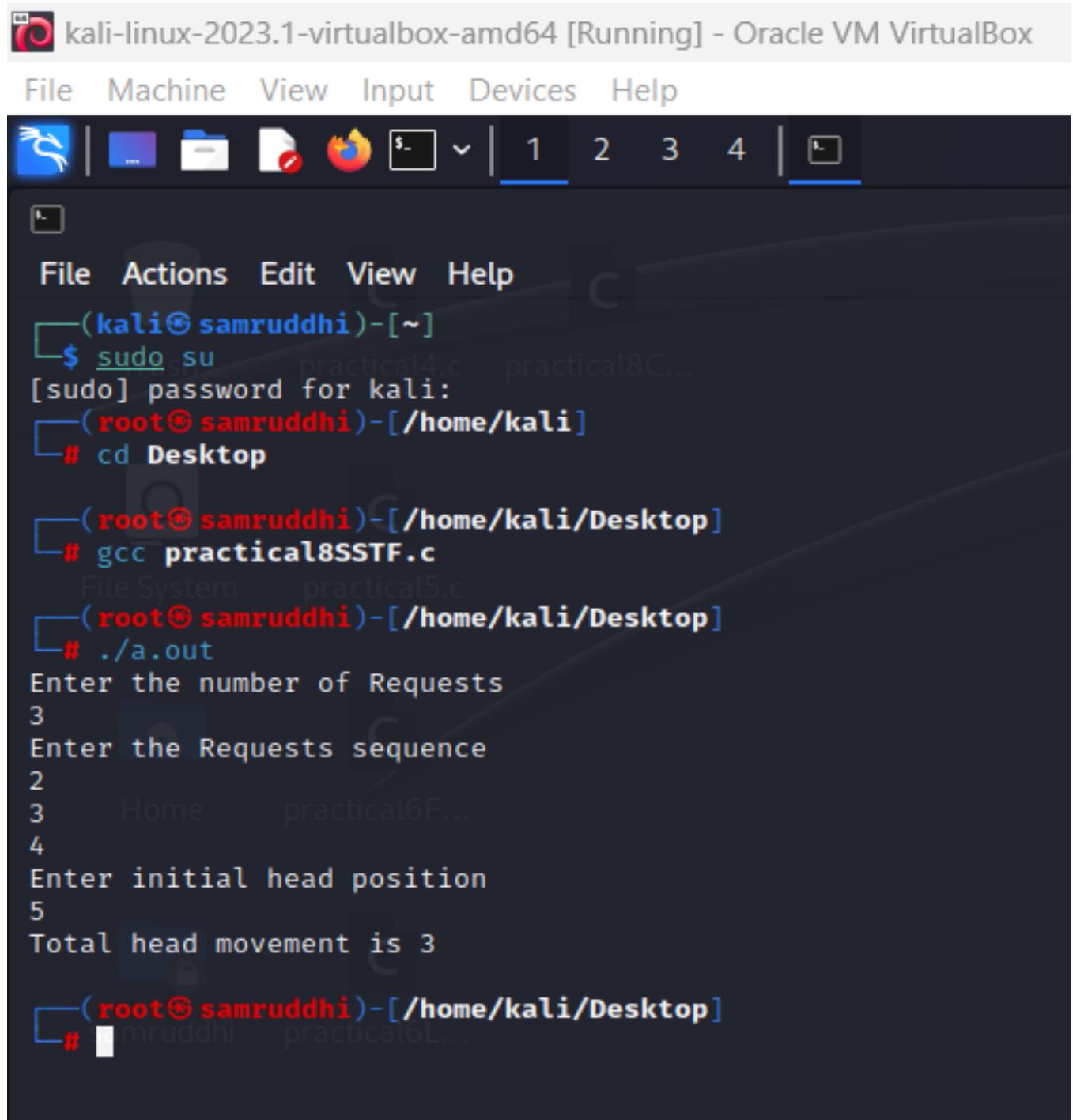
1) SSTF

CODE:

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int RQ[100],i,n,TotalHeadMoment=0,initial,count=0;
    printf("Enter the number of Requests\n");
    scanf("%d",&n);
    printf("Enter the Requests sequence\n");
    for(i=0;i<n;i++)
        scanf("%d",&RQ[i]);
    printf("Enter initial head position\n");
    scanf("%d",&initial);
    while(count!=n)
    {
        int min=1000,d,index;
        for(i=0;i<n;i++)
        {
            d=abs(RQ[i]-initial);
            if(min>d)
            {
                min=d;
                index=i;
            }
        }
        TotalHeadMoment=TotalHeadMoment+min;
```

```
initial=RQ[index];
RQ[index]=1000;
count++;
}
printf("Total head movement is %d",TotalHeadMoment);
return 0;
}
```

OUTPUT:



The screenshot shows a Kali Linux terminal window within Oracle VM VirtualBox. The terminal window has a dark background with light-colored text. It displays a command-line session where the user becomes root and runs a C program named practical8SSTF.c. The user enters three requests (2, 3, 4) and an initial head position (5). The program calculates a total head movement of 3 units. The terminal window also shows other files in the directory.

```
(kali㉿samruddhi)-[~]
$ sudo su
[sudo] password for kali:
(root㉿samruddhi)-[/home/kali]
# cd Desktop

(root㉿samruddhi)-[/home/kali/Desktop]
# gcc practical8SSTF.c
File System practical5.c
[root@samruddhi ~]# ./a.out
Enter the number of Requests
3
Enter the Requests sequence
2
3
4
Enter initial head position
5
Total head movement is 3

[root@samruddhi ~]#
```

2) SCAN

CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int main()
{
    int queue[20], n, head, i, j, k, seek = 0, max, diff, temp, queue1[20],
        queue2[20], temp1 = 0, temp2 = 0;
    float avg;
    printf("Enter the max range of disk\n");
    scanf("%d", &max);
    printf("Enter the initial head position\n");
    scanf("%d", &head);
    printf("Enter the size of queue request\n");
    scanf("%d", &n);
    printf("Enter the queue of disk positions to be read\n");
    for (i = 1; i <= n; i++)
    {
        scanf("%d", &temp);
        if (temp >= head)
        {
            queue1[temp1] = temp;
            temp1++;
        }
        else
        {
            queue2[temp2] = temp;
            temp2++;
        }
    }
}
```

```

for (i = 0; i < temp1 - 1; i++)
{
    for (j = i + 1; j < temp1; j++)
    {
        if (queue1[i] > queue1[j])
        {
            temp = queue1[i];
            queue1[i] = queue1[j];
            queue1[j] = temp;
        }
    }
}

for (i = 0; i < temp2 - 1; i++)
{
    for (j = i + 1; j < temp2; j++)
    {
        if (queue2[i] < queue2[j])
        {
            temp = queue2[i];
            queue2[i] = queue2[j];
            queue2[j] = temp;
        }
    }
}

for (i = 1, j = 0; j < temp1; i++, j++)
queue[i] = queue1[j];
queue[i] = max;

for (i = temp1 + 2, j = 0; j < temp2; i++, j++)
queue[i] = queue2[j];
queue[i] = 0;
queue[0] = head;

```

```
for (j = 0; j <= n + 1; j++)  
{  
    diff = abs(queue[j + 1] - queue[j]);  
    seek += diff;  
    printf("Disk head moves from %d to %d with seek %d\n", queue[j],  
          queue[j + 1], diff);  
}  
printf("Total seek time is %d\n", seek);  
avg = seek / (float)n;  
printf("Average seek time is %f\n", avg);  
return 0;  
}
```

OUTPUT:

The screenshot shows a terminal window titled "kali-linux-2023.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox". The terminal interface includes a menu bar with File, Machine, View, Input, Devices, and Help, and a tab bar with four tabs labeled 1, 2, 3, and 4. The main area displays the following terminal session:

```
(kali㉿samruddhi)-[~]
$ sudo su
[sudo] password for kali:
(root㉿samruddhi)-[/home/kali]
# cd Desktop

(root㉿samruddhi)-[/home/kali/Desktop]
# gcc practical8SCAN.c
File System practical5.c
(root㉿samruddhi)-[/home/kali/Desktop]
# ./a.out
Enter the max range of disk
4
Enter the initial head position
2
Enter the size of queue request
3
Enter the queue of disk positions to be read
4
5
6
Disk head moves from 2 to 4 with seek 2
Disk head moves from 4 to 5 with seek 1
Disk head moves from 5 to 6 with seek 1
Disk head moves from 6 to 4 with seek 2
Disk head moves from 4 to 0 with seek 4
Total seek time is 10
Average seek time is 3.33333

#
```

3) C-Look

CODE:

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
    int n, i, j, head, item[20], dst[20];
    int cylinders=0;
    printf("Enter no. of locations:");
    scanf("%d",&n);
    printf("Enter position of head:");
    scanf("%d",&head);
    printf("Enter elements of disk queue:");
    for(i=0;i<n;i++)
    {
        scanf("%d",&item[i]);
        dst[i]=(head-item[i]);
    }
    for(i=0;i<n-1;i++)
    {
        for(j=i+1;j<n;j++)
        {
            if(dst[j]>dst[i])
            {
                int temp=dst[j];
                dst[j]=dst[i];
                dst[i]=temp;
                temp=item[i];
                item[i]=item[j];
                item[j]=temp;
            }
        }
    }
}
```

```
        }
    }
for(i=0;i<n;i++)
{
    if(item[i]>=head)
    {
        j=i;
        break;
    }
}
printf("j=%d", j);
printf("\n\nOrder of disk allocation is as follows:\n");
for(i=j;i<n;i++)
{
    printf(" -> %d", item[i]);
    cylinders+= abs(head-item[i]);
    head=item[i];
}
for(i=0;i<j;i++)
{
    printf(" -> %d", item[i]);
    cylinders+= abs(head-item[i]);
    head=item[i];
}
printf("\n\nCylinder movement: %d\n\n", cylinders );
}
```

OUTPUT:

kali-linux-2023.1-virtualbox-amd64 [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

(kali㉿samruddhi)-[~]

\$ sudo su

[sudo] password for kali:

(root㉿samruddhi)-[/home/kali]

cd Desktop

(root㉿samruddhi)-[/home/kali/Desktop]

gcc practical8CLOOK.c

(root㉿samruddhi)-[/home/kali/Desktop]

./a.out

Enter no. of locations:3

Enter position of head:4

Enter elements of disk queue:5

2

3

j=2

Order of disk allocation is as follows:

→ 5 → 2 → 3

Cylinder movement: 5

(root㉿samruddhi)-[/home/kali/Desktop]

#