

**DEPARTMENT OF INFORMATION TECHNOLOGY**

II B.Tech- Information Technology

**21CS301 - OPERATING SYSTEMS LAB**

**PRACTICAL RECORD**

Submitted by

**Name : RASIKA B**

**Reg.No : 727721EUIT126**

**SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY, COIMBATORE-641 008**

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**21CS301- OPERATING SYSTEMS LAB**

**PRACTICAL RECORD**

**Name : RASIKA B                                  Reg.no :727721EUIT126**

**Class    : II BTECH IT ‘C’           Semester :  III**

**BONAFIDE CERTIFICATE**

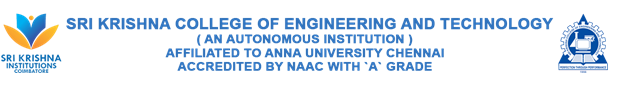
**Certified bonafide record of work done by Mr. /Ms RASIKA B**

**during the academic year 2022-2023 ( Odd Semester)**

**Staff-In Charge                                     HOD**

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.NO** | **DATE** | **NAME OF THE PROGRAM** | **TOTAL**  **MARKS**  **(100)** | **SIGN** |
| 1 | 16.8.22 | BASIC LINUX COMMANDS |  |  |
| 2 | 18.8.22 | USE OF CONTROL STATEMENTS & BRANCHING STATEMENTS |  |  |
| 3(a) | 18.8.22 | SWITCH CASE |  |  |
| 3(b) | 18.8.22 | FILE OPERATIONS |  |  |
| 4(a) | 24.8.22 | USE OF PROCESS SYSTEM CALLS |  |  |
| 4(b) | 24.8.22 | USE OF GETPID |  |  |
| 5 | 24.8.22 | FIRST IN FIRST OUT (FIFO) |  |  |
| 6 | 26.8.22 | C SIMULATION OF vi,cat,cp |  |  |
| 7 | 26.8.22 | USE OF FILE , STATS , DIRECTORY SYSTEM CALLS |  |  |
| 8 | 26.8.22 | PRODUCER CONSUMER PROBLEM |  |  |
| 9 | 1.9.22 | MEMORY MANAGEMENT |  |  |
| 10 | 1.9.22 | BANKER’S ALGORITHM |  |  |
| 11 | 5.9.22 | PAGE REPLACEMENT ALGORITHM |  |  |
| 12 | 12.9.22 | DISK SCHEDULING ALGORITHM |  |  |
| 13 | 12.9.22 | SEGMENTATION PROBLEM |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |



**Department of IT**

**Rubrics for Evaluating Laboratory**

**Subject Code : 21CS301**

**Lab Name : Operating Systems Lab**

***Method: Lab Reports and Observation of Faculty Incharge***

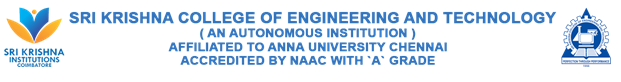
***Outcomes Assessed:***

a) Graduates will demonstrate knowledge of mathematical, scientific and multidisciplinary approach for problem solving.

 b) Graduates will be able to apply their knowledge in various programming skills to create solutions for product   based and application based software.

c) Graduates will possess the ability to create real time solutions for different projects by using modern tools prevailing in the current trends.

e) Graduates attain advanced knowledge in the stream of Information Technology and basic knowledge in Electronics and Communication Engineering to develop and maintain the simple and complex information systems.

****

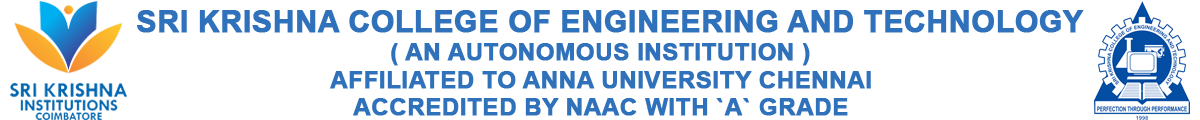
**Department of IT**

**Reg  No:727721EUIT126             Name of the Student:RASIKA B**

**Name of the lab: 21CS301 Operating Systems Lab**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Components** | **Exp No and Date** | | | | | | | | | | | | | | | | | | | | **Average Score** | |
| **Ex1** | **Ex2** | **Ex3** | **Ex4** | **Ex5** | **Ex6** | **Ex7** | **Ex8** | **Ex9** | **Ex10** | **Ex11** | **Ex12** | **Ex13** | **Ex14** | **Ex15** | **Ex16** | **Ex17** | **Ex18** | **Ex19** | **Ex20** | |  | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |
| **Aim & Algorithm**  20 Marks |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  | |
| **Coding**  30 Marks |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  | |
| **Compilation & Debugging**  30 Marks |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  | |
| **Execution & Results**  10 Marks |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  | |
| **Documentation & Viva**  10 Marks |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  | |
| **Total** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | |  | |

**Staff In-charge**



**PROGRAMME OUTCOMES**

 a) Graduates will demonstrate knowledge of mathematical, scientific and multidisciplinary approach for problem solving.

***(Criteria to be used for assessment Aim, Algorithm, Flowchart (Optional) and Description with sample Test cases, Coding, Compilation and Debugging)***

 b) Graduates will be able to apply their knowledge in various programming skills to create solutions for product   based and application based software.

***(Criteria to be used for assessment Coding, Compilation and Debugging)***

c) Graduates will possess the ability to create real time solutions for different projects by using modern tools prevailing in the current trends.

***(Criteria to be used for assessment Aim, Algorithm, Flowchart (Optional) and Description with sample Test cases, Coding, Compilation and Debugging, Execution and Results (Inclusion of Generalization like Subroutines, Modules)***

e) Graduates attain advanced knowledge in the stream of Information Technology and basic knowledge in Electronics and Communication Engineering to develop and maintain the simple and complex information systems.

***(Criteria to be used for assessment Aim, Algorithm, Flowchart (Optional) and Description with sample Test cases, Coding, Compilation and Debugging, Execution and Results (Inclusion of Generalization like Subroutines, Modules***

**Staff In-charge**

|  |  |
| --- | --- |
| **EX:NO:1** | **BASIC LINUX COMMANDS** |
| **DATE:16.08.22** |

**AIM:**

To study and demonstrate the use of basic linux commands.

**COMMANDS:**

**1.man command:**

It is an interface to the on-line reference manual to any of the command.

**Syntax:** man command name

**Ex:** man man

**2.mkdir command:**

mkdir command creates the directory, if they do not already exist.

**Syntax:** mkdir [option] directory

**Ex:** mkdir Nature

**3.cd command:**

Change directory command is used to change the current working directory.

**Syntax:** cd directoryname

**Ex:** cd Nature

**4.cd.. command:**

This command is used to move to parent directory of current directory.

**Syntax:** cd ..

**Ex:** cd ..

**5.Is command:**

It is used to list the directory contents.

**Syntax:** Is [option] [File]

**Is – t:**

This command sorts by time and date

**Ex:**Is - t

**Is – Is:**

This command displays the list with long format with file size.

**Ex:** Is – Is

**Is – r:**

This command displays the list in the reverse order.

**Ex:** Is – r

**Is – s:**

This command displays the list file size.

**Ex:** Is - s

**Is – R:**

This command displays recursively directory tree.

**Ex:** Is – R

**6.cat command:**

cat command is used to concatenate files and print on the standard output.

**Syntax:** cat [option] [File]

Cat > newfile:

To create a new file.

**Ex:** cat > Linux1

Cat file1 >> file2:

This command can append the contents of one file to the end of another file.

**Ex:** cat Linux1 >> Windows

Cat file1 file2 > file3:

This command is used to merge the contents of multiple file.

**Ex:** cat Linux1 Windows > Final

**7.cp command:**

This command is used to copy files or group of files or directory.

**Ex:** cp Linux1 linux

**Syntax:** cp Sourse destination.

**8.mv command:**

This command moves one or more files or directories from one place to another.

**Ex:** mv Linux1 copylinux

**9.pwd command:**

This command gives the full pathname of the current working directory to the standard output.

**Syntax:** pwd

**10.rm command:**

Rm command is used to remove objects such as files.

**Syntax:** rm filename

**Ex:** rm Final

**11.wc command:**

It is used to find out number of newline count, word count byte and characters.

**Syntax:** wc filename

**Ex:** wc windows

**12.Sort command:**

Sorts the contents of a text file, line by line.

**Syntax:** Sort filename

**Ex:** Sort windows

**Sort – r:**

Sorts the contents in descending order.

**Syntax:** Sort – r filename

**Ex:** Sort – r Windows

**13.history command:**

This shows the last five hundred commands we entered.

**Syntax:** history

**14.! n command:**

Does the particular command use mention.

**Syntax:** ! (number)

**Ex:** !2

**15.whoami command:**

It displays the username of the current user.

**Syntax:** whoami

**16.ps command:**

It is used for viewing information related with the processes on a system.

**Syntax:** ps

**17.grep command:**

It is used to search for a string of characters in a specified file.

**grep – v:**

It display the lines which does not match the given word.

**Syntax:** grep – v (word) filename

**Ex:** grep – v It Windows

**grep – c:**

It gives the count of the number of lines having the word.

**Syntax:** grep – c (word) filename

**Ex:** grep – c It Windows

**grep – n:**

It displays the line and line number which has the given word.

**Syntax**: grep – n (word) filename

**Ex:** grep – n It Windows

**18.cmp command:**

It used to compare the two files byte by byte and find out whether two files are identical or not.

**Syntax:** cmp file1 file2

**Ex:** cmp Windows copylinux

**19.diff command:**

This command displays the difference in the files by comparing the files line by line.

**Syntax:** diff file1 file2

**Ex:** diff Windows copylinux

**20.date command:**

It displays the current date and time.

**Syntax:** date

**21. cal command:**

Used to see the calendar of a specific month or a whole year.

**Syntax:** cal year

**Ex:** cal 2022

**Syntax:** (gives today’s date) cal month year

**Ex:** cal May 2022

**22.echo command:**

It displays the given line on the screen.

**Syntax:** echo line

**Ex:** echo Hello world

**23. vi command:**

It is used to edit file.

**Syntax:** vi filename

**Ex:** vi fileOs

**24.rmdir command:**

It is used to remove directory if it is empty.

**Syntax:** rmdir directoryname

**Ex:** rmdir OperatingSystem

**25.head command:**

It prints first n lines specified of each file given to stdout.

**Syntax:** head - number of lines filename

**Ex:** head - 3 final

**26.tail command:**

It prints last n lines specified of each file given to stdout.

**Syntax:** tail - number of lines filename

**Ex:** tail – 3 final

**27.chmod command:**

It changes the mode of the file.

**Syntax:** chmod mode filename

**Ex:** chmod 000 final

**OUTPUT:**

1.man command

NAME

       man - an interface to the system reference manuals

SYNOPSIS

       man [man options] [[section] page ...] ...

       man -k [apropos options] regexp ...

       man -K [man options] [section] term ...

       man -f [whatis options] page ...

       man -l [man options] file ...

       man -w|-W [man options] page ...

DESCRIPTION

       man  is  the system's manual pager.  Each page argument given to man is

       normally the name of a program, utility or function.  The  manual  page

       associated with each of these arguments is then found and displayed.  A

       section, if provided, will direct man to look only in that  section  of

       the  manual.   The  default action is to search in all of the available

       sections following a pre-defined order (see DEFAULTS), and to show only

       the first page found, even if page exists in several sections.

       The table below shows the section numbers of the manual followed by the

       types of pages they contain.

       1   Executable programs or shell commands

       2   System calls (functions provided by the kernel)

       3   Library calls (functions within program libraries)

       4   Special files (usually found in /dev)

       5   File formats and conventions, e.g. /etc/passwd

       6   Games

       7   Miscellaneous (including  macro  packages  and  conventions),  e.g.

           man(7), groff(7)

       8   System administration commands (usually only for root)

2.mkdir command

skcet@SK-ED-59:~$ mkdir nikita

skcet@SK-ED-59:~$ cd nikita

skcet@SK-ED-59:~/nikita$

3.cd command

skcet@SK-ED-59:~$ cd..

4.Is command

skcet@SK-ED-59:~$ ls

'Abirami WT Lab'   Desktop   Documents   Downloads   eclipse-workspace   Music   nikita   Pictures   Public   R   snap   Templates   Videos

5.skcet@SK-ED-59:~$ gedit green

^C

skcet@SK-ED-59:~$ cat green

In the world trees are in green color

Green is a wonderful color

Nature intimate in green color

I love green

But we destroy the nature,so we reduce the green

our national flag also contain green

6.skcet@SK-ED-59:~$ ls -t

 green   R   nikita   Pictures   Documents   Downloads   eclipse-workspace   snap  'Abirami WT Lab'   Desktop   Music   Public   Templates   Videos

7.skcet@SK-ED-59:~$ ls -ls

total 56

4 drwxrwxr-x  2 skcet skcet 4096 Apr 11 14:14 'Abirami WT Lab'

4 drwxr-xr-x  2 skcet skcet 4096 Apr  9 09:50  Desktop

4 drwxr-xr-x  3 skcet skcet 4096 Sep  6 09:37  Documents

4 drwxr-xr-x  3 skcet skcet 4096 Aug 13 09:57  Downloads

4 drwxrwxr-x 17 skcet skcet 4096 Aug  1 12:11  eclipse-workspace

4 -rw-rw-r--  1 skcet skcet  198 Sep 13 09:25  green

4 drwxr-xr-x  2 skcet skcet 4096 Apr  9 09:50  Music

4 drwxrwxr-x  2 skcet skcet 4096 Sep 13 09:10  nikita

4 drwxr-xr-x  2 skcet skcet 4096 Sep  6 16:18  Pictures

4 drwxr-xr-x  2 skcet skcet 4096 Apr  9 09:50  Public

4 drwxrwxr-x  3 skcet skcet 4096 Sep 13 09:13  R

4 drwx------  5 skcet skcet 4096 May 20 13:20  snap

4 drwxr-xr-x  2 skcet skcet 4096 Apr  9 09:50  Templates

4 drwxr-xr-x  2 skcet skcet 4096 Apr  9 09:50  Videos

8.skcet@SK-ED-59:~$ ls -r

 Videos   Templates   snap   R   Public   Pictures   nikita   Music   green   eclipse-workspace   Downloads   Documents   Desktop  'Abirami WT Lab'

9.skcet@SK-ED-59:~$ ls -s

total 56

4 'Abirami WT Lab'  4  Documents  4  eclipse-workspace  4  Music   4  Pictures  4  R     4  Templates

4  Desktop          4  Downloads  4  green              4  nikita  4  Public    4  snap  4  Videos

10.skcet@SK-ED-59:~$ ls -a

 .                 .bash\_history   .cache    Documents   eclipse-workspace   green    .mozilla   Pictures   Public   .Rhistory   .swt        Videos

 ..                .bash\_logout    .config   Downloads   .gnome              .java    Music      .pki       .r       snap        Templates

'Abirami WT Lab'   .bashrc         Desktop   .eclipse    .gnupg              .local   nikita     .profile   R        .ssh        .tooling

11.cat command

skcet@SK-ED-59:~$ cat green

In the world trees are in green color

Green is a wonderful color

Nature intimate in green color

I love green

But we destroy the nature,so we reduce the green

our national flag also contain green

12.cp command

skcet@SK-ED-59:~$ gedit blue

skcet@SK-ED-59:~$ cp green blue

skcet@SK-ED-59:~$ cat blue

In the world trees are in green color

Green is a wonderful color

Nature intimate in green color

I love green

But we destroy the nature,so we reduce the green

our national flag also contain green

13..mv command

skcet@SK-ED-59:~$ mv blue twin

skcet@SK-ED-59:~$ cat twin

In the world trees are in green color

Green is a wonderful color

Nature intimate in green color

I love green

But we destroy the nature,so we reduce the green

our national flag also contain green

14..wc command

skcet@SK-ED-59:~$ wc twin

  9  36 198 twin

15..sort command

skcet@SK-ED-59:~$ sort twin

But we destroy the nature,so we reduce the green

Green is a wonderful color

I love green

In the world trees are in green color

Nature intimate in green color

our national flag also contain green

skcet@sk-dk-76:~$ mkdir 113

skcet@sk-dk-76:~$ ls

 113                    Clientecho.class         files.c    SEARCH.c

 152                    Clientecho.java          hello      Serverecho.class

'152 gedit'             Clientecho.java~         hi         Serverecho.java

skcet@sk-dk-76:~$ rmdir 113

skcet@sk-dk-76:~$ ls

 152                    Clientecho.class         fair       SEARCH

'152 gedit'             Clientecho.java          files.c    search.c

 15c.cpp                Clientecho.java~         hello      SEARCH.c

 16e.cpp                client.java              hi         Serverecho.class

 18euec018              concurrent\_client.java   hk.c       Serverecho.java

16.cp

skcet@sk-dk-76:~$ cp hello good

skcet@sk-dk-76:~$ ls

 113                    Clientecho.class         files.c    search.c

 152                    Clientecho.java          good       SEARCH.c

17.rm

skcet@sk-dk-76:~$ rm good

skcet@sk-dk-76:~$ ls

 113                    Clientecho.class         files.c    SEARCH.c

 152                    Clientecho.java          hello      Serverecho.class

'152 gedit'             Clientecho.java~         hi         Serverecho.java

 15c.cpp                client.java              hk.c       Serverecho.java~

18.history

skcet@sk-dk-76:~$ !537

gedit hello

19.comparing

skcet@sk-dk-76:~$ cp hello new

skcet@sk-dk-76:~$ gedit new

skcet@sk-dk-76:~$ ^C

skcet@sk-dk-76:~$ cmp hello new

hello new differ: byte 15, line 1

skcet@sk-dk-76:~$

20.difference between two files

skcet@sk-dk-76:~$ diff hello new

1c1

< hello everyone

---

> hello everyone...

3c3

< everyone is doing good...

---

> I hope everyone is doing good...

skcet@sk-dk-76:~$

21.pwd

persent working directory

skcet@sk-dk-76:~$ pwd

/home/skcet

22.calender

skcet@sk-dk-76:~$ pwd

/home/skcet

skcet@sk-dk-76:~$ cal 8 2022

    August 2022

Su Mo Tu We Th Fr Sa

    1  2  3  4  5  6

 7  8  9 10 11 12 13

14 15 16 17 18 19 20

21 22 23 24 25 26 27

28 29 30 31

23.who am i

gives the current user name

skcet@sk-dk-76:~$ whoami

skcet

24.date

skcet@sk-dk-76:~$ date

Sat Aug 13 11:35:02 IST 2022

25.head

syntax: head -n filename

skcet@sk-dk-76:~$ head -2 hello

hello everyone

hi everyone how are you..

26.tail

skcet@sk-dk-76:~$ tail -3 hello

how about you..

Thank you..

27.ch mode

syntax: chmod  nnn filename.

-rwx-wx--x 1 skcet skcet  163 Aug 13 11:26  hello

**RESULT:**

Thus the usage of basic linux commands are demonstrated.

|  |  |
| --- | --- |
| **EX:NO:2** | **USE OF CONTROL STATEMENTS AND BRANCHING STATEMENTS** |
| **DATE:18.08.22** |

**AIM:**

To illustrate the use of shell programming using control structures.

**DESCRIPTION:**

**a)if……if statement**

The if…..if statement allows shell to make decisions and execute statements conditionally

**SYNTAX:**

if[expression]

then

statement(s)

fi

**b)if……else statement**

If else statements can be used to select an option from a given set of options.

**SYNTAX:**

if[expression]

then

statement(s)

else

statement(s)

fi

**c)while loop**

The while loop enables us to execute a set of commands repeatedly until some condition occurs.

**SYNTAX:**

while[condition]

do

statement(s)

done

**d)for loop**

The for loop operates on list of items. It repeats a set of commands for every item in a list.

**SYNTAX:**

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

do

statement(s)

done

**ALGORITHM:**

**a) To find whether a given voter is eligible to vote. (if..else.. If)**

1. START

2. get age from user

3. if age is greater than 18

Display THE VOTER IS ELIGIBLE TO VOTE

4. else

Display THE VOTER IS NOT ELIGIBLE

5. STOP

**b) To find the grade of a student using multiple if.**

1. START

2. get marks from the user

3. if marks greater then 95

Display GRADE O

4. elif marks greater then 90

Display GRADE A

5. elif marks greater than 80

Display GRADE B

6. elif marks greater than 70

Display GRADE C

7. elif marks greater than 60

Display GRADE D

8. else display FAIL!!!

9. STOP

**c) To find whether the given year is a leap year or no**

1. START

2. get the year from user

3. assign c = year % 4

4. if c is equal to 0

Display year is a leap year

5. else

Display year is not a leap year

6. STOP

**d) To find the factorial of a given number using while and for loop**

**# using while loop**

1. START

2. get the number to find factorial

3. initialize fact=1

4. when the number is greater than zero do

Assign fact = fact \* num

num=num – 1

5. repeat the step 4 until the condition becomes false

6. stop

**# using for loop**

1. start

2. get the number to find factorial

3. initialize fact = 1

4. inside for loop assign fact = fact \* i

5. repeat step 4 until the for loop condition fails

6. STOP

**e) To find the fibonacci of a given n numbers using while loop.**

1. START

2. get a number from the user

3. assign m=0 and n=1

4. while num is not equal to 2

5. do

Assign c=m+n

Assign m=n and n=c

Assign num=num-1

6. STOP

**f) To find the greatest of three numbers using if ..elif..if**

1. START

2. get three numbers from the user

3. check if num 1 is greater than num 2 and num 3

Display num 1 is greater

4. elif num 2 is greater than num 3

Display num 2 is greater

5. otherwise display num 3 is greater

6. STOP

**PROGRAM:**

**a)To find whether a given voter is eligible to vote. (if..else.. If)**

echo Enter age

read n

if [ $n -ge 18 ]

then

echo The voter is eligible to vote

else

echo The voter is not eligible

fi

**b) To find the grade of a student using multiple if.**

echo Enter marks

read n

if [ $n -ge 95 ]

then

echo GRADE O

elif [ $n -ge 90 ]

then

echo GRADE A

elif [ $n -ge 80 ]

then

echo GRADE B

elif [ $n -ge 70 ]

then

echo GRADE C

elif [ $n -ge 60 ]

then

echo GRADE D

else

echo FAIL!!!

Fi

**c) To find whether the given year is a leap year or no**

echo Enter year

read n

c=$((n % 4))

if [ $c -eq 0 ]

then

echo $n is a leap year

else

echo $n is not a leap year

fi

**d) To find the factorial of a given number using while and for loop**

#using for loop

echo Enter a number

read num

n=1

for((i=1;i<=num;i++))

do

n=$((n\*i))

done

echo The factorial is $n

#using while loop

echo Enter a number

read num

n=1

i=1

while [ $i -le $num ]

do

n=$((n\*i))

i=$((i+1))

done

echo The factorial is $n

**e) To find the fibonacci of a given n numbers using while loop.**

echo Enter a number

read num

m=0

n=1

echo $m

echo $n

while [ $num -ne 2 ]

do

c=$((m+n))

m=$n

n=$c

num=$((num-1))

echo $c

done

**f) To find the greatest of three numbers using if ..elif..if**

echo Enter three numbers

read a

read b

read c

if [ $a -gt $b ]

then

echo $a is greater

elif [ $b -gt $c ]

then

echo $b is greater

elif [ $c -gt $a ]

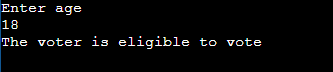
then

echo $c is greater

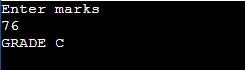
fi

**OUTPUT:**

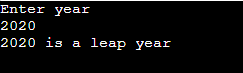
**a)To find whether a given voter is eligible to vote. (if..else.. If)**



**b) To find the grade of a student using multiple if.**

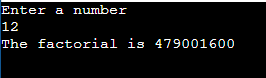


**c) To find whether the given year is a leap year or no**

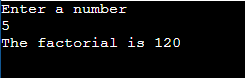


**d) To find the factorial of a given number using while and for loop**

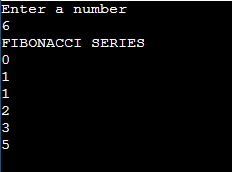
**#using for loop**



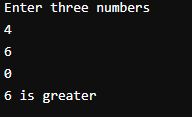
**#using while loop**



**e) To find the fibonacci of a given n numbers using while loop.**



**f) To find the greatest of three numbers using if ..elif..if**



**RESULT:**

Thus the use of control structures by shell programming were constructed and demonstrated successfully.

|  |  |
| --- | --- |
| **EX:NO:3 A** | **SWITCH CASE** |
| **DATE:18.08.22** |

**AIM:**

            To illustrate the use of switch statement by menu driven using shell programming

**ALGORITHM:**

**1.MENU DRIVEN PROGRAM FOR SOME OPERATIONS :**

1.Start

2.Read the choice from the user

3.Use switch case statement to do the operation

* 1. if choice is 1(Fibonacci series)

                      1.Start

                      2. Get the number from user

                      3. Assign x,y to 1,i to 2,x to 0

                      4. Display x and y

                      5. while (i<n)

                                  i=i+1

                                 z=x+y

                      6. Repeat until condition fails

                      7. Display z

                      8. Assign y to x,z to y

                      9. Stop

        3.2 if choice is 2(sum of numbers)

1.Start

2.Get the number from user

3.Assign o to s

4.Do the operation

 for (i=1;i<=n;i++)

      s=s+i

5.Repeat until condition fails

6. Display s

7.Stop

       3.3 if choice is 3(Armstrong number)

1.Start

2.Get the number from user,assign to t

3.Perform the operation

                                      while(n>0)

                                            r=n%10

                                            i=r\*r\*r

                                            s=s+i

                                            n=n/10

4.Repeat the step 3 until condition fails

5.if(s==t)

                                      Display Armstrong number

                              else

                                      Display not an Armstrong number

6.Stop

* 1. if choice is 4(Sum of digits)

1.Start

2.Read the input number

3.Assign o to s

4.Perform the operation

                                     while(num>0)

                                           k=num%10

                                         num=num/10

                                          s=s+k

5. Repeat step4 until condition fails

6.Display s

7.Stop

* 1. if choice is 5(Swapping of numbers)

1.Start

2.Get  2 numbers from user

3.Assign that to a.b

4.Assign a to temp

                                         b to a

                                         temp to b

5.Display a,b

6.Stop

4.Stop

**PROGRAM:**

echo 1 - Even or odd :

echo 2 - sum of n numbers :

echo 3 - Armstrong number :

echo 4 - Number is positive or negative :

echo 5 - swap two numbers :

echo Enter the choice :

read choice

case $choice in

1)

echo Enter the number :

read a

c=$((a%2))

if [ $c -eq 0 ]

then

   echo The number $a ia an even number.

else

   echo The number $a is a odd number.

fi

;;

2)

echo Enter the number :

read a

sum=0

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

do

sum+$((sum+i))

done

echo The sum of $a numbera be $sum.

;;

3)

echo Enter the number:

read a

b=$a

c=$a

d=0

while [ $a -ne 0 ]

do

d=$((d+1))

a=$((a/10))

done

while [ $a -ne 0 ]

do

m=$((b%10))

b=$((b/10))

n=$((n+(m\*\*$d)))

done

if [ $c -eq $n ]

then

   echo The number is an Armstrong number.

else

   echo The number is not an Armstrong number.

fi

;;

4)

echo enter the number :

read a

if [ $a -gt 0 ]

then

    echo The number $a is positive.

else

    echo The number $a is negative.

fi

;;

5)

read a

read b

echo Before swapping : a be $a and b be $b.

temp=0

$temp =$a

$a=$b

$b=$temp

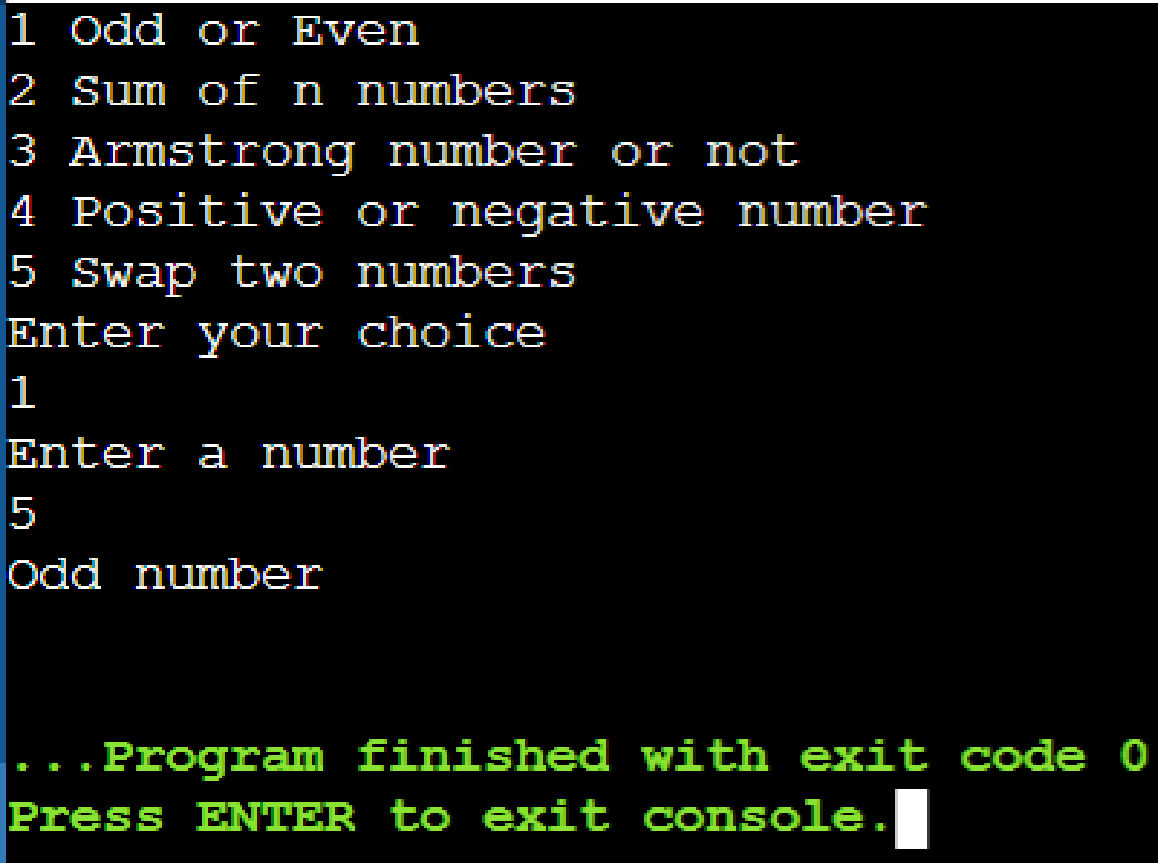
echo After swapping : a be $b and b be $a.

;;

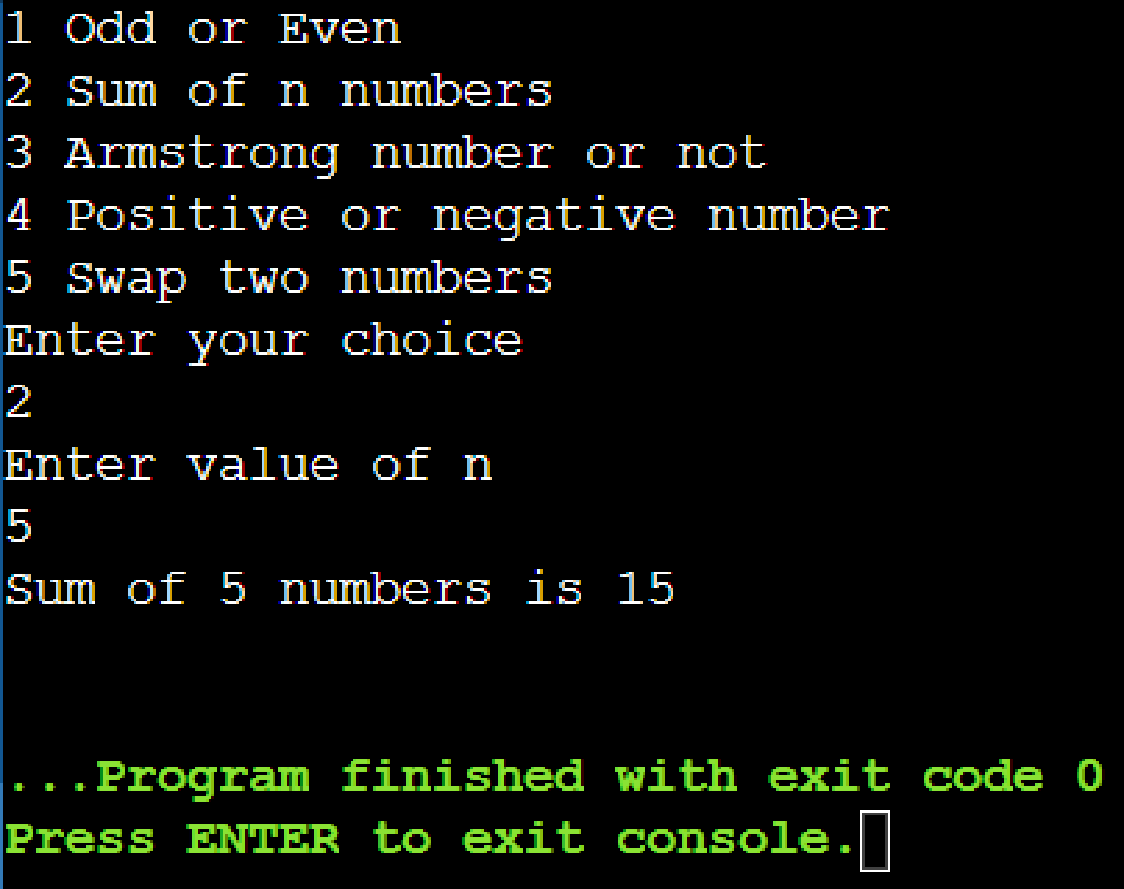
esac

**OUTPUT:**

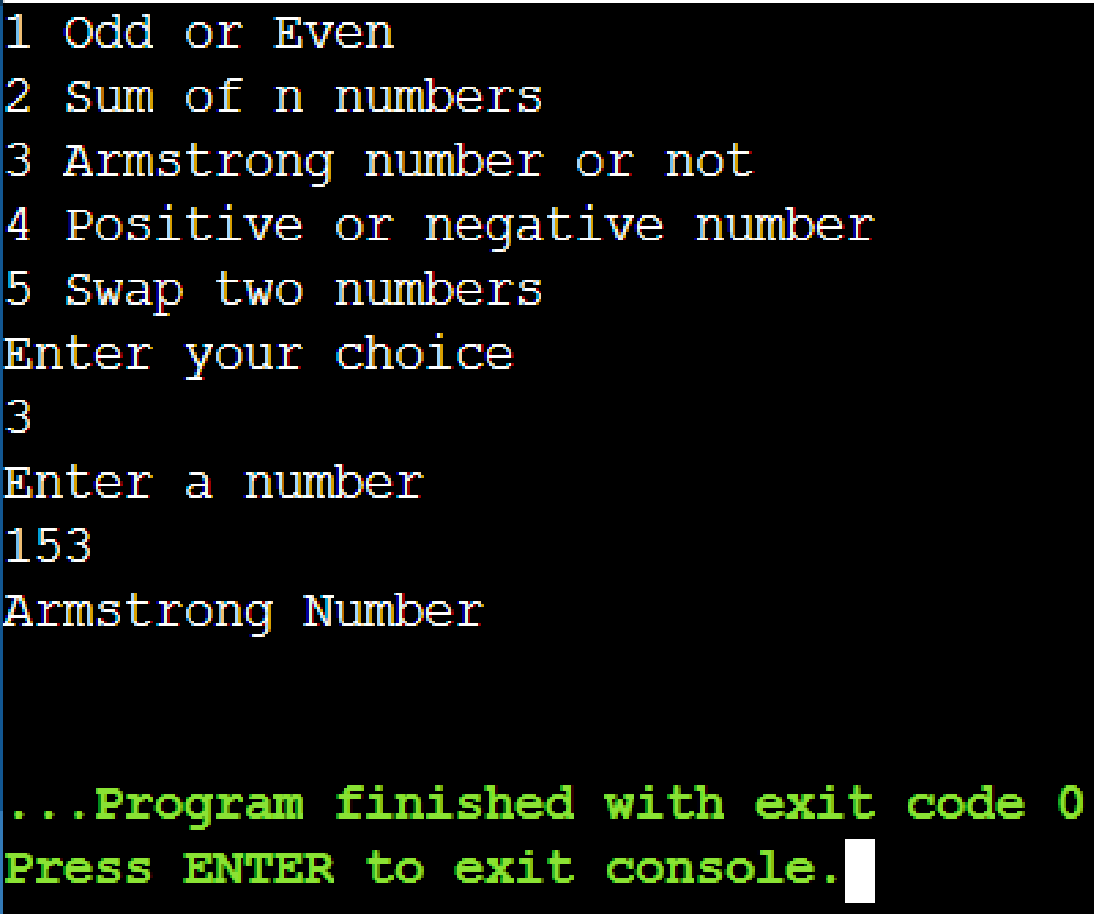
**Odd or even:**



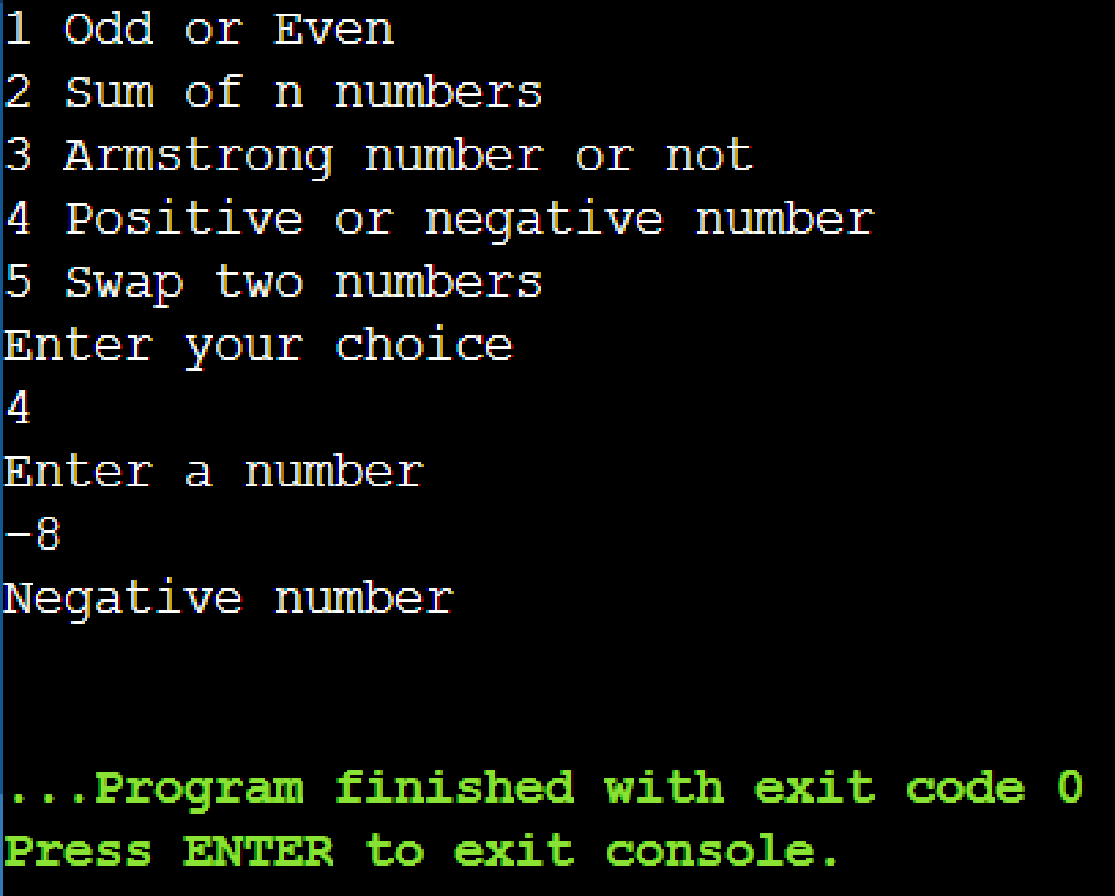
**Sum of n numbers:**



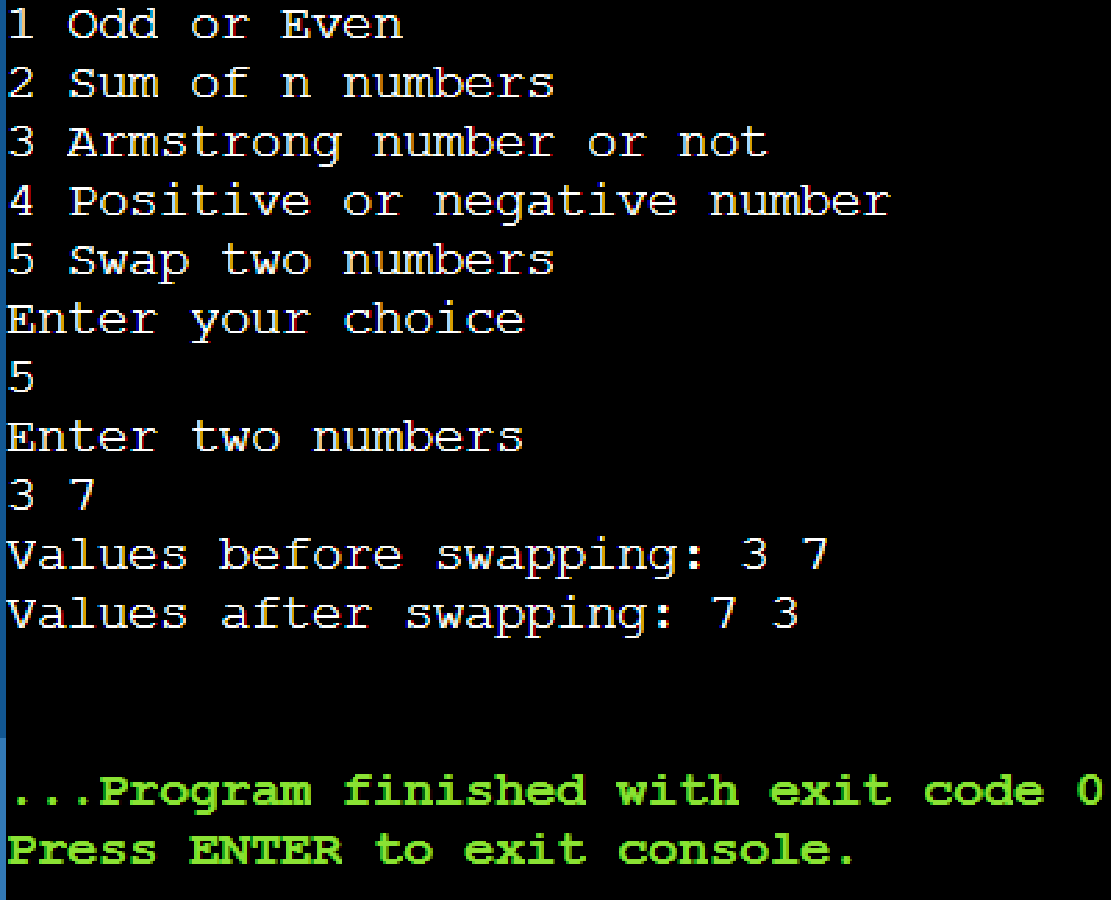
**Armstrong number:**



**Positive or negative:**



**Swap two numbers:**



**RESULT:**

Thus the use of menu driven using shell programming were constructed

and demonstrated successfully.

|  |  |
| --- | --- |
| **EX:NO:3 B** | **FILE OPERATIONS** |
| **DATE:18.08.22** |

**AIM:**

To illustrate the file operations by menu driving shell programming.

**ALGORITHM:**

1. Start

2. Using while loop, perform

3.Read the choice from user

4.Use switch case ,to perform file operations

4.1 if choice is 1(cp)

                      Read source

                      Read destination

                      cp $source $desti

4.2 if choice is 2(mv)

                      Read source

                      Read destination

                      mv $source $desti

4.3 if choice is 3(sh)

Read filename

sh $filename

4.4 if choice is 4(grep)

                        Read file name

                        Read word

                        Read option

                        Grep-$opt $word $file

              4.5 if choice is 5(sort)

                            Read filename

                            Read option

                            sort $option $file

4.6 if choice is 6(cat)

                       Read file name

                       cat $file name

5.Stop

**PROGRAM:**

echo File operations:

echo 1 Copy files

echo 2 Move files

echo 3 Sort contents of the files

echo 4 Display contents of the files

echo 5 File match

echo 6 Execute the file

echo Enter your choice

read ch

case $ch in

1)

echo Enter file1

read file1

echo Enter file2

read file2

cp $file1 $file2

;;

2)

echo Enter file1

read file1

echo Enter file2

read file2

mv $file1 $file2

;;

3)

echo Enter file to be sorted

read s

sort $s

;;

4)

echo Enter file to be displayed

read s

cat $s

;;

5)

echo File match

read s

echo Enter word to be matched

read m

echo Enter any operation

read op

case $op in

v)

grep -v $m $s

;;

c)

grep -c $m $s

;;

h)

grep -h $m $s

;;

esac

;;

6)

echo Enter file to be executed

read s

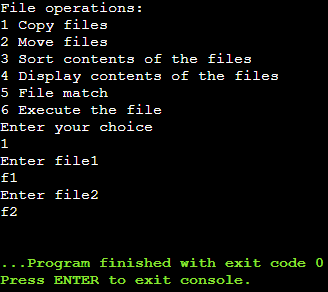
sh $s

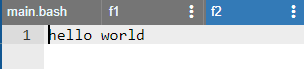
;;

Esac

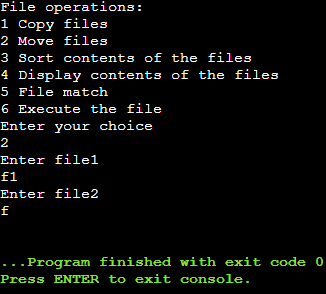
**OUTPUT:**

**Copy files**

****

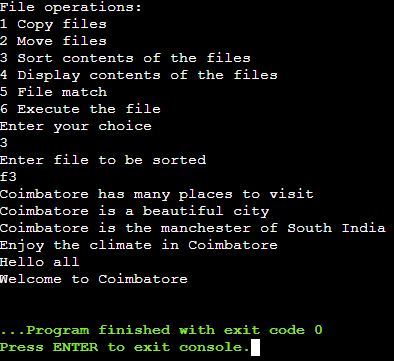
****

**Move files**

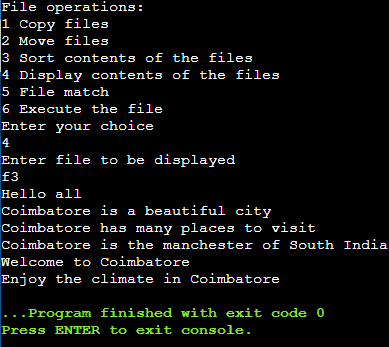




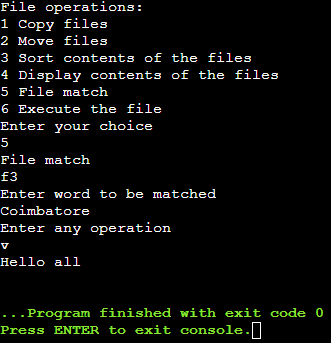
**Sort**

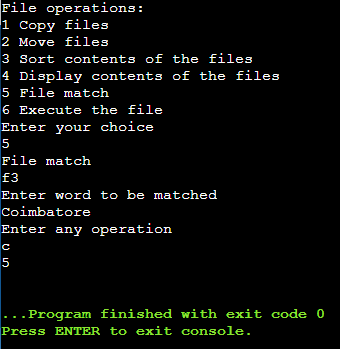


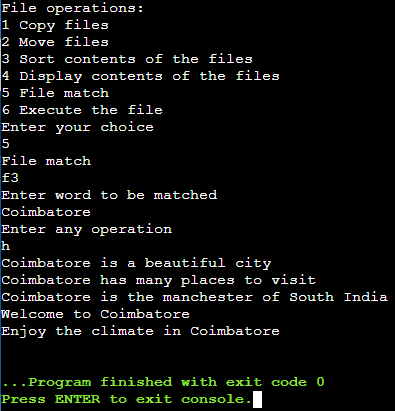
**Display file**



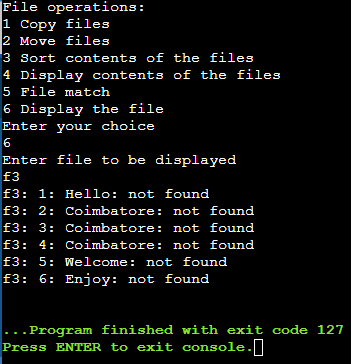
**File match**







**Execute the file**

****

**RESULT:**

Thus the use of menu driven using shell programming were constructed

and demonstrated successfully.

|  |  |
| --- | --- |
| **EX:NO:4A** | **USE OF PROCESS OF SYSTEM CALLS** |
| **DATE:24.08.22** |

**AIM:**

To illustrate the use of process of system calls using C program

**ALGORITHM:**

1. Start
2. Declare pid
3. Create a new process using fork()
4. Perform if pid<0

Display fork cannot be created

else if pid==0

Display parent by getppid and child by getpid

Else

Display parent by getpid and grandparent getppid

1. Stop

**INFERENCE:**

Processes use the fork() system call to create a program that is a copy of themselves.

This is one of the major methods of process creation in operating systems.

When a parent process creates a child process and the execution of the parent process is suspended until the child process executes.

The process which is called fork() call is the parent process and the process which is created newly is the child process.

The child process will be exactly the same as the parent .

**PROGRAM:**

#include<stdio.h>

#include<unistd.h>

#include<stdlib.h>

void main()

{

int pid;

pid=fork();

if (pid < 0)

{

printf("The fork cannot be created");

exit(0);

}

else

if (pid==0)

{

execlp("/bin/ps","ps");

printf("\n The process id of the child: %d", getpid());

printf("\n The process id of the parent: %d", getppid());

}

else{

printf("\n The process id of the parent: %d", getpid());

printf("\n The process id of the grandparent: %d", getppid());

}

}

**OUTPUT:**



**RESULT:**

Thus the use of process system call using c program has been illustrated and executed

successfully.

|  |  |
| --- | --- |
| **EX:NO:4B** | **USE OF GETPID** |
| **DATE:24.08.22** |

**AIM:**

To illustrate the use of process of system calls using C program

**ALGORITHM:**

1.Start

2.Declare pid

3.Create a new process using fork()

4.Perform if pid<0

Display fork cannot be created

else if pid==0

Display parent by getppid and child by getpid

Else

Display parent by getpid and grandparent getppid

5.Stop

**INFERENCE:**

Processes use the fork() system call to create a program that is a copy of themselves.

This is one of the major methods of process creation in operating systems.

When a parent process creates a child process and the execution of the parent process is suspended until the child process executes.

The process which is called fork() call is the parent process and the process which is created newly is the child process.

The child process will be exactly the same as the parent .

**PROGRAM:**

#include<stdio.h>

#include<unistd.h>

#include<stdlib.h>

void main()

{

int pid;

pid=fork();

if (pid < 0)

{

printf("The fork cannot be created");

exit(0);

}

else

if (pid==0)

{

execlp("/bin/ps","ps");

printf("\n The process id of the child: %d", getpid());

printf("\n The process id of the parent: %d", getppid());

}

else{

printf("\n The process id of the parent: %d", getpid());

printf("\n The process id of the grandparent: %d", getppid());

}

}

**OUTPUT:**



**RESULT:**

Thus the use of process system call using c program has been illustrated and executed

Successfully.

|  |  |
| --- | --- |
| **EX NO: 5** | **FCFS CPU SCHEDULING** |
| **DATE:24.08.22** |

**AIM:**

To illustrate FCFS CPU Scheduling using C Program.

**ALGORITHM:**

1. Start
2. Declare the variables
3. Input the number of process from user
4. Using for loop input the arrival time and burst time for each process
5. Using for loop, for each process

Calculate turn around time by

TAT = completion - arrival

Calculate waiting time by

WT = Turn around time – burst time

6. Now calculate average turn around time and waiting time

7. Display every calculated values

8. Stop.

**PROGRAM:**

#include<stdio.h>

int main(){

 int bt[10]={0},wt[10]={0},ct[10]={0};

float at[10]={0},tat[10]={0};

int n,sum=0;

float totalTAT=0,totalWT=0;

 printf("\_\_\_FCFS CPU SCHEDULING\_\_\_");

printf("\n\nEnter number of processes ");

scanf("%d",&n);

printf("Enter arrival time and burst time for each process\n\n");

 for(int i=0;i<n;i++)

{

printf("Arrival time of process[%d] ",i+1);

scanf("%f",&at[i]);

printf("Burst time of process[%d] ",i+1);

scanf("%d",&bt[i]);

printf("\n");

}

for(int j=0;j<n;j++)

{

sum+=bt[j];

ct[j]+=sum;

}

 for(int k=0;k<n;k++)

{

tat[k]=ct[k]-at[k];

totalTAT+=tat[k];

}

 for(int k=0;k<n;k++)

{

wt[k]=tat[k]-bt[k];

totalWT+=wt[k];

}

printf("Solution: \n\n");

printf("P\t AT\t BT\t CT\t TAT\t WT\t\n\n");

for(int i=0;i<n;i++)

{

printf("P%d\t %.2f\t %d\t %d\t %.2f\t %d\n",

i+1,at[i],bt[i],ct[i],tat[i],wt[i]);

}

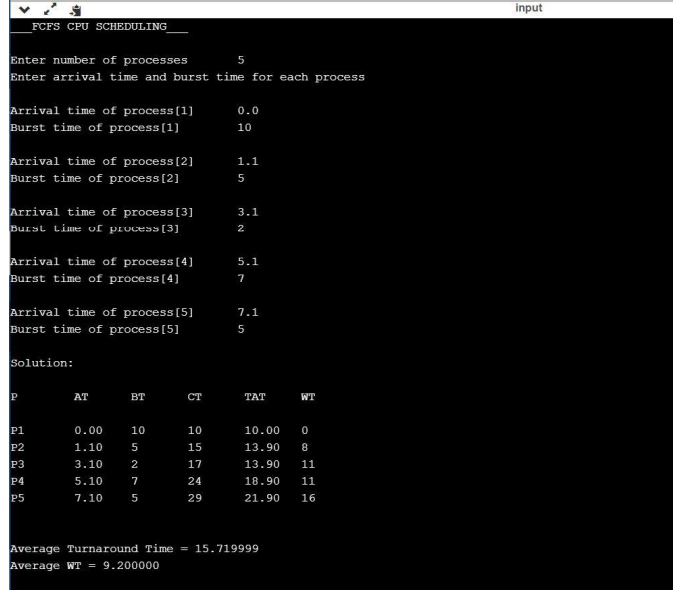
printf("\n\nAverage Turnaround Time = %f\n",totalTAT/n);

printf("Average WT = %f\n\n",totalWT/n);

return 0;

}

**OUTPUT:**



**RESULT:**

Thus the FCFS CPU scheduling using c program has been illustrated and executed successfully.

|  |  |
| --- | --- |
| **EX.NO:6** | **PRODUCER CONSUMER PROBLEM** |
| **DATE:26.08.22** |

**AIM:**

To illustrate interprocess communication producer consumer problem using c program.

**ALGORITHM:**

**1.**Start

2.Declare the variables

3.Using switch case get the choice from user

Case 1:

Call the producer function

Get the data from the user

Add it to buffer front

Front =(front+1)%5

Increment count

If (Consumersleep==1 and count==1)

Display consumer is now ready

Else

Display Buffer is full

Producersleep is one

Case 2:

Call the consumer function

Get the item from user

Buffer [rear]= “ “

Now display the consumed items

Tear + (Tear+1) %5

Decrement count

If producersleep ==1 and count==4

Display Producer is now ready

Else

Display Buffer is empty

Consumer sleep is,

Case 3:

Call view function

Using for loop

Display buffer data

Case 4:

Exit

4.Stop.

**PROGRAM:**

#include<stdio.h>

#include<conio.h>

#define N 5;

int count=0;

int front=0;int rear=0;

char buffer[7];

int prodsleep=0;int consleep=0;

void producer(void){

char item;

if (count<5){

printf("Enter data :");

scanf(" %c",&item);

buffer [front]=item;

front = (front+1)%5;

count++;

if(consleep==1 && count==1){

printf("\n Consumer is now ready ");

}

}

else{

printf("\n Buffer is full...");

prodsleep=1;

}

}

void consumer(void){

char item;

if (count>0){

item = buffer[rear];

buffer[rear]=' ';

printf("\n C: %c",item);

rear=(rear+1)%5;

count--;

if(prodsleep==1 && count==4)

{

printf("\n Producer is now ready");

}

}

else{

printf("\n Buffer is empty...");

consleep=1;

}

}

void view(void)

{

int i;

printf("\n Data of buffer: ");

for(i=0;i<5;i++){

printf("- %c ",buffer[i]);

}

}

void main(){

int i,choice,flag=0;

printf("\_\_\_PRODUCER CONSUMER\_\_\_\n");

printf("\n 1: Produce item ");

printf("\n 2: Consume item ");

printf("\n 3: To view buffer ");

printf("\n 4: Exit");

do{

printf("\n\n Enter your choice :");

scanf("%d",&choice);

switch(choice){

case 1:producer();

break;

case 2:consumer();

break;

case 3:view();

break;

case 4:flag=1;

break;

default:printf("\n Enter correct choice");

break;}

}

while(flag==0);

}

**OUTPUT:** **RESULT:**

Thus the interprocess communication producer consumer problem has been illustrated and executed successfully**.**

|  |  |
| --- | --- |
| **EX:NO:7** | **C-SIMULATION** **OF VI,CAT AND CP** |
| **DATE:26.08.22** |

**AIM:**

To illustrate the simulation of vi,cat and cp using c program.

**ALGORITHM:**

1. START
2. Declare the required variables.
3. Input the choice from the user.
4. Perform the operation using switch case.
5. Case 1(vi)

Get the file from the user

Using while loop

While(a!=’\*’){

Fput c(a,file)

a=getchar()

}

Fclose(f1)

1. Case 2(cat)

Get the file name

File must be in read mode

if (file 1==’\0’)

DISPLAY FILE IS EMPTY

Else

a=fgetc(file 1)

using while loop perform

while(a!=EOF){

DSPLAY a

a=fgetc(file 1)

}

fclose(f1)

1. Case 3(cp)

Get the file name for the source and destination

Source file should be in read mode and destination in write mode

if (file1 ==’\0’ && file2==’\0’)

DISPLAY File is empty

else

b=fgetc(file 1)

using while loop perform

while (b!=EOF){

fputc(b,file 1);

b=getc(file 1)

}

fclose (f1)

1. By default if no cases matches

DISPLAY Enter valid option

1. STOP

**PROGRAM:**

#include<stdio.h>

#include<stdlib.h>

int main()

{

 int ch;

 char a,b,file1[10],file2[10];

 FILE \*f1,\*f2;

 printf("MENU");

 printf("\n1.Press 1 for vi \n2.Press 2 for cat \n3.Press 3 for cp ");

 printf("\nEnter your choice: ");

 scanf("%d",&ch);

 switch(ch)

 {

 case 1:

 printf("\n\_\_\_vi command\_\_\_");

 printf("\nEnter the file name: ");

 scanf("%s",file1);

 f1=fopen(file1,"w");

 a=getchar();

 while(a!='\*')

 {

 fputc(a,f1);

 a=getchar();

 }

 fclose(f1);

 break;

 case 2:

 printf("\n\_\_\_\_cat command\_\_\_");

 printf("\nEnter the file name: ");

 scanf("%s",file1);

 f1=fopen(file1,"r");

 if(f1=='\0')

 {

 printf("\n File is empty");

 exit(0);

 }

 else

 {

 a=fgetc(f1);

 while(a!=EOF)

 {

 printf("%c",a);

 a=fgetc(f1);

 }

 }

 fclose(f1);

 break;

 case 3:

 printf("\n\_\_\_cp command\_\_\_");

 printf("\nEnter the source file name: ");

 scanf("%s",file1);

 printf("\nEnter the destination file name: ");

 scanf("%s",file2);

 f1=fopen(file1,"r");

 f2=fopen(file2,"w");

 if(f1=='\0' && f2=='\0')

 {

 printf("\nFile is empty");

 }

 else

 {

 b=fgetc(f1);

 while(b!=EOF)

 {

 fputc(b,f2);

 b=getc(f1);

 }

 }

 fclose(f1);

 fclose(f2);

 printf("\n File is copied successfully");

 break;

 default:

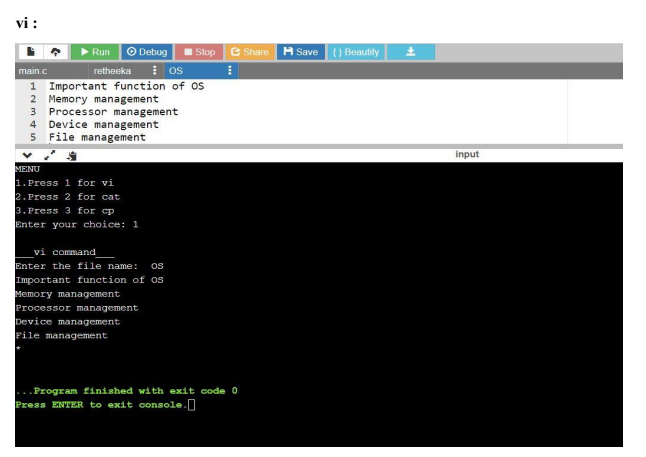
 printf("\nEnter a valid option");

 }

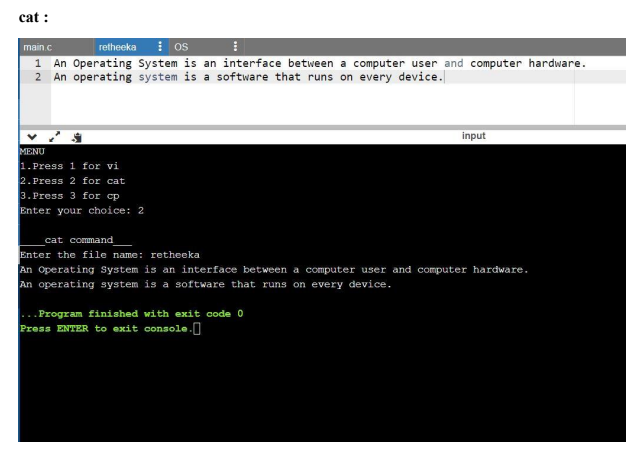
}

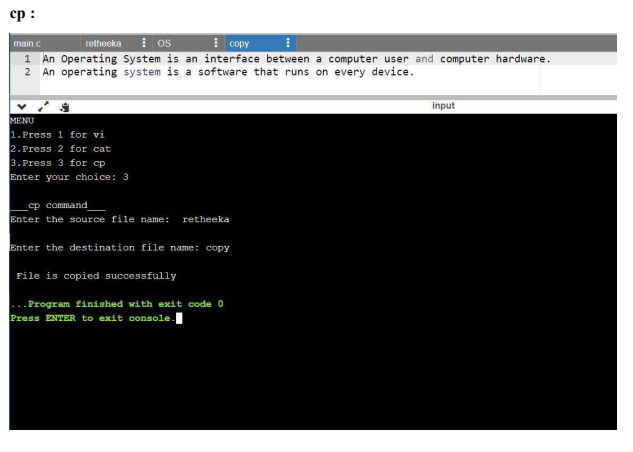
**OUTPUT:**

**vi:**



**cat:**





**RESULT:**

Thus the simulation of vi,cat and cp using c program has been illustrated and executed  successfully.

|  |  |
| --- | --- |
| **EX.NO:08-A** | **USE OF FILE SYSTEM CALLS** |
| **DATE:26.08.22** |

**AIM:**

To write a program to establish the concept of file system call and its uses.

**ALGORITHM:**

1. Start
2. Import the necessary reader file
3. Declare necessary variables
4. Declare a static char message
5. Assign a string ‘HELLOWORLD’ to message
6. Declare char buffer
7. Open the file
8. if fd=-1
9. print file is opened for read/write access
10. write the message into the file
11. lseek(fd,ol,o)
12. if
13. print message written to file
14. else
15. print error and close the file
16. else print file exists
17. stop

**PROGRAM:**

#include <fcntl.h>

#include <stdio.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <stdlib.h>

static char message [] = "Hello, world";

int main()

{

int fd;

char buffer [80];

fd = open("df.dat",O\_RDWR | O\_CREAT | O\_EXCL, S\_IREAD |

S\_IWRITE);

if (fd != -1)

{

printf("datafile df.dat opened for read/write access\n");

write(fd, message, sizeof(message));

lseek(fd, 0L, 0);

if (read(fd, buffer, sizeof(message)) == sizeof (message))

printf("\"%s\" was written to datafile.dat\n", buffer);

else

printf("\*\*\* error reading datafile.dat \*\*\*\n");

close (fd);

}

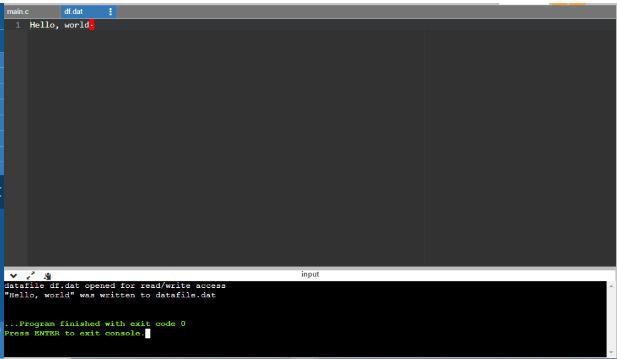
else

printf("\*\*\* datafile.dat already exists \*\*\*\n");

exit (0);

}

**OUTPUT:**



**RESULT:**

Thus the use of stat system call using c program has been illustrated and executed

successfully

|  |  |
| --- | --- |
| **EX.NO:08-B** | **USE OF STAT SYSTEM CALLS** |
| **DATE: 26.08.22** |

**AIM:**

To wite a program to exhibit the concept of stat system calls and its uses.

**ALGORITHM:**

1. Start
2. Declare s structure and a variable s
3. Declare necessary variables
4. If(stat(“test’,&s)==-1)
5. Show a perror
6. Exit
7. Compute size of input files
8. Print the file sizes

**PROGRAM :**

#include <stdio.h>

#include <sys/stat.h>

#include <stdlib.h>

int main()

{

struct stat s;

int a; int b;

if(stat("HEMA",&s)==(-1))

{

perror("Error: cannot stat file");

exit(0);

}

a=s.st\_blksize;

b=s.st\_size;

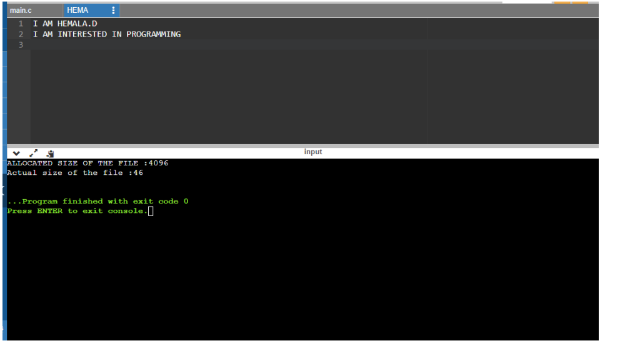
printf("ALLOCATED SIZE OF THE FILE :%d\nActual size of the file

:%d\n",a,b);

return 0;

}

**OUTPUT:**



**RESULT:**

Thus the use of stat system call using c program has been illustrated and executed

successfully

|  |  |
| --- | --- |
| **EX.NO:08-C** | **USE OF DIRECTORY SYSTEM CALLS** |
| **DATE: 26.08.22** |

**AIM:**

To write a program to illustrate the concept of directory system calls.

**ALGORITHM:**

1. Start
2. Im[port necessary header files and variables
3. Search( )
4. Create a pointer for DIR
5. If file is empty or directory is empty
6. Print unable to open directory
7. Get the name of file to be searched
8. While dir is empty or NULL
9. Check for the file
10. Add 1 to flag
11. If flag is 1
12. Print file is found
13. Else
14. Print file not found
15. Main( )
16. Get name of directory
17. Search (name)
18. Stop

**PROGRAM :**

#include<stdio.h>

#include<dirent.h>

#include<stdlib.h>

#include<string.h>

void sea(char \*dname)

{

DIR \*dir;

struct dirent \*ent;

int flag = 0;

char a[15];

if ((dir= opendir(dname))==NULL)

{

printf("\n unable to open directory ");

exit(1);

}

printf("\n Enter the name of the file to be searched :");

scanf("%s",a);

while((ent=readdir(dir))!=NULL)

{

if(!strcmp(a,ent->d\_name))

{

printf("%s",ent->d\_name);

flag++;

}

}

if(flag==1)

printf("\n the given file is found\n\n");

else

printf("\nfile not found");

if(closedir(dir)!=0)

printf("unable to close directory");

}

void main()

{

char dirname[25];

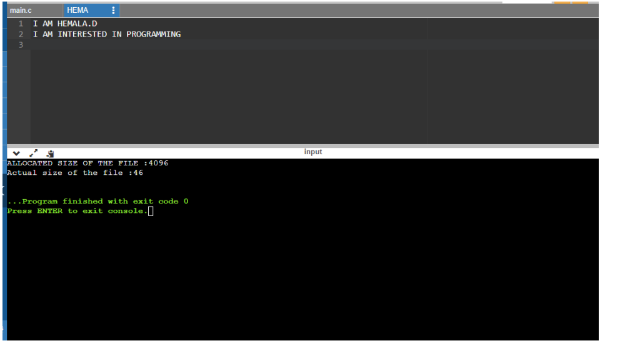
printf("\n Enter the directory to be searched :\n");

scanf("%s",dirname);

sea(dirname);

}

**OUTPUT:**



**RESULT:**

Thus the use of stat system call using c program has been illustrated and executed

Successfully.

|  |  |
| --- | --- |
| **EX NO: 9 A** | **MEMORY MANAGEMENT –**  **FIRST FIT ALGORITHM** |
| **DATE:01.09.22** |

**AIM:**

To illustrate the first fit algorithm using c program.

**PROGRAM:**

#include<stdio.h>

#define max 25

void main(){

int frag[max],b[max],f[max],i,j,nb,nf,temp,highest=0;

static int bf[max],ff[max];int flag,flagn[max],fragi = 0,fragx = 0;

printf("\n\_\_\_First Fit\_\_\_\n");

printf("\nEnter the number of blocks:");

scanf("%d",&nb);

printf("Enter the number of Process:");

scanf("%d",&nf);

printf("\nEnter the size of the blocks:-\n");

for(i=1;i<=nb;i++) {

printf("Block %d:",i);

scanf("%d",&b[i]);

ff[i] = i;

}

printf("Enter the size of the Processes :-\n");

for(i=1;i<=nf;i++) {

printf("Process %d:",i);

scanf("%d",&f[i]);

}

int x = 1;

printf("\n\nProcess\_No\tProcess\_Size\tBlock\_No\tBlock\_Size\tFragment\n");

for(i=1;i<=nf;i++){

flag = 1;

for(j=x;j<=nb;j++){

if(f[i] <= b[j]){

flagn[j] = 1;

printf("%-15d\t%-15d\t%-15d\t%-15d\t",i, f[i],ff[j],b[j]);

b[j] = b[j] - f[i];

fragi = fragi + b[j];

printf("%-15d\n",b[j]);

break;

}

else{

flagn[j] = 0;

x = 1;

flag++;

} }

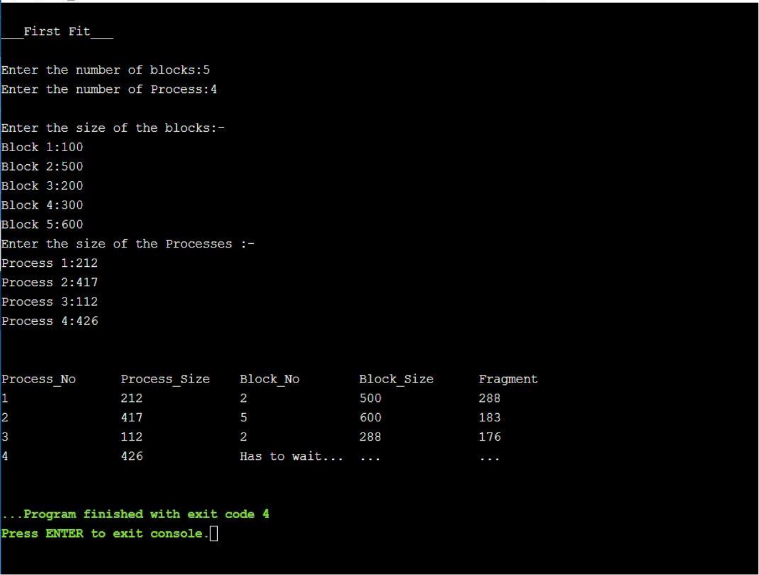
if(flag >nb)

printf("%-15d\t%-15d\t%-15s\t%-15s\t%-15s\n",i,f[i],"Has to wait...","...","...");

}

}

**OUTPUT:**

****

**RESULT:**

Thus the first fit algorithm using c program has been illustrated and executed successfully.

|  |  |
| --- | --- |
| **EX NO: 9B** | **BEST FIT ALGORITHM** |
| **DATE:01.09.22** |

**AIM:**

To illustrate thebest fit algorithm using c program.

**PROGRAM:**

#include<stdio.h>

#define max 25

void main()

{

int frag[max],b[max],f[max],i,j,nb,nf,temp,lowest=10000;

static int bf[max],ff[max],fragi = 0;

printf("\n\_\_\_Best Fit\_\_\_\n");

printf("\nEnter the number of blocks:");

scanf("%d",&nb);

printf("Enter the number of files:");

scanf("%d",&nf);

printf("\nEnter the size of the blocks:-\n");

for(i=1;i<=nb;i++) {

printf("Block %d:",i);

scanf("%d",&b[i]);

ff[i] = i;

}

printf("Enter the size of the Processes :-\n");

for(i=1;i<=nf;i++) {

printf("Process %d:",i);

scanf("%d",&f[i]);

}

int y,m,z,temp1,flag;

for(y=1;y<=nb;y++)

{

for(z=y;z<=nb;z++)

{

if(b[y]>b[z])

{

temp=b[y];

b[y]=b[z];

b[z]=temp;

temp1=ff[y];

ff[y]=ff[z];

ff[z]=temp1;

}

}

}

int flagn[max];

int fragx = 0;

printf("\n\nProcess\_No\tProcess\_Size\tBlock\_No\tBlock\_Size\tFragment\n");

for(i=1;i<=nf;i++)

{

flag = 1;

for(j=1;j<=nb;j++)

{

if(f[i] <= b[j]){

flagn[j] = 1;

printf("%-15d\t%-15d\t%-15d\t%-15d\t",i, f[i],ff[j],b[j]);

b[j] = b[j] - f[i];

fragi = fragi + b[j];

printf("%-15d\n",b[j]);

break;

}

else

{flagn[j] = 0;

flag++;

}

}

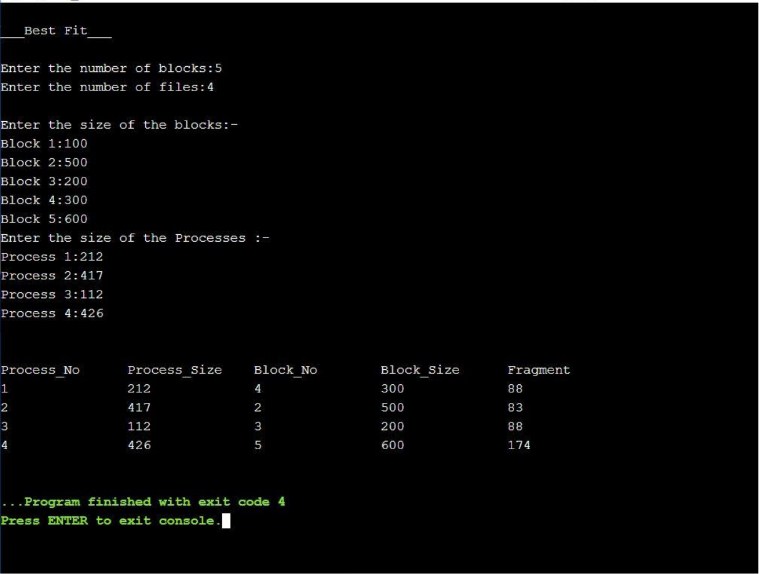
if(flag >nb)

printf("%-15d\t%-15d\t%-15s\t%-15s\t%-15s\n",i, f[i],"Has to wait..","...","...");

}

}

**OUTPUT:**

****

**RESULT:**

Thus the best fit algorithm using c program has been illustrated and executed successfully.

|  |  |
| --- | --- |
| **EX NO: 9C** | **WORST FIT ALGORITHM** |
| **DATE:01.09.22** |

**AIM:**

To illustrate theworst fit algorithm using c program.

**ALGORITHM:**

1. Start
2. Declare the variables
3. Input the no:of blocks and no: of processors from the users
4. Using for loop iput the memory block size and process size from user
5. Using for loop check

\*In which memory block the remaining is higher (i.e)the remaining memory when the process occupied should be greater than other.

\*If that it the memory block having higher remaining memory then assign the process to that memory block.

b[j]=b[j]-f[i];

fragi=fragi+b[j];

1. If no then repeat the step 5 for all processers
2. If there is insufficient space then display “Has to wait”
3. Display process no: , size , memory no; , size and the remaining size of memory
4. Stop.

**PROGRAM:**

#include<stdio.h>

#define max 25

void main()

{

int frag[max],b[max],f[max],i,j,nb,nf,temp,highest=0;

static int bf[max],ff[max];int flag,fragi = 0;

printf("\n\_\_\_Worst Fit\_\_\_\n");

printf("\nEnter the number of memory blocks:");

scanf("%d",&nb);

printf("Enter the number of Process:");

scanf("%d",&nf);

printf("\nEnter the size of the memory blocks:\n");

for(i=1;i<=nb;i++) {

printf("Block %d: ",i);

scanf("%d",&b[i]);

ff[i] = i;

}

printf("Enter the size of the Processes :\n");

for(i=1;i<=nf;i++) {

printf("Process %d: ",i);

scanf("%d",&f[i]);

}

int y,z,temp1;

for(y=1;y<=nb;y++)

{

for(z=y;z<=nb;z++)

{

if(b[y]<b[z])

{

temp=b[y];

b[y]=b[z];

b[z]=temp;

temp1=ff[y];

ff[y]=ff[z];

ff[z]=temp1;

}

}

}

int flagn[max];

int fragx = 0;

printf("\n\nProcess No\tProcess Size\tMemory No\tMemory Size\tRemaining\n");

for(i=1;i<=nf;i++)

{

flag = 1;

for(j=1;j<=nb;j++)

{

if(f[i] <= b[j]){

flagn[j] = 1;

printf("%-15d\t%-15d\t%-15d\t%-15d\t",i, f[i],ff[j],b[j]);

b[j] = b[j] - f[i];

fragi = fragi + b[j];

printf("%-15d\n",b[j]);

break;

}

else

{flagn[j] = 0;

flag++;

}

}

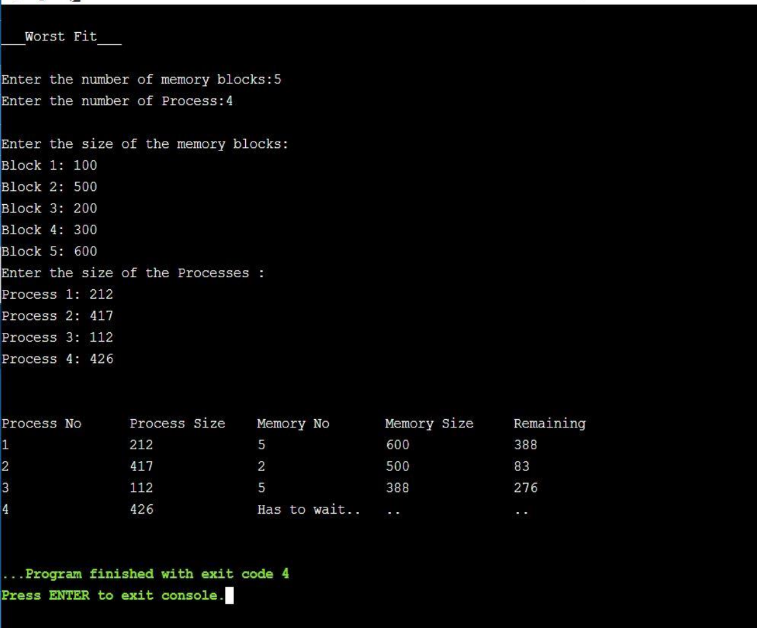
if(flag >nb)

printf("%-15d\t%-15d\t%-15s\t%-15s\t%-15s\n",i,f[i],"Has to wait..","..","..");

}

}

**OUTPUT:**

****

**RESULT:**

Thus the worst fit algorithm using c program has been illustrated and executed successfully.

|  |  |
| --- | --- |
| **EX.NO:10** | **BANKER’S ALGORITHM** |
| **DATE:01.09.22** |

**AIM:**

To illustrate Banker’s algorithm using C Program.

**ALGORITHM:**

1. Start
2. Declare the variables
3. Create various method to calculate
4. Print method,

        Using for loop,

               DISPLAY the required values to be printed

1. Safety method,

        In this method predict whether the resources can be allocated or not and ensure safety of the process.

1. Resource requested method,

        Create a resources request method is the user needs additional request then this method will be called

         And the process are checked for the new request and safety is also ensured.

1. Mainly used method is Banker’s method
2. Here in Banker’s method

          Calculate need matrix by

                              Maximum - Allocation

1. Create a accept method

          To input total no: of process and resources

          Also input the available resources from the user.

1. DISPLAY Allocation , Maximum requirement and Need matrix
2. From the main method call every method to perform the task

          Ask whether there is an resource request from the user,

                  if yes then use that request method

1. DISPLAY output
2. Stop

**PROGRAM:**

#include<stdio.h>

#include<stdlib.h>

void print(int x[][10],intn,int m){

inti,j;

for(i=0;i<n;i++){

printf("\n");

for(j=0;j<m;j++){

printf("%d\t",x[i][j]);

}

}

}

void res\_request(int A[10][10],int N[10][10],int AV[10][10],intpid,int m)

{

intreqmat[1][10];

inti;

printf("\n\_\_\_FOR ADDITINAL REQUEST:\_\_\_");

printf("\n Enter additional request :- \n");

for(i=0;i<m;i++){

printf(" Request for resource %d : ",i+1);

scanf("%d",&reqmat[0][i]);

}

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

if(reqmat[0][i] > N[pid][i]){

printf("\n The request can be granted.\n");

exit(0);

}

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

if(reqmat[0][i] > AV[0][i]){

printf("\n Resources unavailable.\n");

exit(0);

}

for(i=0;i<m;i++){

AV[0][i]-=reqmat[0][i];

A[pid][i]+=reqmat[0][i];

N[pid][i]-=reqmat[0][i];

}

}

intsafety(int A[][10],int N[][10],int AV[1][10],intn,intm,int a[]){

inti,j,k,x=0;

int F[10],W[1][10];

intpflag=0,flag=0;

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

F[i]=0;

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

W[0][i]=AV[0][i];

for(k=0;k<n;k++){

for(i=0;i<n;i++){

if(F[i] == 0){

flag=0;

for(j=0;j<m;j++){

if(N[i][j] > W[0][j])

flag=1;

}

if(flag == 0 && F[i] == 0){

for(j=0;j<m;j++)

W[0][j]+=A[i][j];

F[i]=1;

pflag++;

a[x++]=i;

}

}

}

if(pflag == n)

return 1;

}

return 0;

}

void accept(int A[][10],int N[][10],int M[10][10],int W[1][10],int \*n,int \*m){

inti,j;

printf("\n Enter total no. of processes : ");

scanf("%d",n);

printf("Enter total no. of resources : ");

scanf("%d",m);

for(i=0;i<\*n;i++){

printf("\n\tProcess %d\n",i+1);

for(j=0;j<\*m;j++){

printf(" Allocation for resource %d : ",j+1);

scanf("%d",&A[i][j]);

printf(" Maximum for resource %d : ",j+1);

scanf("%d",&M[i][j]);

}

}

printf("\n Available resources : \n");

for(i=0;i<\*m;i++){

printf(" Resource %d : ",i+1);

scanf("%d",&W[0][i]);

}

for(i=0;i<\*n;i++)

for(j=0;j<\*m;j++)

N[i][j]=M[i][j]-A[i][j];

printf("\n Allocation Matrix");

print(A,\*n,\*m);

printf("\n Maximum Requirement Matrix");

print(M,\*n,\*m);

printf("\n Need Matrix");

print(N,\*n,\*m);

}

intbanker(int A[][10],int N[][10],int W[1][10],intn,int m){

intj,i,a[10];

j=safety(A,N,W,n,m,a);

if(j != 0 ){

printf("\n\n");

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

printf(" P%d -->",a[i]);

printf("\n Hence the process sequence is safe...\n");

return 1;

}else{

printf("\n The process sequence is not safe...\n");

return 0;

}

}

intmain(){

int ret;

intA[10][10];

intM[10][10];

intN[10][10];

intW[1][10];

intn,m,pid,ch;

printf("\n\_\_\_BANKER'S ALGORITHM\_\_\_\n");

accept(A,N,M,W,&n,&m);

ret=banker(A,N,W,n,m);

if(ret !=0 ){

printf("\n Want make an additional request ? (1=Yes|0=No)");

scanf("%d",&ch);

if(ch == 1){

printf("\n Enter process no. : ");

scanf("%d",&pid);

res\_request(A,N,W,pid-1,m);

ret=banker(A,N,W,n,m);

if(ret == 0 )

exit(0);

}

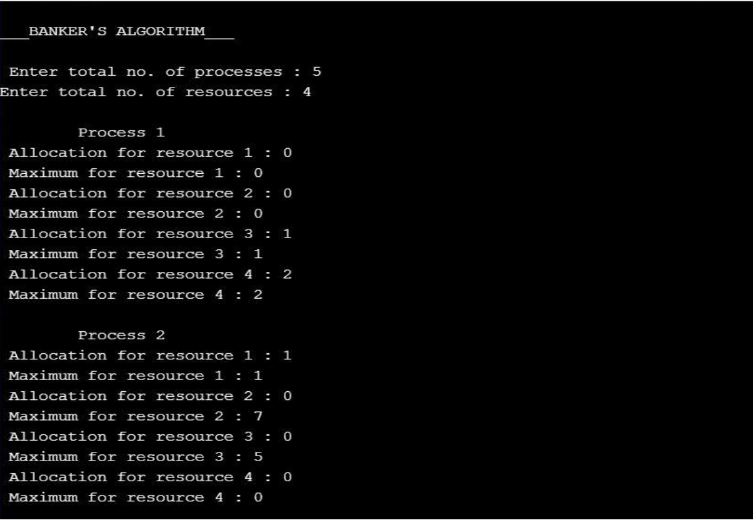
}else

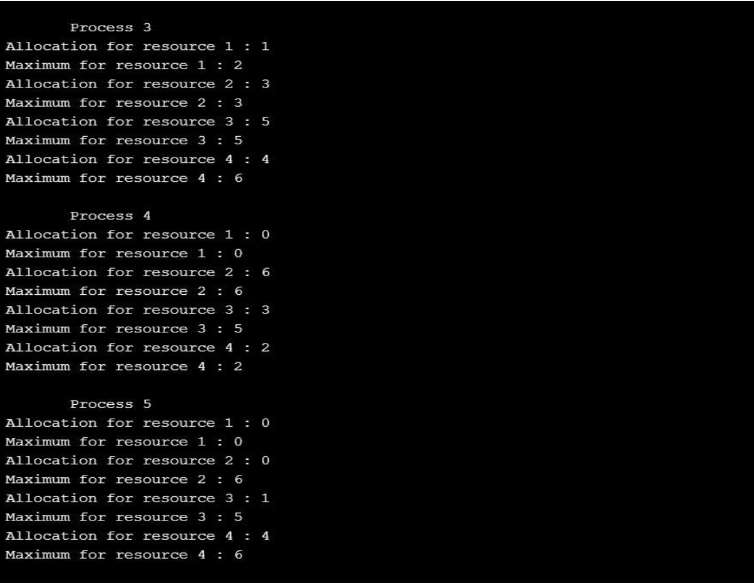
exit(0);

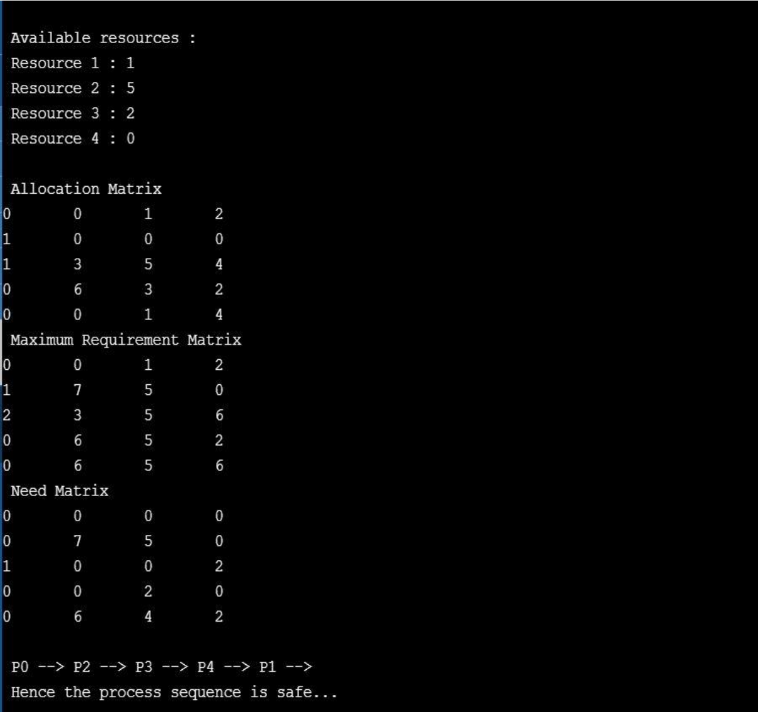
return 0;

}

**OUTPUT:**







  
  
**RESULT:**

Thus the Bankers algorithm using c program has been illustrated and executed successfully.

|  |  |
| --- | --- |
| **EX NO: 11** | **PAGE REPLACEMENT ALGORITHM** |
| **DATE:05.09.22** |

**FIFO PAGE REPLACEMENT ALGORITHM:**

**AIM:**

To illustrate the page replacement algorithm using C program

**ALGORITHM:**

1. START
2. Declare the variables required
3. Input the page numbers and page frames from the user using for loop
4. Check the need of replacement from old page to new page in memory using for loop

If frame[k]==a[i]

Initialize avail =1

If (avail==0)

Assign frame[j]=a[i]

j = (j+1) % num

Increment count by one

1. Form a queue to hold the pages
2. Get the page numbers and insert into the queue
3. Check for the page fault
4. Display the page numbers
5. Display the total numbers of page fault
6. STOP

**PROGRAM:**

#include<stdio.h>

intmain()

{

inti,j,n,a[50],frame[10],no,k,avail,count=0;

printf("\_\_\_FIFO PAGE REPLACEMENT ALGORITHM:\_\_\_");

printf("\n\nENTER THE NUMBER OF PAGES:\n");

scanf("%d",&n);

printf("\nENTER THE PAGE NUMBER :\n");

for(i=1;i<=n;i++)

scanf("%d",&a[i]);

printf("\nENTER THE NUMBER OF FRAMES :");

scanf("%d",&no);

for(i=0;i<no;i++)

frame[i]= -1;

j=0;

printf("Reg page\t Frames\n");

for(i=1;i<=n;i++)

{

printf("%d\t\t",a[i]);

avail=0;

for(k=0;k<no;k++)

if(frame[k]==a[i])

avail=1;

if (avail==0)

{

frame[j]=a[i];

j=(j+1)%no;

count++;

for(k=0;k<no;k++)

printf("%d\t",frame[k]);

}

printf("\n");

}

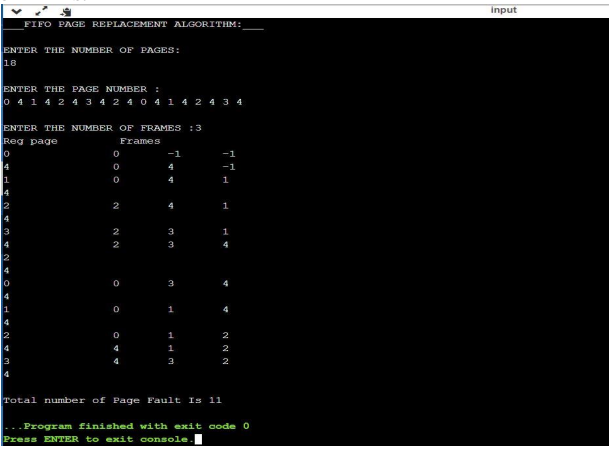
printf("\nTotal number of Page Fault Is %d",count);

return 0;

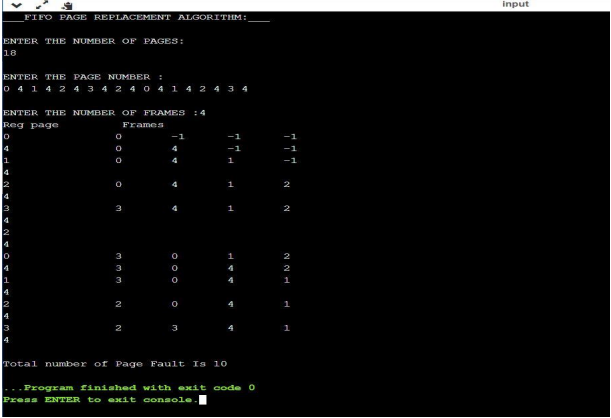
}

**OUTPUT:**

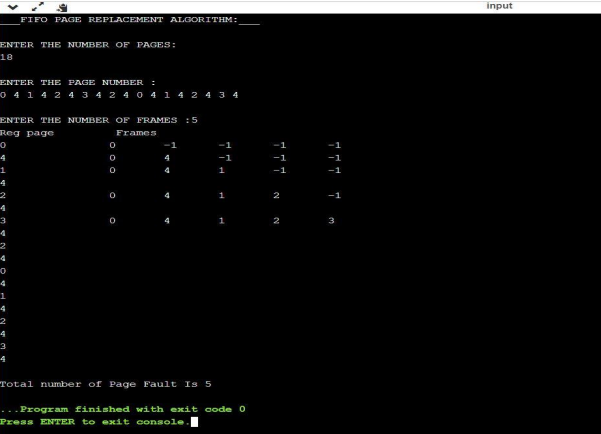
**3 FRAMES:**



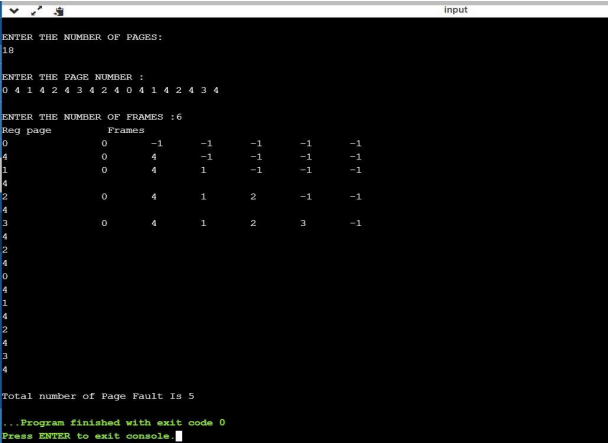
**4 FRAMES:**



**5 FRAMES:**



**6 FRAMES:**



**RESULT:**

Thus the page replacement algorithm using c program has been completed and executed successfully.

|  |  |
| --- | --- |
| **EX.NO:** **12** | **DISK SCHEDULING ALGORITHM** |
| **DATE:12.09.22** |

**AIM:**

To illustrate the given disk scheduling using c programme.

**ALGORITHM:**

**LOOK:**

1. START
2. Declare the required variables
3. Get the current head position from user
4. Input the number of requests
5. Get all the request and store it and also the upper bound
6. To check whether the head is moving towards upper bound or lower bound get the info from user
7. Using for loop perform

The head continue in moving in same direction until all the request in the direction are not finished

While moving in this direction calculate the absolute distance of track of the head

Increment total seek count with this distance

1. Display the need output
2. STOP

**SCAN:**

1. START
2. Declare the required variables
3. Input the head position and the total requests
4. Let direction represents the head is moving towards right or left
5. Using for loop perform

Calculate the absolute distance of the track from the head

Increment the total seek count with this distance

Currently serviced track will become the head now

1. Perform the operation until one end of distance is reached
2. If one end is reached then reverse the direction continue the looping process until are the tracks are serviced
3. Display the required output
4. STOP

**PROGRAM:**

**LOOK:**

#include<math.h>

#include<stdio.h>

intmain()

{

inti,n,j=0,k=0,x=0,l,req[50],mov=0,cp,ub,end, lower[50],upper[50], temp,a[50];

printf("\_\_\_DISK SCHEDULING\_\_\_(LOOK)\n\n");

printf("Enter the current head position: ");

scanf("%d",&cp);

printf("Enter the number of requests: ");

scanf("%d",&n);

printf("Enter the request order:\n");

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

{

scanf("%d",&req[i]);

}

printf("Enter the upper bound: ");

scanf("%d",&ub);

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

{

if(req[i]<cp)

{

lower[j]=req[i];

j++;

}

if(req[i]>cp)

{

upper[k]=req[i];

k++;

}

}

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

{

for(l=0;l<j

-1;l++)

{

if(lower[l]<lower[l+1])

{

temp=lower[l];

lower[l]=lower[l+1];

lower[l+1]=temp;

}

}

}

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

{

for(l=0;l<k

-1;l++)

{

if(upper[l]>upper[l+1])

{

temp=upper[l];

upper[l]=upper[l+1];

upper[l+1]=temp;

}

}

}

printf("Enter the end to which the head is moving 0 - for lower end and 1 - for upper

end\n");

scanf("%d",&end);

printf("--------------------------------------------------------------------------------------------------

\n");

printf("Solution:");

printf("\n\n Movement:\n");

switch(end)

{

case 0:

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

{

a[x]=lower[i];

x++;

}

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

{

a[x]=upper[i];

x++;

}

break;

case 1:

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

{

a[x]=upper[i];

x++;

}

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

{

a[x]=lower[i];

x++;

}

break;

}

mov=mov+abs(cp-a[0]);

printf("%d -> %d",cp,a[0]);

for(i=1;i<x;i++)

{

mov=mov+abs(a[i]-a[i-1]);

printf(" -> %d",a[i]);

}

printf("\n");

printf("Total distance in cylinders = %d cylinders\n",mov);

}

**SCAN:**

#include <stdio.h>

#include <stdlib.h>

#define LOW 0

#define HIGH 4299

intmain(){

intqueue[20];

int head, max, q\_size, temp, sum;

intdloc;

printf("\n\_\_\_DISK SCHEDULING\_\_\_(SCAN)\n\n");

printf("Enter head position:");

scanf("%d", &head);

printf("Enter no.of Disk Requests:");

scanf("%d", &q\_size);

printf("Enter the elements into disk queue:\n");

for(inti=0; i<q\_size; i++){

scanf("%d", &queue[i]);

}

queue[q\_size] = head;

q\_size++;

for(inti=0; i<q\_size;i++){

for(int j=i; j<q\_size; j++){

if(queue[i]>queue[j]){

temp = queue[i];

queue[i] = queue[j];

queue[j] = temp;

}

}

}

max = queue[q\_size-1];

for(inti=0; i<q\_size; i++){

if(head == queue[i]){

dloc = i;

break;

}

}

if(abs(head-LOW) <= abs(head-HIGH)){

for(int j=dloc; j>=0; j--){

}

for(int j=dloc+1; j<q\_size; j++){

}

} else {

for(int j=dloc+1; j<q\_size; j++){

}

for(int j=dloc; j>=0; j--){

}

}

sum = head + max;

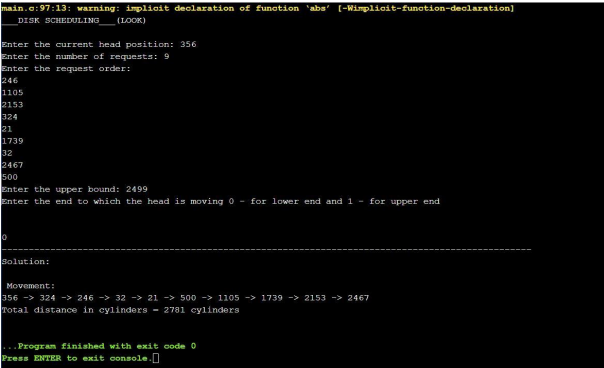
printf("\nTotal Seek Time: %d cylinders", sum);

return 0;

}

**OUTPUT:**

**LOOK:**



**SCAN:**



**RESULT:**

Thus the given disk scheduling using c program has been completed and executed successfully.

|  |  |
| --- | --- |
| **EX.NO:13** | **SEGMENTATION PROBLEM** |
| **DATE:12.09.22** |

**AIM:**

To illustrate the segmentation problem using C program.

**ALGORITHM:**

1. START
2. Declare the required variables
3. Input the number of segments, base values, length values, logical address(offset).
4. Perform the operation

If the logical address (offset)is less than equal to length

Then do

Sum up offset with base address

Else

DISPLAY segmentation fault

1. DISPLAY the required output
2. STOP

**PROGRAM:**

#include <stdio.h>

intmain()

{

intn,bv[10],lv[10],la[10],sum[10];

printf("\t\t\_\_\_SEGMENTATION\_\_\_\n\n");

printf("Enter the number of segments:");

scanf("%d",&n);

printf("Enter the base values:\n");

for(inti=0;i<n;i++)

{

scanf("%d",&bv[i]);

}

printf("\nEnter the length values:\n");

for(inti=0;i<n;i++)

{

scanf("%d",&lv[i]);

}

printf("\nEnter the logical addresses: \n");

for(inti=0;i<n;i++)

{

scanf("%d",&la[i]);

}

printf("\nSEGMENT\t BASE\tLENGTH\tOFF SET\n");

for(inti=0;i<n;i++)

{

printf(" %d\t %d\t %d\t %d\n",i+1,bv[i],lv[i],la[i]);

}

printf("\n\nPHYSICAL ADDRESS\n");

for (inti=0;i<n;i++)

{

if(la[i]<=lv[i])

{

sum[i]=bv[i]+la[i];

printf("%d\n",sum[i]);

}

else{

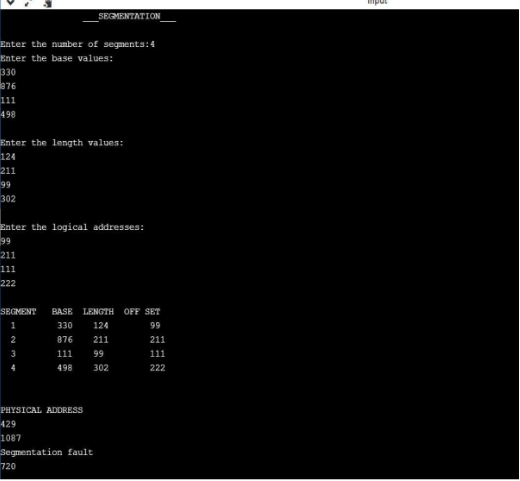
printf("Segmentation fault\n");

}

}

return 0;}

**OUTPUT:**



**RESULT:**

Thus, the segmentation problem using c program has been completed and executed successfully.