OS RECORD

|  |  |
| --- | --- |
| **EX NO:** | **INSTALLATION OF WINDOWS OPERATING SYSTEM CASE STUDY** |
| **DATE:** |

### AIM:

To know how to install the windows operating system.

### INTRODUCTION:

Each version of Microsoft Windows is installed on a computer using similar steps. While there are steps in the installation process that differ between versions of Windows, the following general steps and guidelines help you install Windows on your computer. The steps below are for all recent versions of Windows, including Windows 98, Windows ME, Windows 2000, Windows XP, Windows Vista, Windows 7, Windows 8, Windows 10, and Windows 11. These steps even work for earlier versions (e.g., Windows 95) as long as you use the disc version. The floppy diskette version is similar, but it requires additional steps.

### CHECK HARDWARE COMPATIBILITY

Before installing or upgrading Windows on your computer, check the hardware in the computer to make sure it's compatible with that version of Windows. Microsoft provides a Windows Compatible Products List for checking if the hardware in your computer is compatible with the chosen version of Windows.

If one or more pieces of hardware is not compatible with the chosen Windows version, we recommend replacing that hardware with compatible hardware or purchasing a new computer. Having compatible hardware in your computer helps ensure the Windows install or upgrade process is successful.

### GENUINE WINDOWS CD, DVD, OR USB THUMB DRIVE

First, you need a genuine copy of the Microsoft Windows operating system installation CD, DVD, or USB thumb drive. A genuine Windows product key is included with the installation disc, which is required to activate Windows after installation. If you have an OEM (original equipment manufacturer) computer,the Windows product key is often on the back or side of the computer.

If you have an OEM computer (e.g., Acer, Dell, HP, etc.), the computer will not have a genuine Windows CD, DVD, or USB thumb drive. Instead, you would reinstall Windows and the software using a hidden partition or a set of restore discs.

The steps mentioned on this page would still work, but you'd need a copy of Windows. You can borrow a friend's Windows disc, as long as it's the same version of Windows that came with the computer and have a product key.

### HOW TO ENTER AND EXIT THE BIOS OR CMOS SETUP

Every computer provides a way to enter the BIOS or CMOS setup, which lets you configure some basic settings for your computer and its hardware components. Below is a list of common methods for accessing and exiting your computer's BIOS or CMOS setup and recommendations if you're having trouble.

### ACCESS BIOS OR CMOS ON NEWER COMPUTERS:

Computers manufactured in the last few years allow you to enter the BIOS or CMOS setup by pressing one of the five keys listed below during the boot process.

* F1
* F2
* F10
* Delete or Del
* Esc
* If pressing F2 opens a diagnostics tool, your setup key is likely F10.
* F10 is also used for the boot menu. If pressing F10 opens a boot menu, your setup key is likely F2.

Setup keys are to be pressed as the computer is booting up. Most users see a message similar to the example below upon startup. Some older computers may also display a flashing block of text to indicate when to press F1 or F2.

Press F2 to enter BIOS setup. Once you have successfully entered the CMOS setup, a screen similar to the example below appears. Your CMOS setup may look different, depending on the manufacturer, but it should still share a lot of the same options and information.

### HOW DO I CHANGE AND SAVE CHANGES IN CMOS SETUP

Once in CMOS setup, the method for changing the settings often depends on the BIOS manufacturer. You may use the arrow keys and the Enter key to select categories and change their values. Some manufacturers may have you press the Page up and Page down keys to change the values.

If you're trying to change the clock, speed, or other settings and don't have the option available, the motherboard doesn't support it. If you believe it should be supported, you may need a BIOS update.

### HOW DO I SAVE THE CHANGES?

If any changes are made, you need to save those changes, which is usually done by pressing the F10 key on the keyboard. If F10 doesn't work, look at the bottom or top of the screen for the key that's used to save the settings.

### ACCESS BIOS OR CMOS ON OLDER COMPUTERS:

Unlike today's computers, older computers (before 1995) had numerous methods of entering the BIOS setup. Below is a listing of key sequences to press as the computer boots to enter the BIOS setup.

* Ctrl+Alt+Esc
* Ctrl+Alt+Insert
* Ctrl+Alt+Enter
* Ctrl+Alt+S
* Page Up
* Page Down

#### Acer BIOS:

If your Acer computer cannot boot or you want to restore the BIOS to its original settings, press and hold the F10 as you turn on the computer. While holding F10, two beeps should be heard to indicate the settings are restored.

### AMI BIOS:

Older AMI BIOS could be restored to bootable settings by pressing and holding

,Insert as the computer is booting.

#### BIOS or CMOS diskettes:

Early 486, 386, and 286 computers required a floppy disk to enter the BIOS setup. These diskettes may be called ICU, BBU, or SCU disks. Because these diskettes are unique to your computer manufacturer, you must obtain the diskettes from them. See the computer manufacturers list for contact information.

#### Access BIOS on early IBM computers:

Some early IBM computers require you to press and hold both mouse buttons as the computer boots to enter the BIOS setup.

#### Additional suggestions for accessing BIOS or CMOS:

Finally, if none of the above suggestions allow access to the setup, try generating a stuck key error, which gives an option to enter the BIOS or CMOS setup. To do this, press and hold any key on the keyboard, and do not let go (you may get several beeps as you are doing this). Keep pressing the key until the computer stops booting, and you have the option to enter setup. If this does not work, make sure your keyboard is working.

### HOW TO EXIT THE BIOS OR CMOS:

There are several ways to exit the BIOS or CMOS setup depending on the computer's type. The most common methods include the following.

* Press the Esc key to exit without saving any changes.
* Press the F10 or F12 key to save changes and exit.
* Access the **Exit** or **Save & Exit** tab in setup and select the **Exit** or **Save and Exit**

option.

If you have trouble exiting the BIOS or CMOS setup, you can try the following methods to fix the problem.

* Press the F9 key to load default settings and press F10 to save and exit.

Access the **Exit** or **Save & Exit** tab in setup, select the **Load Optimized Defaults** option, select **Yes**, and press Enter. Any changes made are reverted, and the BIOS or CMOS is set back to default settings.

* Turn off the computer (use only as a last resort and with caution).

### INSTALLING OR UPGRADING WINDOWS:

To start the Windows install or upgrade process, you need to configure your computer to boot from a CD or DVD before booting to the hard drive. Changing the boot process forces the computer to look for the Windows installation disc before booting from the hard drive.

1. Open the CMOS setup.
2. Change the computer's boot order. Set the CD, DVD, or disc drive as the first boot device if you are trying to boot from a disc. Or, set the first boot device to your USB drive if you're trying to boot from a USB thumb drive. If the drive is not shown, keep

the disc is inserted and reboot the computer. With the disc in the drive, BIOS should recognize and include it in the list.

1. Save the settings change and exit BIOS.

Note: Once you have updated the boot order, you can begin the Windows installation process.

1. Place the Windows disc in the CD/DVD drive or USB thumb drive into the back of the computer.
2. Turn on or restart the computer. As the computer starts up, it should detect the installation disc or drive and show a message similar to *Press any key to boot from CD*. Press any key on the keyboard to have the computer boot from the Windows disc or drive.
3. After the Windows install begins, there are several prompts that you need to answer. Select either **Yes** or the appropriate option to install Windows.
4. When asked which partition to install Windows onto, select the main partition, usually the C: drive or one labeled "Unallocated partition". If upgrading Windows, select the existing installation of Windows on the hard drive.
5. You may be asked if you want to erase all contents on the hard drive, then install Windows. We recommend you choose this option, as it also formats the hard drive to allow the Windows operating system to be installed.
6. The computer may need to restart several times during the Windows install process. The restarts are normal and if prompted to restart, select the **Yes** option.
7. When the install process is nearly complete, the Windows configuration option screens are shown. On these screens, you may be asked to select the time zone you live in, your preferred language, and the account's name you use to access Windows. Select the appropriate options and enter the appropriate information on each configuration screen.
8. The Windows install process is completed when the computer prompts you to log in or when it loads into Windows.

### RESULT:

Thus the study exercise for installing windows operating system is studied successfully.

|  |  |
| --- | --- |
| **EX NO:** | **BASIC UNIX COMMANDS** |
| **DATE:** |

### AIM:

To implement the Basic Unix commands.

**COMMANDS:**

* 1. **GENERAL PURPOSE COMMANDS:**

1. **THE DATE COMMAND:**

The **date** command can also be used with following format.

|  |  |  |
| --- | --- | --- |
| + %m | To display only month | $ date + %m |
| + %h | To display month name | $ date + %h |
| + %d | To display day of month | $ date + %d |
| + %y | To display last two digits of the year | $ date + %y |
| + %H | To display Hours | $ date + %H |
| + %M | To display Minutes | $ date + %M |
| + %S | To display Seconds | $ date + %S |

[exam@fosslab ~]$ date

Sat Feb 14 11:48:18 IST 2015

#### THE echo COMMAND:

[exam@fosslab ~]$ echo learning unix is intresting learning unix is intresting

#### THE Who COMMAND:

[exam@fosslab ~]$ who

exam pts/0 2015-02-14 11:48 (192.168.8.5)

exam20 pts/0 2015-02-14 11:48 (192.168.8.6)

#### THE Who am i COMMAND:

[exam@fosslab ~]$ who am i

exam pts/0 2015-02-14 11:48 (192.168.8.5)

### THE UNIX CALENDER:

#### Cal:

[exam@fosslab ~]$ cal 2 2015

February 2015

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 |

#### THE Finger COMMAND:

[exam@fosslab ~]$ finger exam25 Login: exam Name:

Directory: /home/exam Shell: /bin/bash

On since Sat Feb 14 11:48 (IST) on pts/0 from 192.168.8.5 No mail. No Plan.

#### THE id COMMAND:

[exam@fosslab ~]$ id

uid=662(exam) gid=662(exam) groups=662(exam)

#### THE tty COMMAND:

[exam@fosslab ~]$ tty

/dev/pts/0

### VIEW THE CONTENT:

[exam@fosslab ~]$ cat test

welcome to operating system. it is an interesting subject.

### CLEARING THE SCREEN

[exam@fosslab student ~]$ tput clear

### DIRECTORY COMMANDS

**1.CREATE A DIRECTORY:**

[exam@fosslab ~]$ mkdir student [exam@fosslab ~]$ cd student [exam@fosslab student]$

### 2 CURRENT WORKING DIRECTORY:

[exam@fosslab ~]$ pwd

/home/exam

### REMOVING A DIRECTORY:

[exam@fosslab student]$ rmdir student [exam@fosslab ~]$

### LISTING THE FILES AND DIRECTORIES:

#### ls:

[exam@fosslab student~]$ ls

a.out data program public\_html share stud25 student test.c test1

### CHANGING THE WORKING DIRECTORY:

Cd :Change directory

Pwd: to view the full path of current directory [exam@fosslab ~]$ pwd

/home/exam/

[exam@fosslab ~]$ mkdir student [exam@fosslab ~]$ cd student [exam@fosslab student]$ pwd

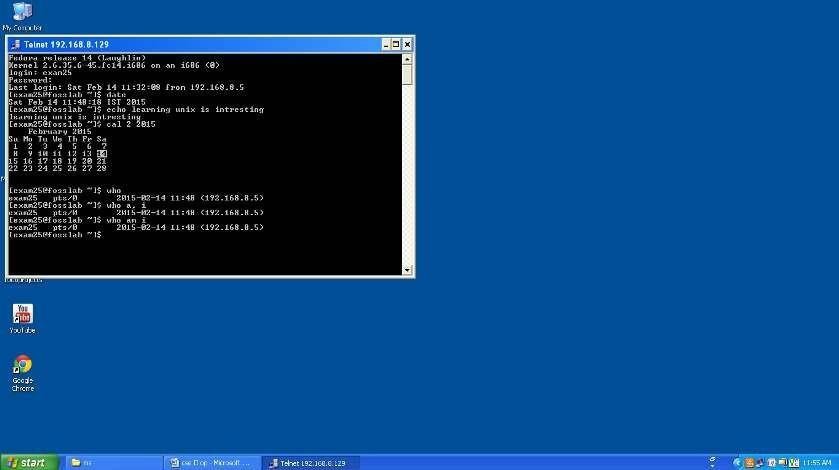
/home/exam/student/

### THE PATH

[exam@fosslab student~]$ echo $PATH

/usr/lib/qt3.3/bin:/usr/lib/mpich2/bin:/usr/lib/ccache:/usr/local/bin:/b in:/usr/bin:/usr/local/sbin:/usr/sbin:/sbin:/usr/lib/alliance/bin:/usr/lib exec/sdcc:/opt/ns2/bin:/opt/ns2/tcl8.4.14/unix:/opt/ns2/tk8.4.14/unix

:/opt/ns2/ns-2.34/:/opt/ ns2/nam-1.14/:/home/exam25/bin



### CHANGE THE PASSWORD:

[exam@fosslab ~]$ passwd

(current) UNIX PASSWORD: \*\*\*\*\*\*\*\* New Password: \*\*\*\*\*\*\*\* Re-enter Password: \*\*\*\*\*\*\*\*

**$**

### FILE HANDLING COMMANDS

1. **THE CAT COMMAND:**

[exam@fosslab ~]$ cat>test

welcome to operating system. it is an interesting subject.

### COPYING THE FILE:

#### Cp:

[exam@fosslab student~]$ cat test

welcome to operating system. it is an interesting subject. [exam@fosslab student ~]$ cat test1

the basic unix commands are cat, pwd, mkdir, rmdir, cd, path,clear,cp,rm, mv,ls,wc. [exam@fosslab student~]$ cp test test1

[exam@fosslab student ~]$cat test1

welcome to operating system. it is an interesting subject.

### REMOVING A FILE:

#### Rm:

[exam@fosslab student ~]$ rm test1 [exam@fosslab student ~]$ cat test1 Cat :test1: No such file or directory

### MOVING A FILE:

[exam@fosslab student~]$ cat >test1

the basic unix commands are cat,pwd,clear,cp,mv,rm,mv,test..^C

#### mv:

[exam@fosslab student ~]$ mv test test1 [exam@fosslab student ~]$ cat test 1

the basic unix commands are cat, pwd, mkdir, rmdir, cd, path,clear,cp,rm, mv,ls,wc.

### DIRECTING OUTPUT TO A FILE:

[exam@fosslab student~]$ ls>test [exam@fosslab student ~]$ cat test a.out

data mylist program

public\_html share

[exam@fosslab ~]$

### COUNTING NUMBER OF WORDS IN A FILE:

#### wc:

[exam@fosslab ~]$ wc test 10 10 70 test

### THE FILE COMMAND

[exam@fosslab ~]$ file test test: ASCII Pascal program text [exam@fosslab ~]$ cat test

a.out data mylist program

public\_html share

### CHANGING THE FILE PERMISSION:

#### Chmod:

[exam@fosslab ~]$ chmod u-wx test [exam@fosslab ~]$ cat > test

-bash: test: Permission denied

### FILTER COMMANDS

#### SORTING THE CONTENTS: (sort)

[exam@fosslab ~]$ sort test1 a.out

data mylist program

public\_html share

#### THE uniq COMMAND:

[exam@fosslab ~]$ cat > dept.lst 01 accounts 3977

* 1. accounts 3977
  2. admin 1707
  3. marketing 39

06 sales 1008^C [exam@fosslab ~]$ uniq dept.lst 01 accounts 3977

1. admin 1707
2. marketing 39
3. personel 77
4. production 1739

### ADDING LINE NUMBERS:

#### nl

[exam@fosslab ~]$ nl test1

1. a.out
2. data
3. mylist
4. program
5. public\_html

### SELECTING FIELDS FROM A LINE:

#### cut

[exam@fosslab ~]$ cat >std Aswini

Bharathi Charu Deepa^C

[exam@fosslab ~]$ cut -c1 std A

B C D

#### THE more COMMAND:

[exam@fosslab ~]$ more test1 a.out

data mylist program

public\_html

### PASTING FILES:

[exam@fosslab ~]$ paste std Aswini

Bharathi Charu Deepa

### COMPARING FILES:

#### cmp

[exam@fosslab ~]$ cmp test1 std test1 std differ: byte 1, line 1

#### THE mesg COMMAND:

**mesg**

[exam@fosslab ~]$ mesg exam25 Usage: mesg [y|n] [exam@fosslab ~]$ y

Message from [exam25@fosslab.linuxpert.in](mailto:exam25@fosslab.linuxpert.in) on pts/0 at 12:34 ... hiiiii ... study well for your exams

hiiiii ... study well for your exams

#### THE write COMMAND:

**write**

[exam@fosslab ~]$ write exam25

Message from [exam25@fosslab.linuxpert.in](mailto:exam25@fosslab.linuxpert.in) on pts/0 at 12:30 ... Hiiii

Hiiii

### SENDING MESSAGE TO ALL THE USERS:

#### wall:

[exam@fosslab ~]$ wall os laboratory lab records

Broadcast message from [exam25@fosslab.linuxpert.in](mailto:exam25@fosslab.linuxpert.in) (pts/0) (Wed Jan 21 13:21:os laboratory lab records

### SENDING MAIL TO USERS:

mail:

[exam@fosslab ~]$ mail user2 Subject: about operating systems

THE OPERATING SYSTEMS BOOk is a "practice of some materials to gain knowledge" EOT

#### THE reply COMMAND:

**reply**

reply exam25

Thanks For Giving A Mail

### NO LOGGING OUT:

[exam@fosslab ~]$ std.lst test1.lst & [1] 4663

#### THE nohup COMMAND:

[exam@fosslab ~]$ nohup test1.lst &

[1] 4685

#### Execution of a Job With Low Priority: nice

[exam@fosslab ~]$ nice wc -l std &

[2] 4721

[1] Exit 127 nohup test1.lst [exam@fosslab ~]$ 4 std

#### THE at COMMAND:

[exam@fosslab ~]$ at 12.54pm at> today at evening 4pm

at> 21.30 tue next at 9.20 at> 2pm apr3 next 3rd

#### THE sleep COMMAND:

[exam@fosslab ~]$ sleep 1

### KILLING PROCESSES WITH SIGNALS

#### kill

[exam@fosslab ~]$ kill 4921

### RESULT:

Thus the basic UNIX commands were executed successfully.

# SHELL PROGRAMMING

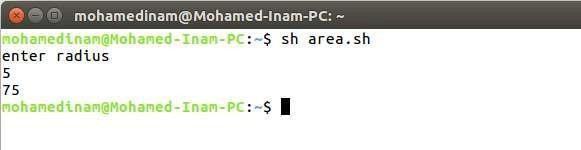
**AREA OF THE CIRCLE**

##### SHELL SCRIPT:

echo "enter radius" read r

val=`expr 3 \\* $r \\* $r` echo "$val"

**OUTPUT:**

****

# SHELL PROGRAMMING

##### BIGGEST OF THREE NUMBERS

**SHELL SCRIPT:**

echo "Enter three numbers"

read a b c

if [ $a -gt $b ] && [ $a -gt $c ]; then

echo "a is big"

elif [ $b -gt $c ]; then

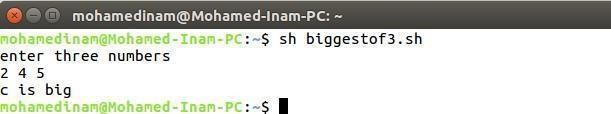
echo "b is big"

else

echo "c is big"

fi

**OUTPUT:**

****

# SHELL PROGRAMMING

##### NATURAL NUMBERS UPTO ‘N’

**SHELL SCRIPT:**

echo "Enter upper limit"

read n

i=0

while [ $i -lt $n ]

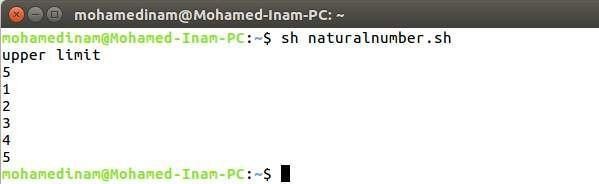
do

echo $i

i=`expr $i + 1`

done

**OUTPUT:**



# SHELL PROGRAMMING

##### ARITHMETIC OPERATIONS

**SHELL SCRIPT:**

echo "Enter two numbers"

read a b

echo "1. Add 2. Sub 3. Mul 4. Div 5. Exit"

read op

case $op in

1) c=`expr $a + $b` ;;

2) c=`expr $a - $b` ;;

3) c=`expr $a \\* $b` ;; # The asterisk needs to be escaped

4) c=`expr $a / $b` ;; # Ensure division works correctly

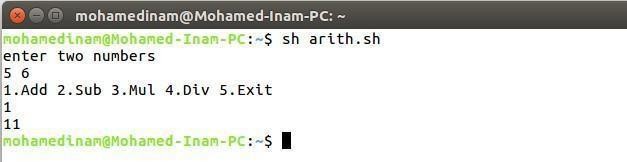
5) exit ;;

\*) echo "Invalid option" ;; # Added for handling invalid options

esac

echo $c

**OUTPUT:**

****

# SHELL PROGRAMMING

##### FACTORIAL OF A NUMBER

**SHELL SCRIPT:**

echo "enter number"

read n

i=1

f=1

while [ $i -le $n ]

do

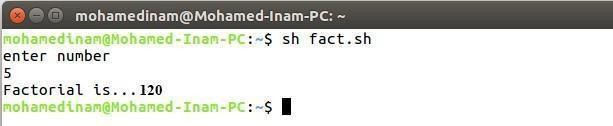
f=`expr $f \\* $i`

i=`expr $i + 1`

done

echo "Factorial is...$f"

**OUTPUT:**

****

# SYSTEM CALLS OF UNIX OPERATING SYSTEM

**CLOSE()**

##### PROGRAM:

#include<stdio.h>

#include<fcntl.h>

#include<unistd.h>

#include<stdlib.h>

int main()

{

int fd1 = open("foo.txt", O\_RDONLY);

if (fd1 < 0)

{

perror("c1");

exit(1);

}

printf("opened the fd = %d\n", fd1);

if (close(fd1) < 0)

{

perror("c1");

exit(1);

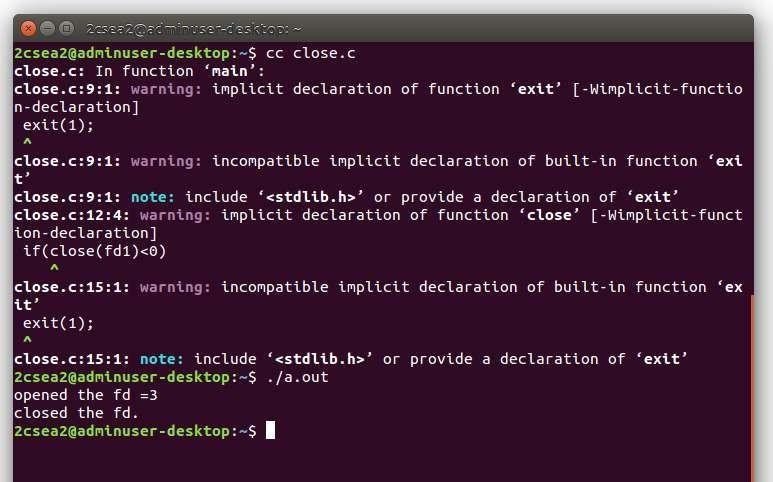
}

printf("closed the fd.\n");

return 0;

}

**OUTPUT:**

****

# SYSTEM CALLS OF UNIX OPERATING SYSTEM

**GETPID()**

##### PROGRAM:

**Example.c** #include <stdio.h> #include <unistd.h> #include <stdlib.h>

int main(int argc, char \*argv[]) {

printf("PID of example.c = %d\n", getpid());

char \*args[] = {"hello", "c", "programming", NULL};

if (execv("./hello", args) == -1) { perror("execv failed"); exit(EXIT\_FAILURE);

}

printf("BACK TO EXAMPLE.C\n");

return 0;

}

###### Hello.c

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

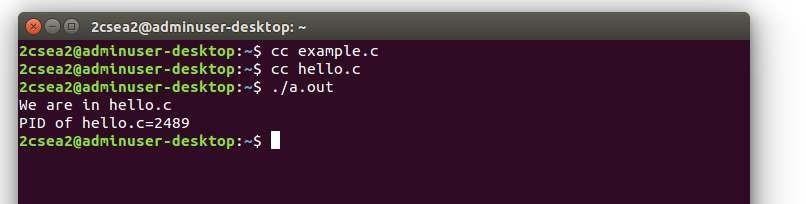
int main (int argc, char \*argv[])

{

printf("We are in hello.c\n"); printf("PID of hello.c=%d\n",getpid()); return 0;

}

**OUTPUT:**

****

# SYSTEM CALLS OF UNIX OPERATING SYSTEM

##### FORK()

**PROGRAM:**

#include <stdio.h> #include <unistd.h>

int main() { int id;

printf("hello world!\n"); id = fork();

if (id > 0) {

printf("this is parent section [process id: %d].\n", getpid());

}

else if (id == 0) {

printf("fork created [process id: %d].\n", getpid()); printf("fork parent process id: %d.\n", getppid());

}

else {

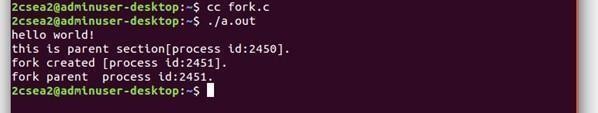
printf("fork creation failed!!\n");

}

return 0;

}

**OUTPUT:**

****

# SYSTEM CALLS OF UNIX OPERATING SYSTEM

**EXIT()**

##### PROGRAM:

#include<stdio.h> #include<stdlib.h> #include<sys/wait.h> #include<unistd.h>

int main() { pid\_t cpid;

if (fork() == 0) {

exit(0); // terminate child

} else {

cpid = wait(NULL); // parent waits for child to terminate printf("Parent pid = %d\n", getpid());

printf("Child pid = %d\n", cpid);

}

return 0;

}

##### OUTPUT:

Parent pid = 12345678 Child pid = 89546848

# SYSTEM CALLS OF UNIX OPERATING SYSTEM

**WAIT()**

##### PROGRAM:

#include<stdio.h> #include<sys/wait.h> #include<unistd.h>

int main() {

if (fork() == 0) {

printf("HC: hello from child\n");

} else {

printf("HP: hello from parent\n"); wait(NULL);

printf("CT: child has terminated\n");

}

printf("Bye\n"); return 0;

}

##### OUTPUT:

HP: hello from parent HC: hello from child HC: Bye

CT: child has terminated

# CPU SCHEDULING ALGORITHMS

##### FIRST COME FIRST SERVE (FCFS)

**PROGRAM:**

#include<stdio.h>

void main() {

int i, n, sum, wt, tat, twt, ttat;

int t[10]; // Array to store burst times of the processes float awt, atat;

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

// Input burst times for each process for (i = 0; i < n; i++) {

printf("\nEnter burst time for process %d: ", i + 1); scanf("%d", &t[i]);

}

printf("\nFIRST COME FIRST SERVE SCHEDULING");

printf("\nProcess ID\tWaiting Time\tTurnaround Time\n");

sum = 0;

twt = 0;

ttat = t[0]; // Turnaround time for the first process is just its burst time

// Process the first process printf("1\t\t0\t\t%d\n", t[0]);

// Calculate waiting time and turnaround time for other processes for (i = 1; i < n; i++) {

sum += t[i - 1]; // Sum of burst times of all previous processes wt = sum; // Waiting time for current process

tat = sum + t[i]; // Turnaround time for current process twt += wt; // Accumulate total waiting time

ttat += tat; // Accumulate total turnaround time printf("%d\t\t%d\t\t%d\n", i + 1, wt, tat);

}

// Calculate average waiting time and average turnaround time awt = (float)twt / n;

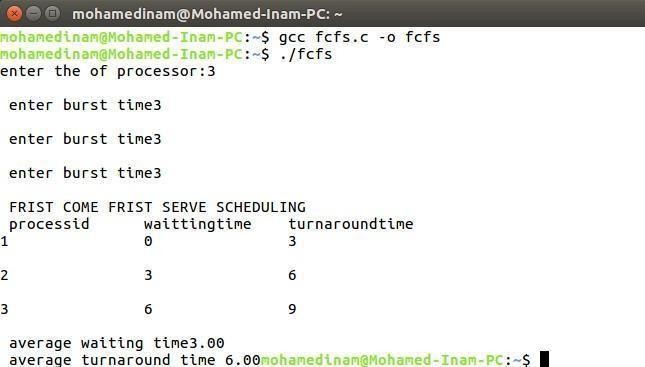
atat = (float)ttat / n;

// Print the average times

printf("\nAverage Waiting Time: %.2f", awt); printf("\nAverage Turnaround Time: %.2f", atat);

}

**OUTPUT:**

****

# CPU SCHEDULING ALGORITHMS

##### SHORTEST JOB FIRST (SJF)

**PROGRAM:**

#include<stdio.h>

void main() {

int i, j, n, sum, wt[10], tt[10], twt = 0, ttat = 0;

int t[10], p[10]; // t[] for burst times, p[] for process IDs float awt, atat;

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

// Input burst times for processes for (i = 0; i < n; i++) {

printf("\nEnter burst time for process %d: ", i + 1); scanf("%d", &t[i]);

p[i] = i; // Assign process IDs

}

// Sorting the processes based on burst time using selection sort for (i = 0; i < n - 1; i++) {

for (j = i + 1; j < n; j++) { if (t[i] > t[j]) {

// Swap burst times int temp = t[i];

t[i] = t[j]; t[j] = temp;

// Swap process IDs to maintain correct order temp = p[i];

p[i] = p[j]; p[j] = temp;

}

}

}

// Printing header for the output

printf("\nSHORTEST JOB FIRST SCHEDULING\n");

printf("\nProcess ID\tBurst Time\tWaiting Time\tTurnaround Time\n");

wt[0] = 0; // First process has no waiting time sum = 0;

// Calculate waiting times for each process for (i = 1; i < n; i++) {

sum += t[i - 1]; // Sum of burst times of previous processes wt[i] = sum; // Waiting time for current process

}

// Calculate turnaround times for each process for (i = 0; i < n; i++) {

tt[i] = t[i] + wt[i]; // Turnaround time = burst time + waiting time

}

// Print the results for each process for (i = 0; i < n; i++) {

printf("%5d\t\t%5d\t\t%5d\t\t%5d\n", p[i] + 1, t[i], wt[i], tt[i]);

}

// Calculate total waiting time and total turnaround time for (i = 0; i < n; i++) {

twt += wt[i]; ttat += tt[i];

}

// Calculate average waiting time and average turnaround time awt = (float)twt / n;

atat = (float)ttat / n;

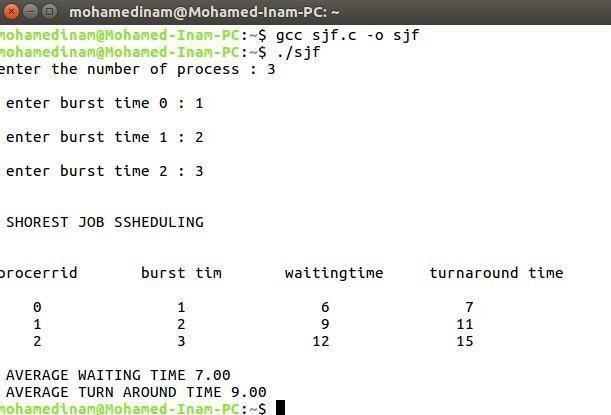
// Print average times

printf("\nAverage Waiting Time: %.2f", awt);

printf("\nAverage Turnaround Time: %.2f", atat);

}

**OUTPUT:**

****

# CPU SCHEDULING ALGORITHMS

##### PRIORITY – SCHEDULING ALGORITHM

**PROGRAM:**

#include<stdio.h> void main()

{

int i, j, n, t, twt = 0, ttat = 0;

int tat[10], wt[10], bt[10], pid[10], pr[10]; float awt, atat;

printf("\n PRIORITY SCHEDULING\n");

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

// Input burst times and priorities for each process for (i = 0; i < n; i++) {

pid[i] = i; // Assign Process IDs printf("Enter the Burst time of Pid %d: ", i); scanf("%d", &bt[i]);

printf("Enter the Priority of Pid %d: ", i); scanf("%d", &pr[i]);

}

// Sorting based on Priority using Selection Sort for (i = 0; i < n; i++) {

for (j = i + 1; j < n; j++) { if (pr[i] > pr[j]) {

// Swap priorities t = pr[i];

pr[i] = pr[j]; pr[j] = t;

// Swap burst times t = bt[i];

bt[i] = bt[j]; bt[j] = t;

// Swap process IDs t = pid[i];

pid[i] = pid[j]; pid[j] = t;

}

}

}

// Calculate waiting times and turnaround times wt[0] = 0; // The first process has no waiting time

tat[0] = bt[0]; // Turnaround time for first process is its burst time

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

wt[i] = wt[i - 1] + bt[i - 1]; // Waiting time is sum of burst times of previous processes tat[i] = wt[i] + bt[i]; // Turnaround time = waiting time + burst time

}

// Printing the results

printf("\nPid \t Priority \t Burst Time \t Waiting Time \t Turnaround Time\n"); for (i = 0; i < n; i++) {

printf("\n %d \t\t %d \t\t %d \t\t %d \t\t %d", pid[i], pr[i], bt[i], wt[i], tat[i]);

}

// Calculate total waiting time and total turnaround time for (i = 0; i < n; i++) {

twt += wt[i]; ttat += tat[i];

}

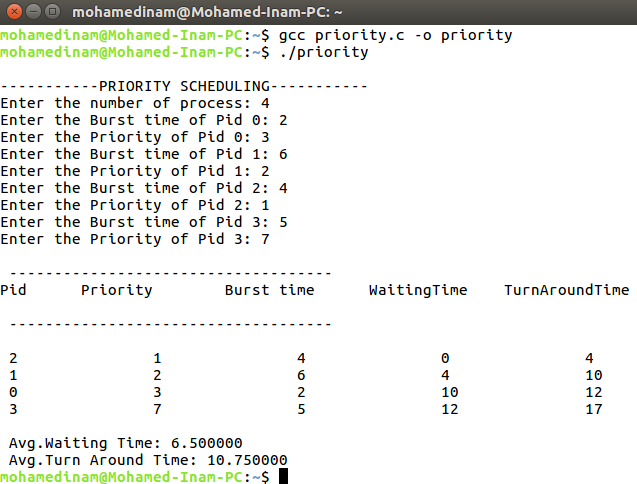
// Calculate average waiting time and average turnaround time awt = (float)twt / n;

atat = (float)ttat / n;

// Printing average times

printf("\n\n Avg.Waiting Time: %.2f\n Avg.Turn Around Time: %.2f\n", awt, atat);

}**OUTPUT:**



# CPU SCHEDULING ALGORITHMS

##### ROUND ROBIN – SCHEDULING ALGORITHM

**PROGRAM:**

#include <stdio.h> void main() {

int ts, pid[10], need[10], wt[10], tat[10], i, j, n, n1; int bt[10], flag[10], ttwt = 0, ttat = 0;

float awt, atat;

printf("\nROUND ROBIN SCHEDULING\n");

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

n1 = n;

printf("Enter the Time Slice: "); scanf("%d", &ts);

// Input Burst Times and Process IDs for (i = 1; i <= n; i++) {

printf("\nEnter Process ID for Process %d: ", i); scanf("%d", &pid[i]);

printf("Enter Burst Time for Process %d: ", i); scanf("%d", &bt[i]);

need[i] = bt[i]; // Remaining burst time initialized to the burst time flag[i] = 1; // Process is still in the ready queue

wt[i] = 0; // Initialize waiting time to 0

}

// Round Robin Scheduling while (1) {

int done = 1; // Flag to check if all processes are completed

for (i = 1; i <= n; i++) { if (need[i] > 0) {

done = 0; // If any process is pending, set done to 0

if (need[i] > ts) {

need[i] -= ts; // Reduce the remaining time by time slice for (j = 1; j <= n; j++) {

if (i != j && need[j] > 0) {

wt[j] += ts; // Increment waiting time for other processes

}

}

} else {

for (j = 1; j <= n; j++) {

if (i != j && need[j] > 0) {

wt[j] += need[i]; // Add the remaining burst time to waiting times

}

}

need[i] = 0; // Mark this process as completed

}

}

}

if (done) {

break; // All processes are completed, exit the loop

}

}

// Calculate Turnaround Time and Total Waiting Time for (i = 1; i <= n1; i++) {

tat[i] = bt[i] + wt[i]; // Turnaround time = burst time + waiting time ttwt += wt[i]; // Total waiting time

ttat += tat[i]; // Total turnaround time

}

awt = (float)ttwt / n1; // Average Waiting Time atat = (float)ttat / n1; // Average Turnaround Time

// Output the results

printf("\nROUND ROBIN SCHEDULING ALGORITHM\n");

printf("\nProcess \t Process ID \t Burst Time \t Waiting Time \t Turnaround Time\n"); for (i = 1; i <= n1; i++) {

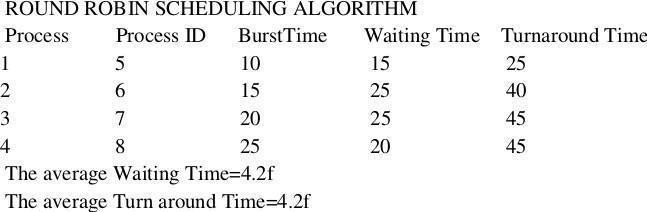
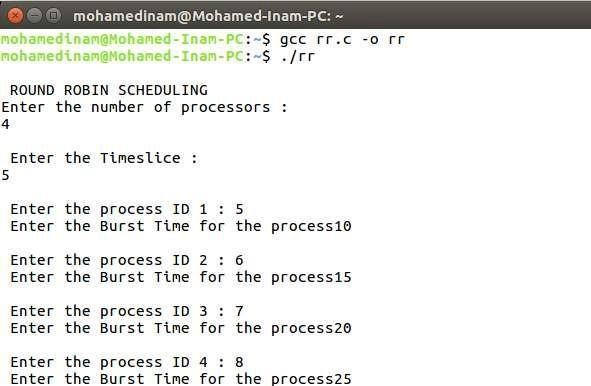
printf("\n%5d \t %5d \t\t %5d \t\t %5d \t\t %5d\n", i, pid[i], bt[i], wt[i], tat[i]);

}

printf("\nThe average Waiting Time = %.2f", awt); printf("\nThe average Turnaround Time = %.2f", atat);

}

**OUTPUT:**

****

# IPC-SHARED MEMORY

##### PROGRAM:

Shared Memory For Writer Process

#include <iostream> #include <sys/ipc.h> #include <sys/shm.h> #include <stdio.h> using namespace std; int main() {

key\_t key = ftok("shmfile", 65); // Generate a unique key

int shmid = shmget(key, 1024, 0666 | IPC\_CREAT); // Create shared memory segment char \*str = (char \*)shmat(shmid, (void \*)0, 0); // Attach shared memory

printf("Write Data : ");

fgets(str, 1024, stdin); // Read input into shared memory printf("Data written in memory: %s\n", str); // Output the stored data shmdt(str); // Detach shared memory

return 0;

}

Shared Memory For Reader Process

#include <iostream> #include <sys/ipc.h> #include <sys/shm.h> #include <stdio.h> using namespace std; int main() {

key\_t key = ftok("shmfile", 65); // Generate a unique key

int shmid = shmget(key, 1024, 0666 | IPC\_CREAT); // Access shared memory char \*str = (char \*)shmat(shmid, (void \*)0, 0); // Attach shared memory printf("Data read from memory: %s\n", str); // Read data from shared memory

shmdt(str); // Detach shared memory

shmctl(shmid, IPC\_RMID, NULL); // Remove shared memory segment return 0;

}

##### OUTPUT:

###### Writer:

gcc write.c –o write

./write

Data written in memory Hii

###### Reader:

Gcc read.c –o read

./read

Data read from memory Hii

# SEMAPHORES

##### PROGRAM:

#include <stdio.h> #include <stdlib.h>

int mutex = 1, full = 0, empty = 3, x = 0; int wait(int s) { return --s; }

int signal(int s) { return ++s; }

void producer() { mutex = wait(mutex); full = signal(full); empty = wait(empty);

printf("\nProducer produces item %d", ++x); mutex = signal(mutex);

}

void consumer() { mutex = wait(mutex); full = wait(full);

empty = signal(empty);

printf("\nConsumer consumes item %d", x--); mutex = signal(mutex);

}

int main() { int n;

printf("\n1. Producer\n2. Consumer\n3. Exit\n"); while (1) {

printf("\nEnter your choice: "); scanf("%d", &n);

switch (n) { case 1:

if (mutex == 1 && empty != 0) producer(); else printf("Buffer is full\n");

break; case 2:

if (mutex == 1 && full != 0) consumer(); else printf("Buffer is empty\n");

break;

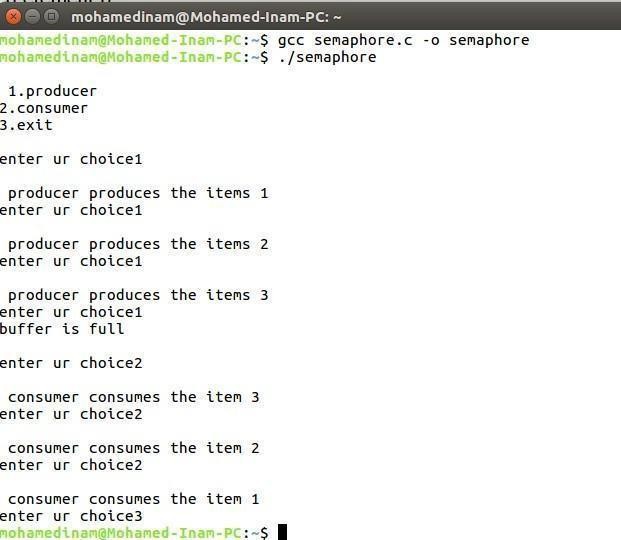
case 3: exit(0);

}

}

}

##### OUTPUT:

****

|  |  |
| --- | --- |
| **EX NO :** | **DEADLOCK AVOIDANCE – BANKER’S ALGORITHM** |
| **DATE :** |

**AIM:**

## To implement deadlock avoidance by using Banker’s Algorithm.

**Banker’s Algorithm:**

## When a new process enters a system, it must declare the maximum number of instances of each resource type it needed. This number may exceed the total number of resources in the system. When the user request a set of resources, the system must determine whether the allocation of each resources will leave the system in safe state. If it will the resources are allocation; otherwise the process must wait until some other process release the resources.

###### Data structures

 n-Number of process, m-number of resource types.

 Available: Available[j]=k, k – instance of resource type Rj is available.  Max: If max[i, j]=k, Pi may request at most k instances resource Rj.

 Allocation: If Allocation [i, j]=k, Pi allocated to k instances of resource Rj

 Need: If Need[I, j]=k, Pi may need k more instances of resource type Rj, Need [I, j]=Max[I, j]-Allocation[I, j];

###### Safety Algorithm

1. Work and Finish be the vector of length m and n respectively, Work=Available and Finish[i] =False.
2. Find an i such that both
   * Finish[i] =False
   * Need<=Work If no such I exists go to 4.
3. work=work+Allocation, Finish[i] =True;
4. if Finish[1]=True for all I, then the system is in safe state.

###### Resource request algorithm

Let Request i be request vector for the process Pi, If request i=[j]=k, then process Pi wants k instances of resource type Rj.

1. if Request<=Need I go to 2. Otherwise raise an error condition.
2. if Request<=Available go to 3. Otherwise Pi must since the resources are available.
3. Have the system pretend to have allocated the requested resources to process Pi by modifying the state as follows;

|  |  |  |
| --- | --- | --- |
| Available | = | Available-Request I; |
| Allocation I | = | Allocation+Request I; |
| Need I | = | Need i-Request I; |

If the resulting resource allocation state is safe, the transaction is completed and process Pi is allocated its resources. However if the state is unsafe, the Pi must wait for Request i and the old resource-allocation state is restored.

##### ALGORITHM:

1. Start the program.
2. Get the values of resources and processes.
3. Get the avail value.
4. After allocation find the need value.
5. Check whether its possible to allocate.
6. If it is possible then the system is in safe state.
7. Else system is not in safety state.
8. If the new request comes then check that the system is in safety.
9. Or not we allow the request.
10. Stop the program.

# BANKER’S ALGORITHM

##### PROGRAM:

#include <stdio.h> #include <stdlib.h>

struct da {

int max[10], a1[10], need[10], before[10], after[10];

} p[10];

int main() {

int i, j, l, r, n, tot[10], av[10], cn = 0, cz = 0, temp = 0, c = 0;

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

printf("Enter the number of resources: "); scanf("%d", &r);

for (i = 0; i < n; i++) { printf("Process %d:\n", i + 1); for (j = 0; j < r; j++) {

printf("Max value for resource %d: ", j + 1); scanf("%d", &p[i].max[j]);

}

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

printf("Allocated from resource %d: ", j + 1); scanf("%d", &p[i].a1[j]);

p[i].need[j] = p[i].max[j] - p[i].a1[j];

}

}

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

printf("Enter total value of resource %d: ", i + 1); scanf("%d", &tot[i]);

}

for (i = 0; i < r; i++) { temp = 0;

for (j = 0; j < n; j++) temp += p[j].a1[i]; av[i] = tot[i] - temp;

}

printf("\n\tProcess\tMax\tAlloc\tNeed\tTotal\tAvail"); for (i = 0; i < n; i++) {

printf("\nP%d\t", i + 1);

for (j = 0; j < r; j++) printf("%d ", p[i].max[j]); printf("\t");

for (j = 0; j < r; j++) printf("%d ", p[i].a1[j]); printf("\t");

for (j = 0; j < r; j++) printf("%d ", p[i].need[j]); printf("\t");

if (i == 0)

for (j = 0; j < r; j++) printf("%d ", tot[j]); printf("\t");

if (i == 0)

for (j = 0; j < r; j++) printf("%d ", av[j]);

}

printf("\n\n\tAvail Before\tAvail After"); for (l = 0; l < n; l++) {

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

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

if (p[i].need[j] > av[j]) cn++;

if (p[i].max[j] == 0) cz++;

}

if (cn == 0 && cz != r) { for (j = 0; j < r; j++) {

p[i].before[j] = av[j] - p[i].need[j];

p[i].after[j] = p[i].before[j] + p[i].max[j]; av[j] = p[i].after[j];

p[i].max[j] = 0;

}

printf("\nP%d\t", i + 1);

for (j = 0; j < r; j++) printf("%d ", p[i].before[j]); printf("\t");

for (j = 0; j < r; j++) printf("%d ", p[i].after[j]); c++;

break;

}

}

}

if (c == n)

printf("\nThe above sequence is a safe sequence.\n"); else

printf("\nDeadlock occurred.\n");

return 0;

}

###### OUTPUT: RUN 1: NO deadlock



***l!***-***'*** mo hamedinam@Mohame**d-lnam-PC:** ~

**oha edlna @Noha ed-Ina -P** :-$ gee bankers.e -o ban ers

**oha edlna @Noha ed-Ina -P** :-$ ./banke s EN ER THE **NO.** OF PROCESSES 4

EN ER THE **NO.** OF RESOURCES 3

PROCESS 1

**MAXIMUM** VALUE FOR RESOURCE 1 3

**MAXIMUM** VALUE FOR RESOURCE 2 2

|  |  |  |  |
| --- | --- | --- | --- |
| **MAXIMUM** VALUE | FOR RESOURCE | 3 | 2 |
| ALLOCATED FROM | RESOURCE 1 | 1 |  |
| ALLOCATED FROM | RESOURCE 2 | 0 |  |
| ALLOCATED FROM | RESOURCE 3 | 0 |  |
| PROCESS 2 |  |  |  |
| **MAXIMUM** VALUE | FOR RESOURCE | 1 | 6 |
| **MAXIMUM** VALUE | FOR RESOURCE | 2 | 1 |
| **MAXIMUM** VALUE | FOR RESOURCE | 3 | 3 |
| ALLOCATED FROM | RESOURCE 1 | 5 |  |
| ALLOCATED FROM | RESOURCE 2 | 1 |  |
| ALLOCATED FROM | RESOURCE 3 | 1 |  |
| PROCESS 3 |  |  |  |
| **MAXIMUM** VALUE | FOR RESOURCE | 1 | 3 |
| **MAXIMUM** VALUE | FOR RESOURCE | 2 | 1 |
| **MAXIMUM** VALUE | FOR RESOURCE | 3 | 4 |
| ALLOCATED FROM | RESOURCE 1 | 2 |  |
| ALLOCATED FROM | RESOURCE 2 | 1 |  |
| ALLOCATED FROM | RESOURCE 3 | 1 |  |
| PROCESS 4 |  |  |  |
| **MAXIMUM** VALUE | FOR RESOURCE | 1 | 4 |
| **MAXIMUM** VALUE | FOR RESOURCE | 2 | 2 |

**MAXIMUM** VALUE FOR RESOURCE 3 2

.

ALLOCATED FROM RESOURCE 1 0

ALLOCATED FROM RESOURCE 2 . 0

ALLOCATED FROM RESOURCE 3 2

|  |  |  |  |
| --- | --- | --- | --- |
| ENTER TOTAL VALUE OF RESOURCE 1 | |  | 9 |
| ENTER | TOTAL VALUE OF RESOURCE | 2 | 3 |
| ENTER | TOTAL VALUE OF RESOURCE | 3 | 6 |

RESOURCES ALLOCATED NEEDED TOTAL AVAIL

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **P1** |  | 322 | **100** | 222 | **936** | **112** |
| **P2** |  | **613** | **511** | **102** | |  |
| **P3** |  | **314** | **211** | **103** | |  |
| **P4** |  | **422** | **002** | **420** | |  |
|  |  | AVAIL BEFORE AVAIL AFTER | | | |  |
| p | 2 | 010 | 623 | | |  |
| p | 1 | 401 | 723 | | |  |
| p | 3 | 620 | 934 | | |  |

p 4 514 936

THE ABOVE SEQUENCE IS A SAFE SEQUENCE

I

**ha edlna @N**• :-$

# RUN2: Deadlock occurs



"- - mohamedinam@Mohamed-lnam-PC: ~

bankers.e -o bankers bankers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **oha edlna @N** | **ha ed** | | :-$ | gee  ./  4  3 |
| **oha edlna @No** | | **d-Ina -P** | |
| ENTER TIHE **NO** | | PROCESSE | S |
| ENTER TIHE **NO** | | RESOURCE | S |
| **MAXIMUM** VALUE | FOR | RESOURCE | 1 | 3 |
| MAXIMUM VALUE | FOR | RESOURCE | 2 | 2 |
| **MAXIMUM** VALUE | FOR | RESOURCE | 3 | 2 |
| ALLOCATE!)FROM | RESOURCE 1 | | 1 |  |
| ALLOCATE!)FROM | RESOURCE 2 | | 0 |
| ALLOCATE!)FROM | RESOURCE 3 | | 1 |  |
| PROCESS 2 |  |  |  |  |
| **MAXIMUM** VALUE | FOR RESOURCE | | 1 | 6 |
| MAXIMUM VALUE | FOR RESOURCE | | 2 | 1 |
| **MAXIMUM** VALUE | FOR RESOURCE | | 3 | 3 |
| ALLOCATEIJFROM | RESOURCE 1 | | 5 |  |
| ALLOCATE!)FROM | RESOURCE 2 | | 1 |  |
| ALLOCATE!)FROM | RESOURCE 3 | | 1 |  |
| PROCESS 3 |  |  |  |  |
| **MAXIMUM** VALUE | FOR RESOURCE | | 1 | 3 |
| **MAXIMUM** VALUE | FOR RESOURCE | | 2 | 1 |
| **MAXIMUM** VALUE | FOR RESOURCE | | 3 | 4 |
| ALLOCATE!)FROM | RESOURCE 1 | | 2 |  |
| ALLOCATE!)FROM | RESOURCE 2 | | 1 |  |
| ALLOCATE!)FROM  **-MAXIMUM** VALUE | | | | |

9

3

6

RESOURCES ALLOCATE!) NEEDEIJ TOTAL AVAIL

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P1 | 322 | **Hl1** | 221 | 936 | 110 |
| P2 | 613 | 511 | 102 |  |  |
| P3 | 314 | 212 | 102 |  |  |
| P4 | 422 | 082 | 420 |  |  |

AVAIL BEFORE AVAIL AFTER

IJEAOLOCK OCCUREIJ

I

**ha edlna @N** :-$

# DEADLOCK DETECTION – BANKER’S ALGORITHM

##### PROGRAM:

#include<stdio.h> int main() {

int np, nr, i, j;

int alloc[10][10], request[10][10], avail[10], r[10], w[10], mark[20] = {0};

printf("\nEnter the number of processes: "); scanf("%d", &np);

printf("\nEnter the number of resources: "); scanf("%d", &nr);

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

printf("\nTotal amount of Resource R%d: ", i + 1); scanf("%d", &r[i]);

}

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]);

for (j = 0; j < nr; j++) { avail[j] = r[j];

for (i = 0; i < np; i++) avail[j] -= alloc[i][j];

}

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;

}

for (j = 0; j < nr; j++) w[j] = avail[j];

int deadlock = 1;

for (i = 0; i < np; i++) { if (mark[i] == 0) {

int canbeprocessed = 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];

deadlock = 0;

}

}

}

if (deadlock)

printf("\nDeadlock detected!\n"); else

printf("\nNo Deadlock possible.\n");

return 0;

}

# OUTPUT:

Enter the no of process: 4 Enter the no of resources: 5

Total Amount of the Resource R1: 2 Total Amount of the Resource R2: 1 Total Amount of the Resource R3: 1 Total Amount of the Resource R4: 2 Total Amount of the Resource R5: 1

###### Enter the request matrix: 0 1 0 0 1

0 0 1 0 1

0 0 0 0 1

1 0 1 0 1

###### Enter the allocation matrix: 1 0 1 1 0

1 1 0 0 0

0 0 0 1 0

0 0 0 0 0

**Deadlock detected**

# IMPLEMENTATION OF THREADING AND SYNCHRONIZATION APPLICATIONS

##### PROGRAM:

#include<stdio.h> #include <string.h> #include<pthread.h>

// Global variable:

int i = 2;

void\* foo(void\* p){

// Print value received as argument:

printf(“Value recevied as argument in starting routine:”); printf(“%i\n”, \* (int\*)p);

// Return reference to global variable: pthread\_exit(&i);

}

int main(void){

// Declare variable for thread’s ID: pthread\_t id;

int j = 1;

pthread\_create(&id, NULL, foo, &j); int\* ptr;

// Wait for foo() and retrieve value in ptr; pthread\_join(id, (void\*\*)&ptr);

printf(“Value recevied by parent from child: “); printf(“%i\n”, \*ptr);

return 0 ;

}

##### OUTPUT:

Value received as argument in starting routine: 1 Value received by parent from child: 2

# PAGING TECHNIQUE OF MEMORY MANAGEMENT

##### PROGRAM:

#include <stdio.h>

struct pstruct {

int fno;

int pbit;

} ptable[10];

int pmsize, lmsize, psize, frame, page, ftable[20], frameno;

void info() {

printf("\n\nMEMORY MANAGEMENT USING PAGING\n\n");

printf("Enter the Size of Physical memory: ");

scanf("%d", &pmsize);

printf("Enter the size of Logical memory: ");

scanf("%d", &lmsize);

printf("Enter the partition size: ");

scanf("%d", &psize);

frame = pmsize / psize;

page = lmsize / psize;

printf("\nThe physical memory is divided into %d no.of frames\n", frame);

printf("The Logical memory is divided into %d no.of pages\n", page);

}

void assign() {

int i;

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

ptable[i].fno = -1;

ptable[i].pbit = -1;

}

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

ftable[i] = 32555;

}

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

printf("\nEnter the Frame number where page %d must be placed: ", i);

scanf("%d", &frameno);

ftable[frameno] = i;

if (ptable[i].pbit == -1) {

ptable[i].fno = frameno;

ptable[i].pbit = 1;

}

}

printf("\nPAGE TABLE\n");

printf("PageAddress\tFrameNo.\tPresenceBit\n");

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

printf("%d\t\t%d\t\t%d\n", i, ptable[i].fno, ptable[i].pbit);

}

printf("\nFRAME TABLE\n");

printf("FrameAddress\tPageNo\n");

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

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

}

}

void cphyaddr() {

int laddr, paddr, disp, phyaddr, baddr;

printf("\n\nProcess to create the Physical Address\n");

printf("Enter the Base Address: ");

scanf("%d", &baddr);

printf("Enter the Logical Address: ");

scanf("%d", &laddr);

paddr = laddr / psize;

disp = laddr % psize;

if (ptable[paddr].pbit == 1) {

phyaddr = baddr + (ptable[paddr].fno \* psize) + disp;

printf("The Physical Address where the instruction is present: %d\n", phyaddr);

} else {

printf("Page is not present in memory.\n");

}

}

int main() {

info();

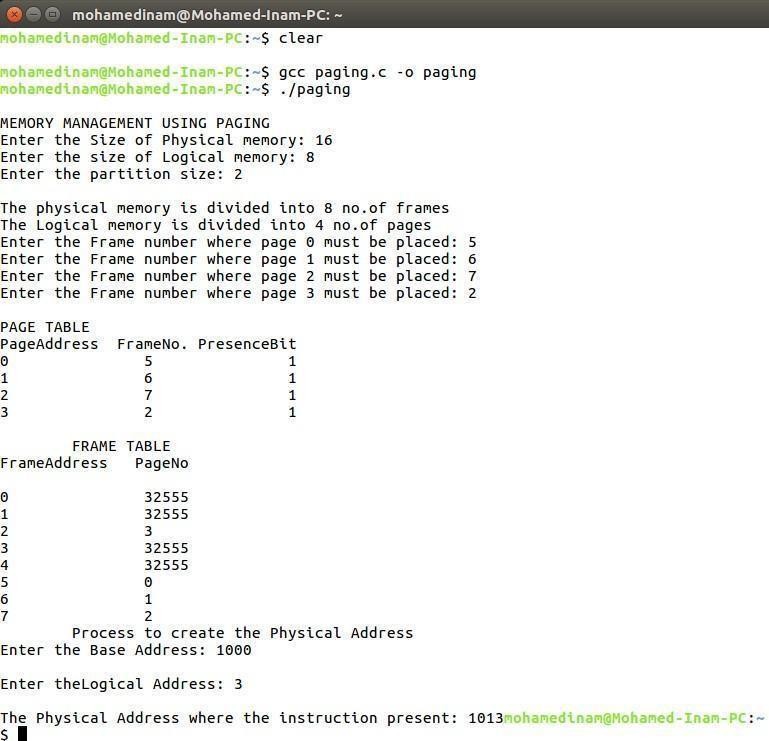
assign();

cphyaddr();

return 0;

}

**OUTPUT:**

****

# MEMORY ALLOCATION METHODS FOR FIXED PARTITION

**FIRST FIT ALLOCATION**

**PROGRAM :**

#include <stdio.h>

struct process {

int size;

int flag;

int holeid;

};

struct hole {

int size;

int actual;

};

int main() {

int i, np, nh, j;

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

scanf("%d", &nh);

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

printf("Enter size for hole H%d: ", i);

scanf("%d", &h[i].size);

h[i].actual = h[i].size;

}

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

scanf("%d", &np);

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

printf("Enter the size of process P%d: ", i);

scanf("%d", &p[i].size);

p[i].flag = 0;

}

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

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

if(p[i].flag != 1) {

if(p[i].size <= h[j].size) {

p[i].flag = 1;

p[i].holeid = j;

h[j].size -= p[i].size;

}

}

}

}

printf("\n\tFirst Fit\n");

printf("\nProcess\tPSize\tHole");

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

if(p[i].flag != 1)

printf("\nP%d\t%d\tNot allocated", i, p[i].size);

else

printf("\nP%d\t%d\tH%d", i, p[i].size, p[i].holeid);

}

printf("\n\nHole\tActual\tAvailable");

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

printf("\nH%d\t%d\t%d", i, h[i].actual, h[i].size);

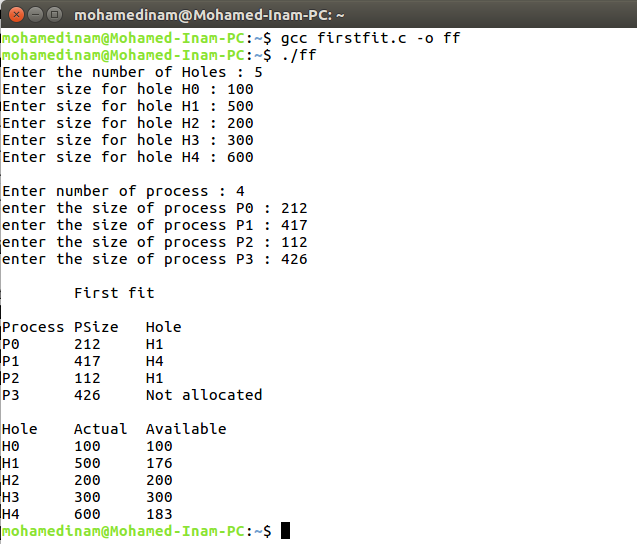
}

printf("\n");

return 0;

}

**OUTPUT:**



# MEMORY ALLOCATION METHODS FOR FIXED PARTITION

**WORST FIT ALLOCATION**

**PROGRAM :**

#include<stdio.h>

#define max 25

void main() {

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

static int bf[max], ff[max];

printf("\n\tMemory Management Scheme - First Fit");

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]);

}

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

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

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

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

}

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

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

if(bf[j] != 1) {

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

if(temp >= 0) {

ff[i] = j;

break;

}

}

}

frag[i] = temp;

bf[ff[i]] = 1;

}

printf("\nFile\_no:\tFile\_size:\tBlock\_no:\tBlock\_size:\tFragment");

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

printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d", i, f[i], ff[i], b[ff[i]], frag[i]);

}

}

**INPUT**

Enter the number of blocks 3

Enter the number of files 2

Enter the size of the blocks:-

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Block 1 | : | 5 |  |  |  |
| Block 2 | : | 2 |  |  |  |
| Block 3 | : | 7 |  |  |  |
| Enter the size of the files:- | | | | | |
| File 1: 1 |  |  |  |  |  |
| File 2: 4 |  |  |  |  |  |
|  |  |  |  | **OUTPUT** |  |
| **File No** | **File Size** | | **Block No** | **Block Size** | **Fragment** |
| 1 | 1 |  | 1 | 5 | 4 |
| 2 | 4 |  | 3 | 7 | 3 |

# MEMORY ALLOCATION METHODS FOR FIXED PARTITION

**BEST FIT ALLOCATION**

##### PROGRAM :

#include <stdio.h>

struct process {

int size;

int flag;

int holeid;

} p[10];

struct hole {

int hid;

int size;

int actual;

} h[10];

void bsort(struct hole bh[], int n);

int main() {

int i, np, nh, j;

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

scanf("%d", &nh);

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

printf("Enter size for hole H%d: ", i);

scanf("%d", &h[i].size);

h[i].actual = h[i].size;

h[i].hid = i;

}

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

scanf("%d", &np);

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

printf("Enter the size of process P%d: ", i);

scanf("%d", &p[i].size);

p[i].flag = 0;

}

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

bsort(h, nh);

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

if(p[i].flag != 1) {

if(p[i].size <= h[j].size) {

p[i].flag = 1;

p[i].holeid = h[j].hid;

h[j].size -= p[i].size;

}

}

}

}

printf("\n\tBest Fit\n");

printf("\nProcess\tPSize\tHole");

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

if(p[i].flag != 1)

printf("\nP%d\t%d\tNot allocated", i, p[i].size);

else

printf("\nP%d\t%d\tH%d", i, p[i].size, p[i].holeid);

}

printf("\n\nHole\tActual\tAvailable");

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

printf("\nH%d\t%d\t%d", h[i].hid, h[i].actual, h[i].size);

}

printf("\n");

return 0;

}

void bsort(struct hole bh[], int n) {

struct hole temp;

int i, j;

for(i = 0; i < n - 1; i++) {

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

if(bh[i].size > bh[j].size) {

temp = bh[i];

bh[i] = bh[j];

bh[j] = temp;

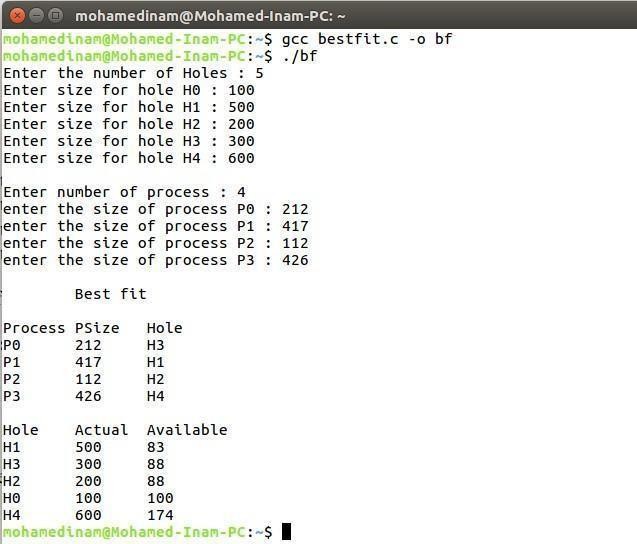
}

}

}

}

##### OUTPUT :

****

**PROGRAM :**

#include<stdio.h>

int i, j, nof, nor, flag = 0, ref[50], frm[50], pf = 0, victim = -1;

int main() {

printf("\n \t\t\t FIFO PAGE REPLACEMENT ALGORITHM");

// Input number of frames

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

scanf("%d", &nof);

// Input number of reference string

printf("Enter number of reference string: ");

scanf("%d", &nor);

// Input reference string

printf("\n Enter the reference string: ");

for (i = 0; i < nor; i++) {

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

}

printf("\nThe given reference string: ");

for (i = 0; i < nor; i++) {

printf("%4d", ref[i]);

}

// Initialize frames with -1 (indicating empty frames)

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

frm[i] = -1;

}

printf("\n");

// Implement FIFO page replacement algorithm

for (i = 0; i < nor; i++) {

flag = 0;

printf("\n\t Reference page %d ->\t", ref[i]);

// Check if the current reference page is already in the frames

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

if (frm[j] == ref[i]) {

flag = 1;

break;

}

}

// If page is not found in the frames, replace the oldest page

if (flag == 0) {

pf++; // Increment page fault counter

victim++; // Move the victim pointer

victim = victim % nof; // Ensure circular allocation

frm[victim] = ref[i]; // Replace the page

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

printf("%4d", frm[j]);

}

}

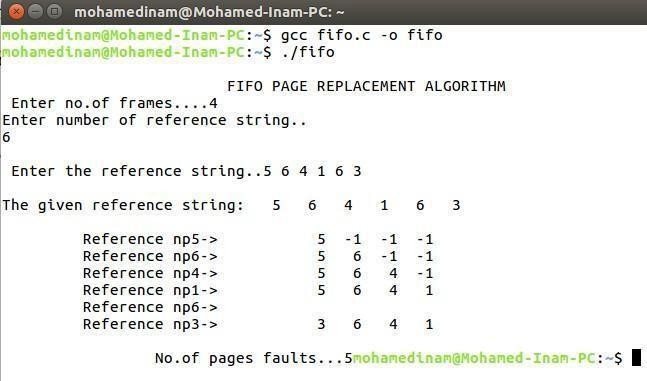
}

printf("\n\n\t\t Number of page faults: %d", pf);

return 0;

}

**OUTPUT :**

****

# LRU PAGE REPLACEMENT ALGORITHM

##### PROGRAM :

#include<stdio.h>

int i, j, nof, nor, flag = 0, ref[50], frm[50], pf = 0, victim = -1;

int recent[50], lrucal[50], count = 0;

int lruvictim();

int main() {

printf("\n\t\t\t LRU PAGE REPLACEMENT ALGORITHM");

// Input number of frames

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

scanf("%d", &nof);

// Input number of reference string

printf("Enter number of reference string: ");

scanf("%d", &nor);

// Input reference string

printf("\n Enter reference string: ");

for (i = 0; i < nor; i++) {

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

}

printf("\n\n\t\t LRU PAGE REPLACEMENT ALGORITHM ");

printf("\n\t The given reference string: ");

for (i = 0; i < nor; i++) {

printf("%4d", ref[i]);

}

// Initialize frames and LRU calculation arrays

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

frm[i] = -1;

lrucal[i] = 0;

}

for (i = 0; i < 50; i++) {

recent[i] = 0;

}

printf("\n");

// Process the reference string for page replacement

for (i = 0; i < nor; i++) {

flag = 0;

printf("\n\t Reference NO %d ->\t", ref[i]);

// Check if the page is already in any of the frames

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

if (frm[j] == ref[i]) {

flag = 1;

break;

}

}

// If page is not found in the frames, replace the least recently used page

if (flag == 0) {

count++; // Increment the page count

if (count <= nof) {

victim++; // Victim is just the next available frame

} else {

victim = lruvictim(); // Get the least recently used page frame

}

pf++; // Increment the page fault counter

frm[victim] = ref[i]; // Replace the page in the victim frame

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

printf("%4d", frm[j]); // Print current frames

}

}

// Update the recent usage information for the current page

recent[ref[i]] = i;

}

printf("\n\n\t Number of page faults: %d", pf);

return 0;

}

// Function to determine the least recently used page

int lruvictim() {

int i, j, temp1, temp2;

// Record the last usage time of each page in frames

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

temp1 = frm[i];

lrucal[i] = recent[temp1];

}

temp2 = lrucal[0]; // Start with the first frame's LRU time

for (j = 1; j < nof; j++) {

if (temp2 > lrucal[j]) {

temp2 = lrucal[j]; // Find the least recently used page

}

}

// Find and return the index of the least recently used page frame

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

if (ref[temp2] == frm[i]) {

return i;

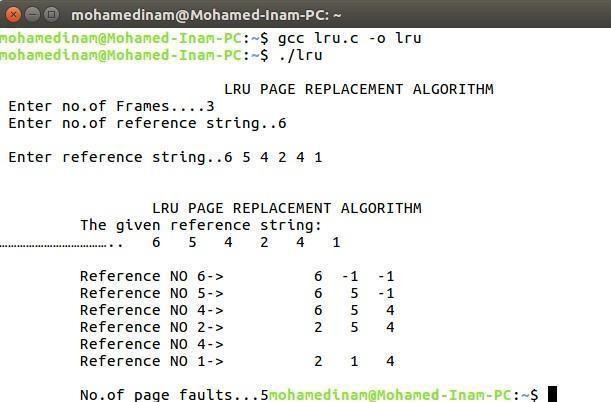
}

}

return 0; // Default return if no match found (though this shouldn't happen)

}

**OUTPUT :**

****

# OPTIMAL (LFU) PAGE REPLACEMENT ALGORITHM

**PROGRAM:**

#include<stdio.h>

int i, j, nof, nor, flag = 0, ref[50], frm[50], pf = 0, victim = -1;

int recent[50], optcal[50], count = 0;

int optvictim(int index);

int main() {

printf("\nOPTIMAL PAGE REPLACEMENT ALGORITHM");

// Input the number of frames

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

scanf("%d", &nof);

// Input the number of reference string

printf("Enter the number of reference string: ");

scanf("%d", &nor);

// Input the reference string

printf("Enter the reference string: ");

for (i = 0; i < nor; i++) {

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

}

// Initialize frames and other arrays

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

frm[i] = -1;

optcal[i] = 0;

}

for (i = 0; i < 50; i++) {

recent[i] = 0;

}

printf("\nThe given reference string: ");

for (i = 0; i < nor; i++) {

printf("%4d", ref[i]);

}

printf("\n");

// Process the reference string for page replacement

for (i = 0; i < nor; i++) {

flag = 0;

printf("\n\tReference no %d ->\t", ref[i]);

// Check if the page is already in any of the frames

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

if (frm[j] == ref[i]) {

flag = 1;

break;

}

}

// If the page is not found in the frames, replace it

if (flag == 0) {

count++; // Increment page count

if (count <= nof) {

victim++; // If there's space, just put it in the next available frame

} else {

victim = optvictim(i); // Use the optimal victim selection

}

pf++; // Increment page fault count

frm[victim] = ref[i]; // Replace the page in the victim frame

// Display the current frame state

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

printf("%4d", frm[j]);

}

}

}

printf("\nNumber of page faults: %d", pf);

return 0;

}

// Function to select the optimal victim page

int optvictim(int index) {

int i, j, temp, notfound;

// For each frame, check when the page will be used next

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

notfound = 1;

for (j = index; j < nor; j++) {

if (frm[i] == ref[j]) {

notfound = 0;

optcal[i] = j;

break;

}

}

// If a page is not found in the future, it is the victim

if (notfound == 1) {

return i;

}

}

// Find the frame with the furthest usage (optimal victim)

temp = optcal[0];

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

if (temp < optcal[i]) {

temp = optcal[i];

}

}

// Return the frame index with the furthest usage

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

if (optcal[i] == temp) {

return i;

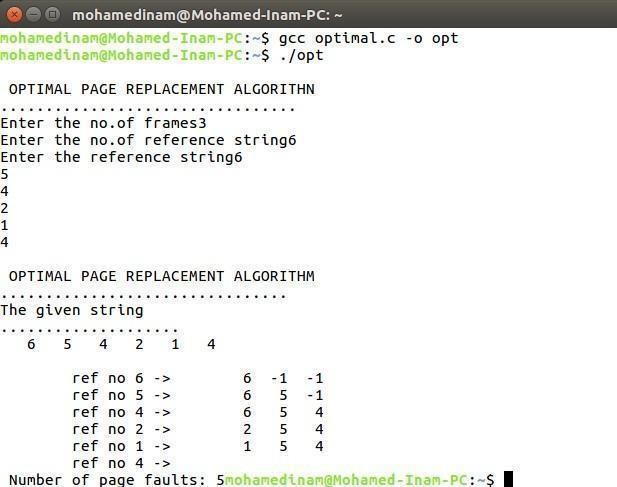
}

}

return 0; // Default return if no match found

}

**OUTPUT :**

****

# VARIOUS FILE ORGANIZATION TECHNIQUES

##### IMPLEMENTATION OF SINGLE LEVEL DIRECTORY PROGRAM :

#include<stdio.h>

int main() {

int master, s[20];

char f[20][20][20]; // Stores filenames (max 20 directories, each directory can have 20 files, and each filename can have 20 characters)

char d[20][20]; // Directory names (max 20 directories, each directory name up to 20 characters)

int i, j;

// Input number of directories

printf("Enter number of directories: ");

scanf("%d", &master);

// Input names of directories

printf("Enter names of directories: ");

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

scanf("%s", d[i]);

}

// Input size of directories (number of files in each directory)

printf("Enter size of directories: ");

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

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

}

// Input filenames for each directory

printf("Enter the file names:\n");

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

for(j = 0; j < s[i]; j++) {

scanf("%s", f[i][j]);

}

}

// Output the directory information

printf("\nDirectory\tSize\tFilenames\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

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

printf("%-12s\t%-4d\t", d[i], s[i]);

for(j = 0; j < s[i]; j++) {

printf("%s ", f[i][j]);

}

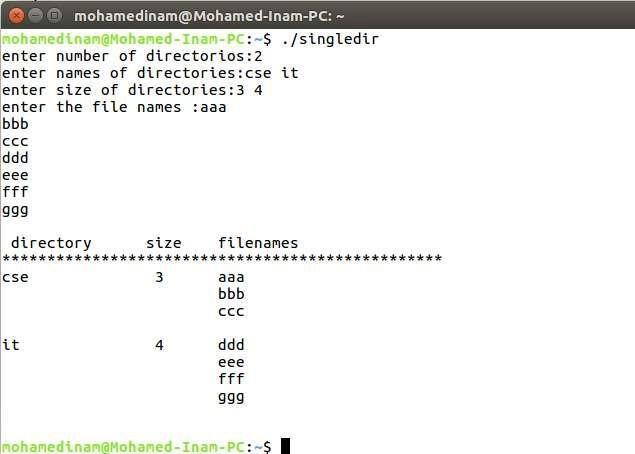
printf("\n");

}

return 0;

}

**OUTPUT :**

****

# IMPLEMENTATION OF TWO-LEVEL DIRECTORY

##### PROGRAM :

#include<stdio.h>

struct st {

char dname[10]; // Directory name

char sdname[10][10]; // Subdirectory names

char fname[10][10][10]; // File names inside subdirectories

int ds; // Number of subdirectories

int sds[10]; // Size of subdirectories (number of files in each subdirectory)

} dir[10];

void main() {

int i, j, k, n;

// Input the number of directories

printf("Enter number of directories: ");

scanf("%d", &n);

// Loop over each directory

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

// Input directory name

printf("Enter directory %d name: ", i + 1);

scanf("%s", dir[i].dname);

// Input the number of subdirectories

printf("Enter size of directory %s (number of subdirectories): ", dir[i].dname);

scanf("%d", &dir[i].ds);

// Loop over each subdirectory in the directory

for (j = 0; j < dir[i].ds; j++) {

// Input subdirectory name and size (number of files)

printf("Enter subdirectory name for %s: ", dir[i].dname);

scanf("%s", dir[i].sdname[j]);

printf("Enter size (number of files) for subdirectory %s: ", dir[i].sdname[j]);

scanf("%d", &dir[i].sds[j]);

// Input file names for each subdirectory

for (k = 0; k < dir[i].sds[j]; k++) {

printf("Enter file name for subdirectory %s: ", dir[i].sdname[j]);

scanf("%s", dir[i].fname[j][k]);

}

}

}

// Output the directory structure

printf("\nDirectory Name\tSize\tSubdirectory Name\tSize\tFiles\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

// Loop over each directory and its subdirectories

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

printf("%s\t\t%d", dir[i].dname, dir[i].ds);

// Loop over each subdirectory

for (j = 0; j < dir[i].ds; j++) {

printf("\t%s\t\t%d\t", dir[i].sdname[j], dir[i].sds[j]);

// Loop over each file in the subdirectory

for (k = 0; k < dir[i].sds[j]; k++) {

printf("%s\t", dir[i].fname[j][k]);

}

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

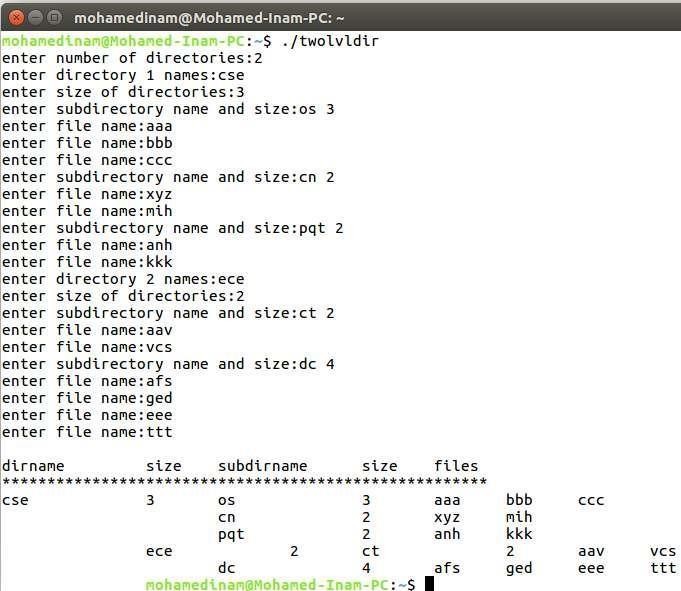
}

printf("\n");

}

}

**OUTPUT :**

****

**PROGRAM :**

# FILE ALLOCATION STRATEGIES

##### SEQUENTIAL FILE ALLOCATION

#include<stdio.h>

void main() {

int n, i, j, b[20], sb[20], t[20], x, c[20][20];

// Take input for number of files

printf("Enter number of files: ");

scanf("%d", &n);

// Input for each file

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

printf("Enter the number of blocks occupied by file %d: ", i + 1);

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

printf("Enter the starting block of file %d: ", i + 1);

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

t[i] = sb[i]; // The starting block of the file is saved

// Calculate the blocks occupied

for (j = 0; j < b[i]; j++) {

c[i][j] = sb[i]++; // Store the block numbers for the file

}

}

// Output the file details (Filename, Start Block, and Length)

printf("\nFilename\tStart Block\tLength\n");

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

printf("%d\t\t%d\t\t%d\n", i + 1, t[i], b[i]); // Printing file details

}

// Ask user to input file number to get details

printf("\nEnter file number to get details: ");

scanf("%d", &x);

// Validate the input file number

if (x < 1 || x > n) {

printf("Invalid file number!\n");

return;

}

// Output the details of the specified file

printf("\nFile name: %d\n", x);

printf("Length: %d\n", b[x - 1]); // Length of the file

printf("Blocks occupied: ");

for (i = 0; i < b[x - 1]; i++) {

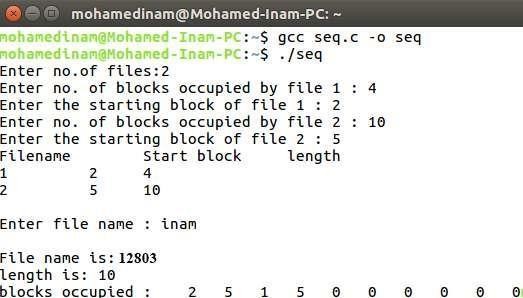
printf("%d ", c[x - 1][i]); // Output blocks occupied by the file

}

printf("\n");

}

OUTPUT:

****

**FILE ALLOCATION STRATEGIES – INDEXED FILE ALLOCATION**

##### PROGRAM :

#include<stdio.h>

void main() {

int n, m[20], i, j, sb[20], s[20], b[20][20], x;

// Input number of files

printf("Enter no. of files: ");

scanf("%d", &n);

// Input for each file

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

printf("Enter starting block and size of file %d: ", i + 1);

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

printf("Enter blocks occupied by file %d: ", i + 1);

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

printf("Enter blocks of file %d: ", i + 1);

for (j = 0; j < m[i]; j++) {

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

}

}

// Output the details of files

printf("\nFile\tIndex\tLength\n");

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

printf("%d\t%d\t%d\n", i + 1, sb[i], m[i]);

}

// Ask for a file number

printf("\nEnter file number to get details: ");

scanf("%d", &x);

// Validate the file number

if (x < 1 || x > n) {

printf("Invalid file number!\n");

return;

}

// Output the details for the specified file

printf("\nFile name is: %d\n", x);

i = x - 1; // Adjust for 0-based indexing

printf("Index is: %d\n", sb[i]);

printf("Blocks occupied are: ");

for (j = 0; j < m[i]; j++) {

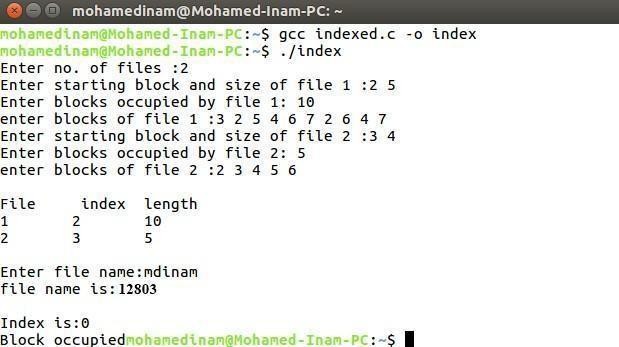
printf("%3d", b[i][j]);

}

printf("\n");

}

**OUTPUT:**

****

# DISK SCHEDULING ALGORITHM

##### FIRST COME FIRST SERVE ALGORITHM

**PROGRAM :**

#include <stdio.h>

#include <stdlib.h>

int main() {

int RQ[100], i, n, TotalHeadMoment = 0, initial;

// Input number of requests

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

scanf("%d", &n);

// Validate that the number of requests is greater than 0

if (n <= 0) {

printf("Invalid number of requests. The number of requests should be greater than 0.\n");

return 1;

}

// Input the request sequence

printf("Enter the Requests sequence: \n");

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

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

}

// Input the initial head position

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

scanf("%d", &initial);

// Calculate the total head movement

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

TotalHeadMoment += abs(RQ[i] - initial); // Absolute difference between current head and request

initial = RQ[i]; // Move the head to the current request position

}

// Output the total head movement

printf("Total head movement is %d\n", TotalHeadMoment);

return 0;

}

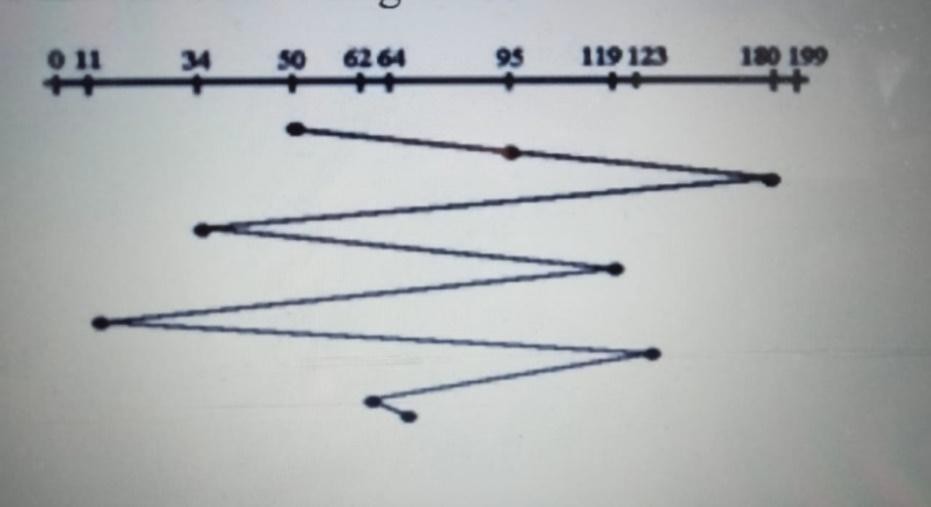
##### OUTPUT:

Enter the number of Request 8 Enter the Requests Sequence 95 180 34 119 11 123 62 64

Enter initial head position 50

Total head movement is 644

Example: Given the following queue -- 95, 180, 34, 119, 11, 123, 62, 64 with the Read-write head initially at the track 50 and the tail track being at 199.



# DISK SCHEDULING ALGORITHM

##### SHORTEST SEEK TIME FIRST (SSTF) ALGORITHM

**PROGRAM :**

#include <stdio.h>

#include <stdlib.h>

int main() {

int RQ[100], i, n, TotalHeadMoment = 0, initial, count = 0;

// Input number of requests

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

scanf("%d", &n);

// Validate if the number of requests is greater than 0

if (n <= 0) {

printf("Invalid number of requests. The number of requests should be greater than 0.\n");

return 1;

}

// Input the request sequence

printf("Enter the Requests sequence:\n");

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

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

}

// Input the initial head position

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

scanf("%d", &initial);

// Process the requests using the SSTF algorithm

while (count != n) {

int min = 1000, d, index;

// Find the request with the minimum distance

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

d = abs(RQ[i] - initial); // Calculate the absolute distance

if (min > d && RQ[i] != -1) { // Ensure not processing already processed request

min = d;

index = i;

}

}

// Update total head movement and move the head to the selected request

TotalHeadMoment += min;

initial = RQ[index];

// Mark this request as processed by setting it to -1

RQ[index] = -1; // We mark the request as processed

count++;

}

// Output the total head movement

printf("Total head movement is %d\n", TotalHeadMoment);

return 0;

}

OUTPUT:

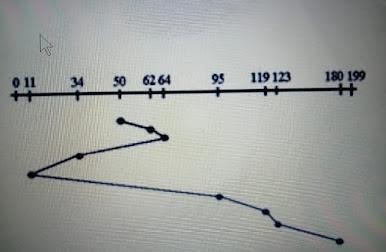
Enter the number of Request 8 Enter Request Sequence

95 180 34 119 11 123 62 64

Enter initial head Position 50

Total head movement is 236

**Example:-**: Given the following queue -- 95, 180, 34, 119, 11, 123, 62, 64 with the Read-write head initially at the track 50 and the tail track being at 199.



|  |  |
| --- | --- |
| **EX NO:** | **LINUX INSTALLATION USING VMWARE** |
| **DATE:** |

**AIM:**

To install the Linux OS using VMware

**Note**: If your Linux distribution is not RPM-based, has a custom kernel, or is unsupported, use the steps below to compile VMware Tools. To install VMware Tools in a Linux guest operating system using Compiler:

1. Ensure that your Linux virtual machine is powered on.
2. If you are running a GUI interface, open a command shell. **Note**: Log in as a root user, or use the sudo command to complete each of these steps.
3. Right Click **VM** in the virtual machine menu, then click **Guest > Install/Upgrade VMware Tools**.
4. Click **OK**. **Note**: In some cases, verify that the CDROM device is **Connected**

from within the **Edit Settings** option of the virtual machine.

1. To create a mount point, run: mkdir /mnt/cdrom
2. To mount the CDROM, run: mount /dev/cdrom /mnt/cdrom
3. To copy the Compiler gzip tar file to a temporary local directory, run: cp

/mnt/cdrom/VMwareTools-*version*.tar.gz /tmp/ Where *version* is the VMware Tools package version.

1. To determine the version of VMware tools, run: ls /mnt/cdrom You see output similar to: # VMwareTools-5.0.0-12124.tar.gz
2. To change to the tmp directory and extract the contents of the tar file into a new directory called vmware-tools-distrib, run:
3. To change directory to vmware-tools-distrib and run the vmware-install.pl PERL script to install VMware Tools, run: cd vmware-tools-distrib ./vmware-install.

###### Notes:

* + Complete the screen prompts to install the VMware Tools. Options in square brackets are default choices and can be selected by pressing **Enter**.
  + To compile VMware Tools successfully, you need gcc Compiler and Linux Kernel sources provided by your Linux distribution. Consult your Linux distribution documentation for details on methods to install these packages.
  + It is normal for the console screen to go blank for a short time during the installation when the display size changes.
  + Some warnings or errors are normal, like when a files does not exist.
  + Depending on the Linux distribution, your network service might restart after installation. VMware recommends that you invoke this command from the console and not remotely.

1. If you are running a GUI interface, restart your X Window session for any mouse or graphics changes to take effect.
2. To start VMware Tools running in the background during an X Window session, using terminal session run the command /usr/bin/vmware-toolbox.
3. Depending on your environment, you may need to unmount the CD-ROM. To unmount the CD-ROM, run: umount /mnt/cdrom
4. Depending on your environment, you may need to manually end the VMware Tools installation. To end the VMware Tools install, click **VM** in the virtual machine menu, then click **Guest > End VMware Tools Install**.
5. To remove VMware Tools installation packages, run: cd rm /tmp/VMwareTools-

*version*.tar.gz rm -rf /tmp/vmware-tools-distrib