NOTRE DAME UNIVERSITY BANGLADESH

Operating System
Lab Report

Submitted to: Khorshed Alam

Submitted by: Shazidul Alam

Student ID: 0692220005101009

Batch: CSE-19

Introduction: The Linux operating system is a type of operating system that is similar to Unix and it is build upon the Linux Kernal. The Linux Kernal is like the brain of the operating system because it manages how the computer interacts with its hardware and resources. It make everything works smoothly and efficiently.

Objective: The purpose of this Lab report is to explore the fundamental operations of the Linux terminal. Which serves as a powerful tool for interacting with the operating system through command line instructions. This lab will cover essential Linux commands for navigating directories, managing files. Checking system status and handling permissions.

Tools:

* ~~VM~~ Oracle VM VirtualBox
* Ubuntu Linux

1) touch

=> Used to make new in current directory.

2) mkdir

=> Used to make new directory in current directory.

3) pwd

=> Prints present working path

4) cd

=> Used to change the file location.

5) cat

=> Views the contents of a file.

6) more

=> To view the contents of files one screen at a time.

7) ls

=> List files in a directory.

8) ls -l

=> Provides long listing of files.

9) ls -l -h

=> Provides size of files in human readable form.

10) ls -F

=> Make all the executable with * and directories with /

11) ls -a

=> Show all the file in the present directory with special dot files.

12) cp

=> Used to copy files and directories.

13) rm

=> Remove a file.

14) rmdir

=> Used to remove directory

23) chmod
=> Used to modify file access right.

24) chown
=> It is used to change the user or group owenership of a given file, directory or symbolic link.

25) redirection (>)
=> Overwrites the file with output of the command.

26) redirection (>>)
=> Appends the file with output of the command.

27) redirection (<)
=> Used to redirect standard input to a file.

28) piping (|)
=> Used to redirect standard output of one command to the standard input of another command.

29) ~~Description~~ sort
=> Sorts the standard input and sends the output to standard output.

30) Filters (uniq)
=> Given a sorted stream of data from standard input, it removes the duplicate lines of data and return the result to the standard output

31) Filters (grep)
=> Examines each line of data it receives from standard input and outputs all lines that contains a specific pattern of characters.

32) Filters (fmt)
=> Reads the text from standard input and output formatted text to standard output.

33) Filters (pr)
=> Takes the data from the standard input and splits data into pages with page breaks, footers and headers in preparation for printing.

34) Filters (head)
=> Outputs the first few lines of a file and returns it to the standard output.

35) Filters (tail)
=> Outputs the last few lines of a file and returns it to the standard output.

36) Filters (tr)

=> Translates characters, can be used to perform tasks such as uppercase to lowercase ~~to~~ conversions.

37) Job control (ps)

=> List the processes running in the system.

38) su

=> Temporarily become super user. It is used to switch from one user to another.

39) alias

=> It lets the user to give names of his/her choice to a command or sequence of commands.

40) df

=> The df command shows the size used and available space on the mounted file system of your computer. Human readable (-h) option displays the sizes in mb or gb instead of bytes. The exclude (-x) option allows you to tell df to dismount file systems you are not interested in.

41) diff

=> compares two text files and shows the difference between them. The -y (side by side) option shows the line differences side by side. The -w (width) option lets you specify the maximum line width to use to avoid wraparound lines. The suppressed-common-lines prevents diff from listing the matching lines, letting you focus on the lines which have differences.

42) echo

=> It prints the string of text to the terminal window.

43) find

=> Used to track down files that the user know exists but forget its path.

44) free

=> Gives a summary of memory usage with computer. -h option provides human friendly numbers and units.

45) groups

=> It tells which group the user is member of.

46) gzip

=> Used to compress the files. By default, it removes the original file and leaves you with the compressed version. To retain both, use -k (keep) option.

47) history

=> The history command lists the commands you have previously issued on the command line. You repeat any of the command from history list by typing enclamation mark (!) and the number of the command from the history list.

48) mv

=> Used to ~~move~~ move files and directories from directory to directory.

49) shutdown

=> Using shutdow with no parameters will shutdown the computer in one minute, shutdown now command will shutdown computer immediately.

1) touch sz

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ touch sz
shazidul@linux:~/Desktop$
```

2) mkdir File1

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ mkdir File1
shazidul@linux:~/Desktop$
```

3) pwd

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ pwd
/home/shazidul/Desktop
shazidul@linux:~/Desktop$
```

4) cd File1

```
shazidul@linux: ~/Desktop/File1
shazidul@linux:~/Desktop$ cd File1
shazidul@linux:~/Desktop/File1$
```

5) cat File1

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ cat File1
cat: File1: Is a directory
shazidul@linux:~/Desktop$
```

6) more File1

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ more File1

*** File1: directory ***
```

7) ls

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ ls
File1  sz
shazidul@linux:~/Desktop$
```

8) ls –l

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ ls -l
total 4
drwxrwxr-x 2 shazidul shazidul 4096 Nov 22 14:34 File1
-rw-rw-r-- 1 shazidul shazidul    0 Nov 22 14:32 sz
shazidul@linux:~/Desktop$
```

9) ls –l –h

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ ls -l -h
total 4.0K
drwxrwxr-x 2 shazidul shazidul 4.0K Nov 22 14:34 File1
-rw-rw-r-- 1 shazidul shazidul    0 Nov 22 14:32 sz
shazidul@linux:~/Desktop$
```

10) ls –F

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ ls -F
File1/  sz
shazidul@linux:~/Desktop$
```

11) ls –a

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ ls -a
.  ..  File1  sz
shazidul@linux:~/Desktop$
```

12) cp A B

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ cp A B
shazidul@linux:~/Desktop$ cat A
I am Ndubian.
shazidul@linux:~/Desktop$ cat B
I am Ndubian.
shazidul@linux:~/Desktop$
```

13) rm B

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ rm B
shazidul@linux:~/Desktop$ The File is Deleted
```

14) rmdir File1

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ rmdir File1
shazidul@linux:~/Desktop$ File is Removed
```

15) clear

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ rmdir File1
shazidul@linux:~/Desktop$ File is Removed
```

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$
```

16) man

```
shazidul@linux: ~/Desktop
TOUCH(1)                User Commands                TOUCH(1)

NAME
       touch - change file timestamps

SYNOPSIS
       touch [OPTION]... FILE...

DESCRIPTION
       Update  the  access  and modification times of each FILE to the current
       time.

       A FILE argument that does not exist is created empty, unless -c  or  -h
       is supplied.

       A  FILE  argument  string of - is handled specially and causes touch to
       change the times of the file associated with standard output.

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

       -a     change only the access time

       -c, --no-create
              do not create any files

 Manual page touch(1) line 1/82 25% (press h for help or q to quit)
```

17) tree File1

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ tree File1
File1
├── A
└── sz

1 directory, 2 files
```

Bug:

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ tree File1
Command 'tree' not found, but can be installed with:
snap install tree  # version 2.1.3+pkg-5852, or
apt  install tree  # version 2.0.2-1
See 'snap info tree' for additional versions.
```

18) locate

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ locate A.txt
/usr/share/doc/wireless-tools/PCMCIA.txt.gz
shazidul@linux:~/Desktop$
```

19) kill

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ ps
    PID TTY          TIME CMD
  64547 pts/0    00:00:00 bash
  64913 pts/0    00:00:00 ps
shazidul@linux:~/Desktop$ kill 64547
shazidul@linux:~/Desktop$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL   5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL 10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM 15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP 20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU 25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO   30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2       37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7       42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12      47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13      52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8       57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3       62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
shazidul@linux:~/Desktop$
```

20) less A

```
shazidul@linux: ~/Desktop

I am Ndubian.
A (END)
```

21) who

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ who
shazidul tty2         2024-11-22 13:47 (tty2)
shazidul@linux:~/Desktop$
```

22) top

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ who
shazidul tty2         2024-11-22 13:47 (tty2)
shazidul@linux:~/Desktop$ top


top - 18:09:09 up  4:23,  1 user,  load average: 0.00, 0.05, 0.09
Tasks: 256 total,   1 running, 251 sleeping,   3 stopped,   1 zombie
%Cpu(s):  0.3 us,  0.6 sy,  0.0 ni, 99.0 id,  0.0 wa,  0.0 hi,  0.2 si,  0.0 st
MiB Mem :   7561.1 total,    844.3 free,   1330.4 used,   5386.5 buff/cache
MiB Swap:   2048.0 total,   2047.7 free,      0.3 used.   5901.4 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND

   2836 shazidul  20   0 6069468 600476 138136 S   9.2   7.8  25:32.64 gnome-shell

  64529 shazidul  20   0  563236  52292  40000 S   1.8   0.7   0:01.41 gnome-terminal-

     17 root      20   0       0      0      0 I   0.5   0.0   0:17.55 rcu_preempt

   1776 root      20   0  290148   3456   3072 S   0.5   0.0   0:03.59 VBoxService

   2677 shazidul  20   0 1694168  28152  22520 S   0.5   0.4   0:47.92 pulseaudio

  60064 root      20   0       0      0      0 I   0.5   0.0   0:00.52 kworker/u17:3-events_
unbo+
      1 root      20   0  167944  13052   8188 S   0.0   0.2   0:08.00 systemd

      2 root      20   0       0      0      0 S   0.0   0.0   0:00.21 kthreadd

      3 root      20   0       0      0      0 S   0.0   0.0   0:00.00 pool_workqueue_releas
e
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/R-rcu_g
top - 18:09:37 up  4:23,  1 user,  load average: 0.00, 0.04, 0.09
Tasks: 256 total,   1 running, 251 sleeping,   3 stopped,   1 zombie
%Cpu(s):  0.2 us,  0.2 sy,  0.0 ni, 99.5 id,  0.1 wa,  0.0 hi,  0.1 si,  0.0 st
MiB Mem :   7561.1 total,    839.3 free,   1331.2 used,   5390.6 buff/cache
MiB Swap:   2048.0 total,   2047.7 free,      0.3 used.   5896.7 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   2836 shazidul  20   0 6074236 604848 142348 S   7.3   7.8  25:37.12 gnome-shell
   2677 shazidul  20   0 1694168  28152  22520 S   1.0   0.4   0:48.21 pulseaudio
     17 root      20   0       0      0      0 I   0.3   0.0   0:17.57 rcu_preempt
   2994 shazidul  20   0  324640  11732   6784 S   0.3   0.2   0:07.44 ibus-daemon
   3178 shazidul  20   0  207760  64736  49832 S   0.3   0.8   0:05.17 Xwayland
   3476 shazidul  20   0  228192   3200   2944 S   0.3   0.0   0:32.76 VBoxClient
```

23) chmod 765 File1

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ chmod 765 File1
shazidul@linux:~/Desktop$ ls -l
total 8
-rw-rw-r-- 1 shazidul shazidul   14 Nov 22 16:38 A
drwxrw-r-x 2 shazidul shazidul 4096 Nov 22 17:34 File1
-rw-rw-r-- 1 shazidul shazidul    0 Nov 22 14:32 sz
shazidul@linux:~/Desktop$
```

24) chown

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ sudo chown guest A
/etc/sudoers:45:15: syntax error
Shazidul Alam ALL=(ALL:ALL) ALL
              ^~~
[sudo] password for shazidul:
Sorry, try again.
[sudo] password for shazidul:
shazidul is not in the sudoers file.  This incident will be reported.
shazidul@linux:~/Desktop$
```

25) ls>A

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ ls>A
shazidul@linux:~/Desktop$ cat A
A
B
File1
sz
shazidul@linux:~/Desktop$
```

26) >>

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ cat A
A
B
File1
sz
shazidul@linux:~/Desktop$ echo "hello world" >> A
shazidul@linux:~/Desktop$ cat A
A
B
File1
sz
hello world
shazidul@linux:~/Desktop$
```

27) <

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ cat A
A
B
File1
sz
hello world
shazidul@linux:~/Desktop$ sort < A
A
B
File1
hello world
sz
shazidul@linux:~/Desktop$
```

28) sort < A

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ cat A
A
B
File1
sz
hello world
shazidul@linux:~/Desktop$ sort < A
A
B
File1
hello world
sz
shazidul@linux:~/Desktop$
```

29) ls | head -3

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ ls
A  B  File1  sz
shazidul@linux:~/Desktop$ ls | head -3
A
B
File1
shazidul@linux:~/Desktop$
```

30) uniq

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ cat A
A
B
File1
sz
hello world
shazidul@linux:~/Desktop$ uniq A
A
B
File1
sz
hello world
shazidul@linux:~/Desktop$ Duplicate File removed
```

31) grep

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ cat A
A
B
File1
sz
hello world
shazidul@linux:~/Desktop$ grep "h" A
hello world
shazidul@linux:~/Desktop$
```

32) fmt A

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ cat A
A
B
File1
sz
hello world
shazidul@linux:~/Desktop$ fmt A
A B File1 sz hello world
shazidul@linux:~/Desktop$
```

33) pr A

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ pr A


2024-11-22 23:01                    A                    Page 1


A
B
File1
sz
hello world
```

34) head -3 A

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ cat A
A
B
File1
sz
hello world
shazidul@linux:~/Desktop$ head -3 A
A
B
File1
shazidul@linux:~/Desktop$
```

35) tail -3 A

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ cat A
A
B
File1
sz
hello world
shazidul@linux:~/Desktop$ tail -3 A
File1
sz
hello world
shazidul@linux:~/Desktop$
```

36) tr [:lower:] [:upper:] < A

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ tr [:lower:] [:upper:] < A
A
B
FILE1
SZ
HELLO WORLD
shazidul@linux:~/Desktop$
```

37) ps

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ ps
    PID TTY          TIME CMD
  64547 pts/0    00:00:00 bash
  64919 pts/0    00:00:00 less
  64929 pts/0    00:00:00 less
  64938 pts/0    00:00:00 less
  64948 pts/0    00:00:00 top
  66265 pts/0    00:00:00 ps
shazidul@linux:~/Desktop$
```

38) su shazidul

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ su shazidul
Password:
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
```

39) alias

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ alias list="grep "h" <"
shazidul@linux:~/Desktop$ list A
hello world
shazidul@linux:~/Desktop$
```

40) df

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ df -h A
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda3        39G   12G   25G  33% /
shazidul@linux:~/Desktop$
```

41) diff

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ cat A
A
B
File1
sz
hello world
shazidul@linux:~/Desktop$ cat B
shazidul@linux:~/Desktop$ diff -y A B
A                                                       <
B                                                       <
File1                                                   <
sz                                                      <
hello world                                             <
```

42) echo

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ echo "HLW WORLD"
HLW WORLD
shazidul@linux:~/Desktop$
```

43) find

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ find . -name *A*
./File1/A
./A
shazidul@linux:~/Desktop$
```

44) free

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ free
              total        used        free      shared  buff/cache   available
Mem:        7742612     1441552      776372       37204     5524688     5962020
Swap:       2097148         268     2096880
```

45) group

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ groups guest
groups: 'guest': no such user
shazidul@linux:~/Desktop$
```

46) gzip

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ ls
 A   B   File1  'New Folder'   sz
shazidul@linux:~/Desktop$ gzip -k A
shazidul@linux:~/Desktop$ ls
 A   A.gz   B   File1  'New Folder'   sz
```

47) history

```
shazidul@linux: ~/Desktop
shazidul@linux:~/Desktop$ ls
 A   A.gz   B   File1  'New Folder'   sz
shazidul@linux:~/Desktop$ history
    1  su -
    2  sudo apt-get update
    3  clear
    4  visudo
    5  clear
    6  su -
    7  sudo -
    8  su -
    9  sudo apt-get update
   10  clear
   11  sudo apt-get update
   12  ./autorun.sh
   13  clear
   14  touch
   15  cd
   16  cd dekstop
   17  cd desktop
   18  more
   19  ls
   20  cd Desktop
   21  touch
   22  mkdir
   23  pwd
   24  touch sz
   25  cler
   26  clean
   27  clear
   28  touch Destop
   29  clear
   30  touch Shazid
   31  clear
   32  touch sz
   33  clear
   34  mkdir File1
   35  clear
   36  pwd
   37  clear
   38  cd File1
   39  clear
   40  cat File1
   41  clear
   42  cd Desktop
```

```
43  pwd
44  cd /home/shazidul/Desktop
45  clear
46  cat File1
47  cat SZ
48  clear
49  cat File1
50  more File1
51  more SZ.txt
52  clear
53  more File1
54  clear
55  ls
56  clear
57  ls -l
58  clear
59  ls -l -h
60  clear
61  ls -F
62  clear
63  ls -a
64  clear
65  cp sz
66  clear
67  ls a
68  clear
69  ls -a
70  clear
71  cp
72  clear
73  cp sz.txt
74  clear
75  cp SZ.txt File1
76  cp File1 SZ.txt
77  clear
78  cp SZ.txt File1
79  cp SZ File1
80  cp File1 SZ
81  cp File1 SZ.txt
82  clear
83  touch A
84  touch B
85  clear
86  cp A.txt B.txt
87  cp File1 B.txt
```

```
 88  cat A.txt
 89  cat A
 90  clear
 91  cp A B
 92  cat A
 93  cat B
 94  clear
 95  rm B
 96  clear
 97  rmdir
 98  clear
 99  pwd
100  clear
101  pwd
102  clear
103  rmdir /home/shazidul/Desktop/File1
104  rmdir /home/shazidul/Desktop/SZ
105  clear
106  ls
107  rmdir sz
108  edmir SZ
109  clear
110  rmdir File1
111  emdir SZ.txt
112  apt install rmdir
113  apt install 8.32-4.1ubuntu1.2
114  clear
115  ls
116  rmdir File1 SZ
117  rmdir A
118  rmdir File1
119  clear
120  rmdir File1
121  clear
122  man touch
123  clear
124  tree Desktop
125  ls
126  pwd
127  clear
128  tree /home/shazidul/Desktop
129  tree File1
130  clear
131  tree File1
132  snap install tree
```

```
133  tree File1
134  clear
135  tree File1
136  clear
137  locate A
138  apt install plocate
139  clear
140  locate File1
141  yes
142  locate
143  pwd
144  /home/shazidul/Desktop
145  mkdir /home/shazidul/Desktop
146  mkdir/home/shazidul/Desktop
147  mkdir /home/shazidul/Desktop
148  cd /home/shazidul/Desktop
149  clear
150  locate Desktop
151  sudo apt update
152  clear
153  locate "A"
154  clear
155  locate A [.txt]
156  locate -A [.txt]
157  apt install plocate
158  clear
159  locate -i *A.txt*
160  apt-file search /usr/bin/locate
161  apt install apt-file
162  apt update && apt install mlocate
163  locate A
164  clear
165  updatedb
166  apt install locate
167  locate ~/Desktop/A
168  sudo apt update
169  clear
170  su
171  cd Desktop
172  clear
173  locate A
174  man sudo_root
175  clear
176  alias list="grep "Do" <"
177  list A
```

```
178  clear
179  alias list="grep "h" <"
180  list A
181  clear
182  df -h A
183  clear
184  cat A
185  cat B
186  diff -y A B
187  clear
188  echo "HLW WORLD"
189  clear
190  find . -name *A*
191  clear
192  free
193  clear
194  ls
195  gzip -k A
196  ls
197  clear
198  group guest
199  clear
200  groups guest
201  clear
202  ls
203  history
hazidul@linux:~/Desktop$
```

48) mv

```
shazidul@linux: ~
shazidul@linux:~$ cd Desktop
shazidul@linux:~/Desktop$ ls
 A   A.gz   File1  'New Folder'   snap   sz
shazidul@linux:~/Desktop$ rm snap
rm: cannot remove 'snap': Is a directory
shazidul@linux:~/Desktop$ cd ..
shazidul@linux:~$ ls
Desktop    Downloads  Pictures  Templates
Documents  Music      Public    Videos
shazidul@linux:~$ mv Public Desktop
shazidul@linux:~$ ls
Desktop  Documents  Downloads  Music  Pictures  Templates  Videos
shazidul@linux:~$
```

49) shutdown

```
shazidul@linux: ~
shazidul@linux:~$ shutdown 20.00
```

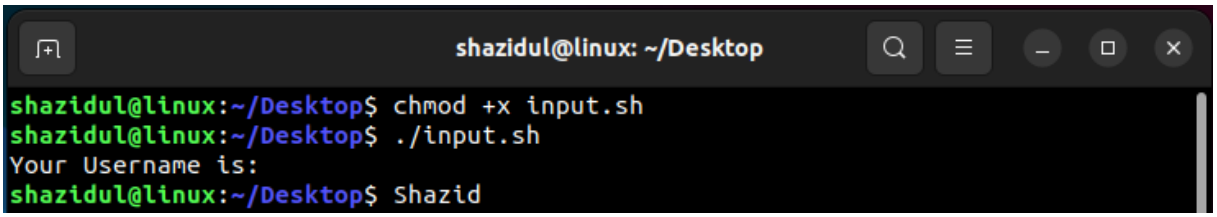
Shell Programming 1

Work: Ask for the user's name

Code:

```
chmod +x input.shShazid
./input.sh
```

Sample Input & Output:



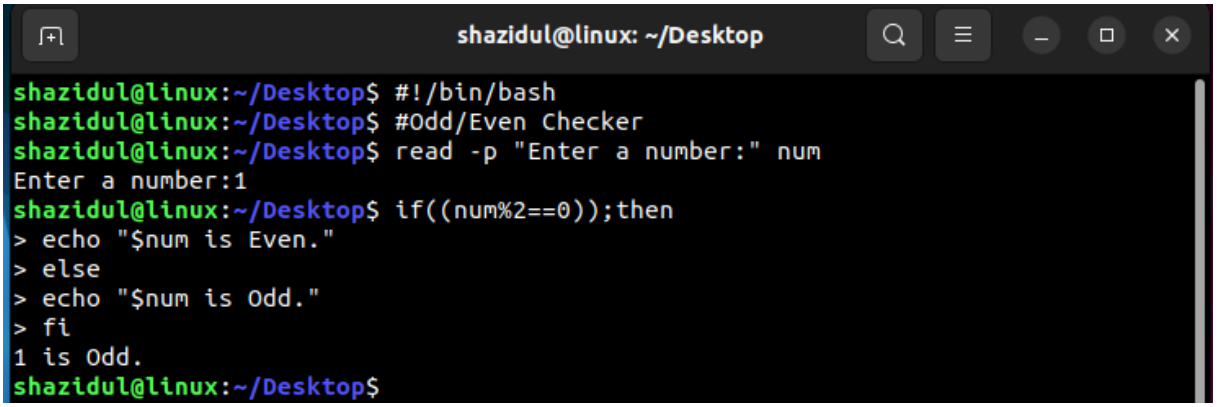
Shell Programming 2

Work: Odd/Even Checker

Code:

```
#!/bin/bash
# Odd/Even Checker
read -p "Enter a number: " num
if (( num % 2 == 0 )); then
   echo "$num is Even."
else
   echo "$num is Odd."
fi
```

Sample Input & Output:

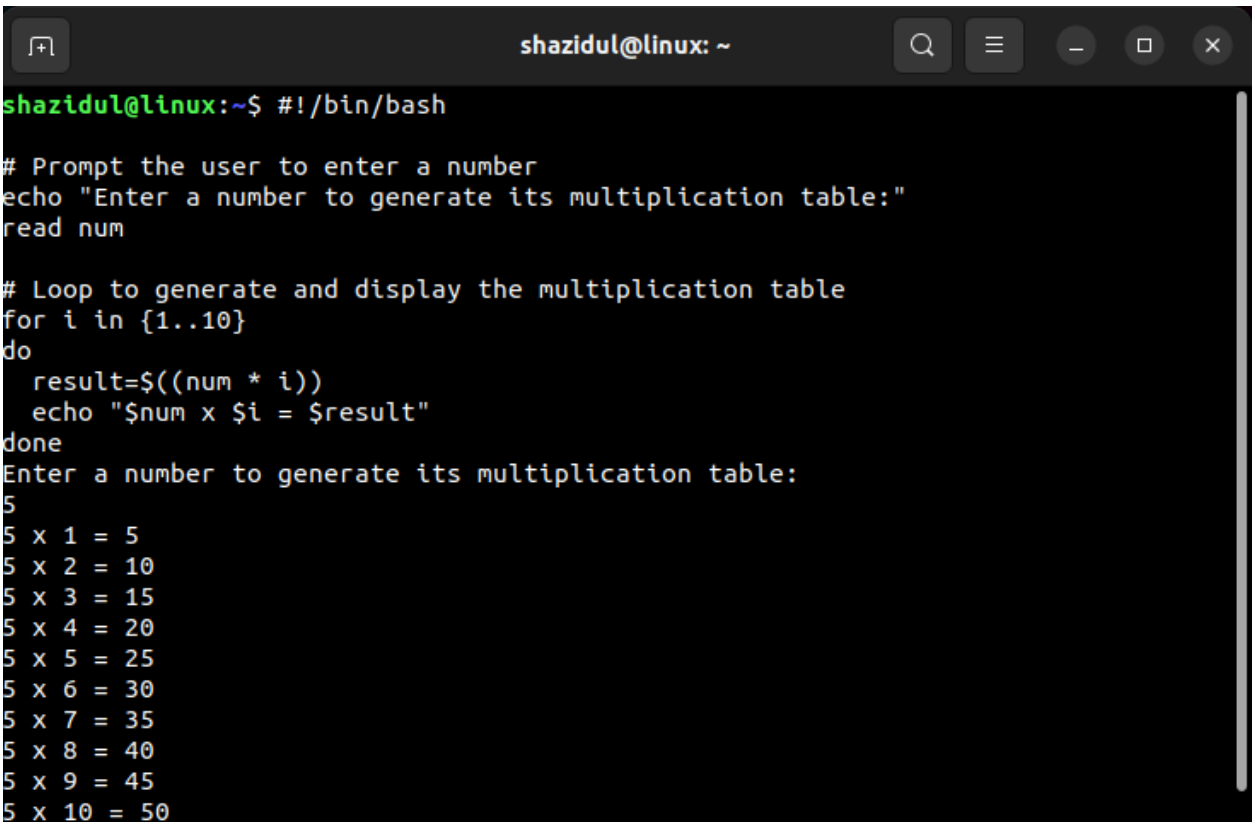


Shell Programming 3

Works: Multiplication Table

Code:

```
#!/bin/bash
# Prompt the user to enter a number
echo "Enter a number to generate its multiplication table:"
read num
# Loop to generate and display the multiplication table
for i in {1..10}
do
   result=$((num * i))
   echo "$num x $i = $result"
done
```



1) First Come First Serve (FCFS)

Code:

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <unistd.h>
#define MAX 100
int initialize();
int p[MAX], a[MAX], b[MAX], w[MAX], ta[MAX], t1, gantt[MAX][2],
gant[MAX][2];
void main()
{
int i, j, n, k, t, sum = 0, ef = 0, e= 0, m =0;
float aw, at, sum1;
FILE *fpo;
fpo = fopen("C:\output.txt", "w");
n = initialize();
printf("\nExecuting FCFS algorithm..\n");
sleep(3);
printf("\n\nProcess Burst Arrival Waiting Turnaround\n\n");
for(i = 0; i < n; i++)
{
for(j=i+1; j<n; j++)
{
if(a[i]>a[j])
{
t = a[i];
a[i] = a [j];
a[j] = t;
```



```
t = b[i];

            b[i] = b[j];
            b[j] = t;

            t = p[i];
            p[i] = p[j];
            p[j] = t;
        }
    }
  }
  t1 = a[0];
  for (i = 0; i < n; i++)
  {
    if(t1 - a[i] < 0)
    {
      gant[ef][0] = -1;
```

```
            gant[ef++][1] = t1;
            t1 += a[i] - t1;
        }
        if (i == 0)
            w[i] = 0;
        else
            w[i] = t1 - a[i];
        gant[ef][0] = a[i];
        gant[ef++][1] = t1;
        t1 += b[i];
    }
    gant[ef][1] = t1;
    for (i = 0; i< n; i++)
    {
        ta[i] = b[i] + w[i];
    }
    for (i = 0; i < n; i++)
    {
        printf("P[%d] \t %d \t %d \t %d \t\t %d\n", p[i], b[i], a[i], w[i], ta[i]);
    }
    printf("\n\nAverage waiting time is:");
    sum = 0.0;
for (i = 0; i < n; i++)
{
sum += w[i];
}
aw = sum1 / n;
printf("%f", aw);
printf("\n\nAverage turn around time is:");
sum1 = 0.0;
for (i = 0; i < n; i++)
{
sum1 += ta[i];
}
at = sum1 / n;
printf("%f", at);
fprintf(fpo, "First Come First Serve:\n\n");
for (i = 0; i< ef; i++)
{
fprintf(fpo, "P[%d],%d,%d,\t", p[i], gant[i][1], gant[i+1][1]);
}

fprintf(fpo, "\n\nAverage Waiting Time = %f\nAverage Turnaround Time =

%f\n", aw, at);

printf("\n\nThe Gantt chart is:\n\n");

printf(" ");
```

```
for (i = 0; i < ef; i++)

printf("--- ");

printf("\n");

for (i = 0; i < ef; i++)

printf(gant[i][0] == -1 ? "|  " : "| %d ", gant[i][0]);

printf("|\n ");

for (i = 0; i < ef; i++)

printf("--- ");

printf("\n");

for (i = 0; i <= ef; i++)

printf("%d   ", gant[i][1]);
 }
  int initialize()
   {
      int n, n1, n2, n3, i = 0;
      FILE *fp;
      fp = fopen("C:\input.txt", "r");
      if (fp == NULL)
      {
         printf ("Error locating the file. Please try again!\n");
         exit(1);
      }
      while (fscanf(fp, "%d,%d,%d,", &n1, &n2, &n3) != EOF)
      {
         p[i] = n1;

         if (p[i] > MAX)
         {
            printf("Woah! I am not a super computer. Please input upto 100 processes.");
            exit(1);
         }

         b[i] = n2;

         a[i] = n3;
         i++;

      }
      fclose(fp);
      return i;
   }
```

```
Executing FCFS algorithm..

Process Burst Arrival Waiting Turnaround

P[1]     3       0       0               3
P[2]     3       1       2               5
P[3]     2       2       4               6

Average waiting time is:0.000000

Average turn around time is:4.666667

The Gantt chart is:

 --- --- ---
| 0 | 1 | 2 |
 --- --- ---
0   3   6   8
Process returned 4 (0x4)   execution time : 3.160 s
```

2) Round Robin Algorithm

Code:

```
#include <iostream>
using namespace std;
void queueUpdation(int queue[],int timer,int arrival[],int n, int
maxProccessIndex){
int zeroIndex;
for(int i = 0; i < n; i++){
if(queue[i] == 0){
zeroIndex = i;
break;
}
}
queue[zeroIndex] = maxProccessIndex + 1;
}
void queueMaintainence(int queue[], int n){
for(int i = 0; (i < n-1) && (queue[i+1] != 0) ; i++){
int temp = queue[i];
queue[i] = queue[i+1];
queue[i+1] = temp;
}
}
void checkNewArrival(int timer, int arrival[], int n, int maxProccessIndex,int
queue[]){
if(timer <= arrival[n-1]){
bool newArrival = false;
for(int j = (maxProccessIndex+1); j < n; j++){
if(arrival[j] <= timer){
if(maxProccessIndex < j){
maxProccessIndex = j;
newArrival = true;
}
}
}
//adds the incoming process to the ready queue
//(if any arrives)
if(newArrival)
queueUpdation(queue,timer,arrival,n, maxProccessIndex);
}
}
//Driver Code
int main(){
int n,tq, timer = 0, maxProccessIndex = 0;
float avgWait = 0, avgTT = 0;
cout << "\nEnter the time quanta : ";
cin>>tq;
cout << "\nEnter the number of processes : ";
```

```
cin>>n;
int arrival[n], burst[n], wait[n], turn[n], queue[n], temp_burst[n];
bool complete[n];
cout << "\nEnter the arrival time of the processes : ";
for(int i = 0; i < n; i++)
cin>>arrival[i];
cout << "\nEnter the burst time of the processes : ";
for(int i = 0; i < n; i++){
cin>>burst[i];
temp_burst[i] = burst[i];
}
for(int i = 0; i < n; i++){    //Initializing the queue and complete array
complete[i] = false;
queue[i] = 0;
}
while(timer < arrival[0])    //Incrementing Timer until the first process arrives
        timer++;
    queue[0] = 1;
     while(true){
        bool flag = true;
        for(int i = 0; i < n; i++){
            if(temp_burst[i] != 0){
                flag = false;
                break;
            }
        }
        if(flag)
            break;
         for(int i = 0; (i < n) && (queue[i] != 0); i++){
            int ctr = 0;
            while((ctr < tq) && (temp_burst[queue[0]-1] > 0)){
                temp_burst[queue[0]-1] -= 1;
                timer += 1;
                ctr++;
                 //Checking and Updating the ready queue until all the processes arrive
                checkNewArrival(timer, arrival, n, maxProccessIndex, queue);
            }
            //If a process is completed then store its exit time
            //and mark it as completed
            if((temp_burst[queue[0]-1] == 0) && (complete[queue[0]-1] == false)){
                //turn array currently stores the completion time
                turn[queue[0]-1] = timer;
                complete[queue[0]-1] = true;
            }
              //checks whether or not CPU is idle
            bool idle = true;
            if(queue[n-1] == 0){
                for(int i = 0; i < n && queue[i] != 0; i++){
```

```
                if(complete[queue[i]-1] == false){
                    idle = false;
                }
            }
        }
        else
            idle = false;

        if(idle){
            timer++;
            checkNewArrival(timer, arrival, n, maxProccessIndex, queue);
        }

        //Maintaining the entries of processes
        //after each premption in the ready Queue
        queueMaintainence(queue,n);
    }
}

for(int i = 0; i < n; i++){
    turn[i] = turn[i] - arrival[i];
    wait[i] = turn[i] - burst[i];
}

cout << "\nProgram No.\tArrival Time\tBurst Time\tWait Time\tTurnAround
Time"
     << endl;
for(int i = 0; i < n; i++){
    cout<<i+1<<"\t\t"<<arrival[i]<<"\t\t"
     <<burst[i]<<"\t\t"<<wait[i]<<"\t\t"<<turn[i]<<endl;
}
for(int i =0; i< n; i++){
    avgWait += wait[i];
    avgTT += turn[i];
}
cout<<"\nAverage wait time : "<<(avgWait/n)
  <<"\nAverage Turn Around Time : "<<(avgTT/n);
return 0;    }
```

```
Enter the time [redacted] 2

Enter the number of processes [redacted] 3

Enter the arrival time of the processes : 0 1 2

Enter the burst time of the processes : 5 4 3

Program No.     Arrival Time    Burst Time      Wait Time       TurnAround Time
1               0               5               7               12
2               1               4               5               9
3               2               3               6               9

Average wait time : 6
Average Turn Around Time : 10
Process returned 0 (0x0)   execution time : 248.487 s
```

3) Banker's Algorithm

Code:

```
#include<stdio.h>
#include<stdbool.h>
static int mark[20];
int i, j, np, nr;
int main()
{
int alloc[10][10], request[10][10], avail[10], r[10], w[10];
int sequence[10];   // Array to store the safe sequence
int sequence_index = 0; // Index to track position in the sequence
printf("\nEnter the number of processes: ");
scanf("%d", &np);
printf("\nEnter the number of resources: ");
scanf("%d", &nr);
printf("\nEnter the total resources of each type: ");
for(j = 0; j < nr; j++)
scanf("%d", &r[j]);
printf("\nEnter the request matrix:\n");
for(i = 0; i < np; i++)
for(j = 0; j < nr; j++)
scanf("%d", &request[i][j]);
printf("\nEnter the allocation matrix:\n");
for(i = 0; i < np; i++)
for(j = 0; j < nr; j++)
scanf("%d", &alloc[i][j]);
/* Available Resource calculation */
for(j = 0; j < nr; j++) {
avail[j] = r[j];
        for(i = 0; i < np; i++) {
            avail[j] -= alloc[i][j];
        }
    }
    // Mark processes with zero allocation
    for(i = 0; i < np; i++) {
        int count = 0;
        for(j = 0; j < nr; j++) {
            if(alloc[i][j] == 0)
                count++;
            else
                break;
        }
        if(count == nr)
            mark[i] = 1;
    }
    // Initialize W with available resources
    for(j = 0; j < nr; j++)
        w[j] = avail[j];
```

```
    // Process each unmarked process to check if it can be satisfied
    bool progress_made;
    do {
        progress_made = false;
        for(i = 0; i < np; i++) {
            int canbeprocessed = 1;
            if(mark[i] != 1) {
                for(j = 0; j < nr; j++) {
                    if(request[i][j] > w[j]) {
                        canbeprocessed = 0;
                        break;
                    }
                }
                if(canbeprocessed) {
                    mark[i] = 1;
                    for(j = 0; j < nr; j++)
                        w[j] += alloc[i][j];
                    // Add process to safe sequence
                    sequence[sequence_index++] = i;
                    progress_made = true;
                }
            }
        }
    } while(progress_made);
    // Check for unmarked (deadlocked) processes
    int deadlock = 0;
    for(i = 0; i < np; i++) {
        if(mark[i] != 1)
            deadlock = 1;
    }
    // Output Deadlock Status and Safe Sequence if no deadlock
    if(deadlock) {
        printf("\nDeadlock detected");
    } else {
        printf("\nNo Deadlock possible\n");
        printf("Safe sequence: ");
        for(i = 0; i < sequence_index; i++) {
            printf("P%d ", sequence[i]);
        }
    }
    return 0;
}
```

```
Enter the number of processes: 4

Enter the number of resources: 3

Enter the total resources of each type: 10 5 7

Enter the request matrix:
3 2 2
1 1 1
4 0 1
2 1 2

Enter the allocation matrix:
3 2 2
2 1 1
3 3 2
2 1 3

Deadlock detected
Process returned 0 (0x0)   execution time : 78.210 s
```

4) First Fit

Code:

```
// C implementation of First - Fit algorithm
#include<stdio.h>
 // Function to allocate memory to
// blocks as per First fit algorithm
void firstFit(int blockSize[], int m, int processSize[], int n)
{
 int i, j;
 // Stores block id of the
 // block allocated to a process
 int allocation[n];
  // Initially no block is assigned to any process
 for(i = 0; i < n; i++)
 {
  allocation[i] = -1;
 }
  // pick each process and find suitable blocks
 // according to its size ad assign to it
 for (i = 0; i < n; i++)  //here, n -> number of processes
 {
  for (j = 0; j < m; j++)  //here, m -> number of blocks
  {
   if (blockSize[j] >= processSize[i])
   {
    // allocating block j to the ith process
    allocation[i] = j;
     // Reduce available memory in this block.
    blockSize[j] -= processSize[i];
     break; //go to the next process in the queue
   }
}
}
printf("\nProcess No.\tProcess Size\tBlock no.\n");
for (int i = 0; i < n; i++)
{
printf(" %i\t\t\t", i+1);
printf("%i\t\t\t\t", processSize[i]);
if (allocation[i] != -1)
printf("%i", allocation[i] + 1);
else
printf("Not Allocated");
printf("\n");
}
}
// Driver code
int main()
```

```
{
int m; //number of blocks in the memory
int n; //number of processes in the input queue
int blockSize[] = {100, 500, 200, 300, 600};
int processSize[] = {212, 417, 112, 426};
m = sizeof(blockSize) / sizeof(blockSize[0]);
n = sizeof(processSize) / sizeof(processSize[0]);
firstFit(blockSize, m, processSize, n);
return 0 ;
}
```

```
Process No.     Process Size    Block no.
 1                      212                             2
 2                      417                             5
 3                      112                             2
 4                      426                             Not Allocated
```

5) Best Fit

Code:

```
#include<iostream>
using namespace std;
 // Method to allocate memory to blocks as per Best fit algorithm
void bestFit(int blockSize[], int m, int processSize[], int n)
{
 // Stores block id of the block allocated to a process
 int allocation[n];
  // Initially no block is assigned to any process
 for (int i = 0; i < n; i++)
  allocation[i] = -1;
  // pick each process and find suitable blocks
 // according to its size ad assign to it
 for (int i = 0; i < n; i++)
 {
  // Find the best fit block for current process
  int bestIdx = -1;
  for (int j = 0; j < m; j++)
  {
   if (blockSize[j] >= processSize[i])
   {
    if (bestIdx == -1)
     bestIdx = j;
    else if (blockSize[bestIdx] > blockSize[j])
     bestIdx = j;
   }
  }
   // If we could find a block for current process
  if (bestIdx != -1)
  {
   // allocate block j to p[i] process
   allocation[i] = bestIdx;

   // Reduce available memory in this block.
   blockSize[bestIdx] -= processSize[i];
  }
 }

 cout << "\nProcess No.\tProcess Size\tBlock no.\n";
 for (int i = 0; i < n; i++)
 {
  cout << " " << i+1 << "\t\t" << processSize[i] << "\t\t";
  if (allocation[i] != -1)
   cout << allocation[i] + 1;
  else
   cout << "Not Allocated";
  cout << endl;
```

```
 }
}
 // Driver Method
int main()
{
 int blockSize[] = {100, 500, 200, 300, 600};
 int processSize[] = {212, 417, 112, 426};
 int m = sizeof(blockSize) / sizeof(blockSize[0]);
 int n = sizeof(processSize) / sizeof(processSize[0]);
  bestFit(blockSize, m, processSize, n);
  return 0 ;
}
```

```
Process No.     Process Size    Block no.
 1              212             4
 2              417             2
 3              112             3
 4              426             5

Process returned 0 (0x0)   execution time : 0.126 s
```

6) First In First Out (FIFO)

Code:

```
#include<stdio.h>
int main()
{
int incomingStream[] = {4 , 1 , 2 , 4 , 5};
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;
}
```

```
 Incoming  t Frame 1  t Frame 2  t Frame 3
4                      4                      -                      -
1                      4                      1                      -
2                      4                      1                      2
4                      4                      1                      2
5                      5                      1                      2
Total Page Faults:     4

Process returned 0 (0x0)   execution time : 0.079 s
```

7) Optimal Page Replacement Algorithm

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#define NUM_FRAMES 3
#define NUM_PAGES 10
// Function to find the page that will be referenced furthest in the future
int findOptimalPage(int page[], int pageFrames[], int index, int numFrames) {
int farthest = -1;
int farthestIndex = -1;
for (int i = 0; i < numFrames; i++) {
int j;
for (j = index; j < NUM_PAGES; j++) {
if (pageFrames[i] == page[j]) {
if (j > farthest) {
farthest = j;
farthestIndex = i;
}
break;
}
}
if (j == NUM_PAGES) {
return i;
}
}
if (farthestIndex == -1) {
return 0;
}
return farthestIndex;
}
int main() {
int pageReferences[NUM_PAGES] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3};
int pageFrames[NUM_FRAMES];
bool isPageInFrame[NUM_FRAMES];
int pageFaults = 0;
for (int i = 0; i < NUM_FRAMES; i++) {
pageFrames[i] = -1;
isPageInFrame[i] = false;
}
printf("Page Reference String: ");
for (int i = 0; i < NUM_PAGES; i++) {
printf("%d ", pageReferences[i]);
}
printf("\n");
for (int i = 0; i < NUM_PAGES; i++) {
int page = pageReferences[i];
```

```
if (!isPageInFrame[page]) {
int pageToReplace = findOptimalPage(pageReferences, pageFrames, i + 1,
NUM_FRAMES);
pageFrames[pageToReplace] = page;
isPageInFrame[pageToReplace] = true;
pageFaults++;
printf("Page %d loaded into frame %d\n", page, pageToReplace);
}
}
printf("Total Page Faults: %d\n", pageFaults);
return 0;
}
```

```
Page Reference String: 7 0 1 2 0 3 0 4 2 3
Page 7 loaded into frame 0
Page 1 loaded into frame 0
Page 2 loaded into frame 0
Page 3 loaded into frame 1
Page 4 loaded into frame 2
Page 3 loaded into frame 0
Total Page Faults: 6
```