LAB MANUAL

FOR

Operating System Concepts

**Lab Syllabus for Operating System Concepts**

**CMP-320**

## Course Outline

|  |  |
| --- | --- |
| Title | **Operating Systems Lab Manual** |
| **Credit Hours** | 1 |
| **Prerequisite** | - |
| Required Study Hours | 15 Lectures |
| **class** | BSSE |
| **Aims and Objectives** | Through this lab work the students will learn the practical implementation of algorithms to understand the basic concepts of operating systems. The course will use C++ and Java for labs and programming projects and teaches the implementation of main concepts of operating systems like CPU scheduling algorithms, Page replacement algorithms, process creation, threads etc. |
| **Learning Outcomes** | Upon successful completion of this course, the students will be able to:   1. Understand the implementation of basic concepts and algorithms of operating systems |
| **Text Book/s** | Operating System concepts by Silberchatz, Galvin, Gagne, (Tenth Edition) |
| **Reference Books** |  |
| **Instructional Aids/Resources** | |  |  |  |  | | --- | --- | --- | --- | | **Class Assessment 20%** | **Mid 30%** | **Final 50%** | **Total**  **100%** | | **Lab Work** |  |  |  | |  |  |  |  | |  |  |  |  | |
| **Assessment Criteria** |  |
| **Policies and Regulations** | **Class Attendance and Absenteeism**  Students are required to attend all classes and lab meetings. Regular attendance in their class/laboratory sessions will be very helpful to maintain a satisfactory progress throughout their course. Attendance will be strictly enforced and evaluated according to the Student Attendance Control Criteria announced by the DOCSIT and UOS. Any student who exceeds the maximum allowable absence limit during the course will not be allowed to sit in the exams. The maximum allowed limit for this course is 25% which include both excused and unexcused absences.  **Academic Integrity**  Cheating in any form will not be tolerated and could lead to severe consequences. Academic work submitted by the students in the form of homework, assignment, or a project must be the result of their own effort.  **Make-Up Exam Policy**  A student who has missed an exam will be allowed to sit in a make-up exam only if he or she provides a medical report from a government hospital/clinic.  **General Behavior**  Students must maintain a good behavior both in and outside their classes. They are required to keep their mobile phones switched off while attending their class/laboratory sessions or writing their exams. Any student who engages in a behavior that disrupts the learning environment may face disciplinary action under the UOS code. Students must also maintain a smoke free environment in all college facilities. |
| **Recommendations** |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Framework** | | | | | | |
| **Week** | | Topic | Source (Book-Chapter No. Article no.) | | Study Hours | |
| 1,2 | Process Creation   * Parent child processes concept * Creation of child process | | | Chapter 3 | | 2 |
| 3 | **Inter Process Communication**   * Sockets * 1 Handout | | | Chapter 3 | | 2 |
| 4,5 | **Threads**   * Java Threads | | | Chapter 4 | | 2 |
| 6 | **Implementation of Scheduling Algorithms**   * FIFO, SJF, Priority Scheduling | | | Chapter 5 | | 2 |
| 7,8 | **Implementation of Scheduling Algorithms**   * SRF, Round Robin | | | Chapter 5 | | 2 |
| 9 | **Deadlocks**   * Implementation of Banker’s Algorithm | | | Chapter 7 | | 2 |
| 10 | **Memory Management Algorithms**   * Implementation of first fit * Implementation of Best fit | | | Chapter 8 | | 2 |
| 11 | **Implementation of Page Replacement Algorithm**   * Implementation of FIFO, LRU | | | Chapter 9 | | 2 |
| 12 | **Implementation of Page Replacement Algorithm**   * Implementation of Page Buffering, LFU | | | Chapter 9 | | 2 |
| 13 | **Implementation of Page Replacement Algorithm**   * Implementation of Page Buffering, MFU | | | Chapter 9 | | 2 |

**Lab Session # 1,2**

In this lab session, you will learn about process creation, process control and process termination in windows. More specifically, we will focus on using the *CreateProcess* Win32 API. The following code implements process creation in Windows. Visual C++ 6.0 IDE is recommended for editing/compiling the program.

#include<stdio.h>

#include<windows.h>

int main()

{

STARTUPINFO si;

PROCESS\_INFORMATION pi;

// allocate memory

ZeroMemory(&si, sizeof (si)) ;

si.cb = sizeof (si) ;

ZeroMemory(&pi, sizeof(pi));

// create child process

if (!CreateProcess(NULL, // use command line

"C:\\WINDOWS\\system32\\mspaint.exe", // command lin

NULL, // don’t inherit process handle

NULL, // don’t inherit thread handle

FALSE, // disable handle inheritance

0, //no creation flags

NULL, // use parent’s environment block

NULL, // use parent’s existing directory

&si,

&pi))

{

fprintf(stderr, "Create Process Failed");

return -1;

}

// parent will wait for the child to complete

WaitForSingleObject(pi.hProcess, INFINITE);

printf("Child Complete");

// close handles

CloseHandle(pi.hProcess);

CloseHandle(pi.hThread);

}

The Procedure CreateProcess has the following form (you don’t have to implement CreateProcess).

BOOL CreateProcess(

LPCTSTR lpApplicationName, // name of executable module

LPTSTR lpCommandLine, // command line string

LPSECURITY\_ATTRIBUTES lpProcessAttributes, // SD

LPSECURITY\_ATTRIBUTES lpThreadAttributes, // SD

BOOL fInheritHandles, // handle inheritance option

DWORD dwCreationFlags, // creation flags

LPVOID lpEnvironment, // new envirnment block

LPCTSTR lpCurrentDirectory, // current directory name

LPSTARTUPINFO lpStartupInfo, // startup information

LPPROCESS\_INFORMATION lpProcessInformation // process information

);

**Part – I**

Implement the code segment given at the start of the first page in Visual C++. Create a win32 console application (empty project) names “ProcessCreate”. Then go to the file view and add a file process.c, implement the code in this file.

**Part – II**

Experiment by executing different types of applications using the child process e.g. notepad.

**Part – III**

Create a new win32 console application “Hello”. The application should simply display Hello World on the screen. Compile and build an executable file “Hello.exe” for this program. Place the executable in the same folder as the “ProcessCreate” source files. Then modify the “ProcessCreate” source code to execute the Hello world executable.

**Implemetaion of Grand Parent**

#include<stdio.h>

#include<windows.h>

void main()

{

STARTUPINFO si;

PROCESS\_INFORMATION pi;

//allocate memmory

ZeroMemory(&si,sizeof (si));

si.cb=sizeof (si);

ZeroMemory(&pi,sizeof(pi) );

printf("grand Parent process runs \n");

getch();

// create child process

if(!CreateProcess(NULL,//use command line

"E://Bushra Data//MS(QAU)//AOS//lab session//Lab 1//lab 1 process creation//GrandParent//Process.exe",//command line

NULL, // dont inherit process handle

NULL, //don not inherit thread handle

FALSE,//disable handle inheritance

0,//no creation flags

NULL,//use parents environment block

NULL,//use parent existing directory

&si,

&pi))

{

fprintf(stderr,"Crate Process Failed \n");

return -1;

}

//Parent will wait for child to complete

WaitForSingleObject(pi.hProcess, 500);

TerminateProcess(pi.hProcess,0);

printf("1st Child Complete \n");

CloseHandle(pi.hProcess);

CloseHandle(pi.hThread);

}

**Lab Session # 3**

**Sockets**

In this lab session, you will learn about sockets in java for inter-process communication.

import java.net.\*;

import java.io.\*;

public class DateServer

*{*

public static void main(String[] args) *{*

try *{*

ServerSocket sock = new ServerSocket(6013);

/\* now listen for connections \*/

while (true)

*{*

Socket client = sock.accept();

PrintWriter pout = new PrintWriter(client.getOutputStream(), true);

/\* write the Date to the socket \*/

pout.println(new java.util.Date().toString());

/\* close the socket and resume \*/

/\* listening for connections \*/

client.close();

*}*

*}*

catch (IOException ioe) *{*

System.err.println(ioe);

*}*

*}*

*}*

import java.net.\*;

import java.io.\*;

public class DateClient

*{*

public static void main(String[] args) *{*

try *{*

/\* make connection to server socket \*/

Socket sock = new Socket("127.0.0.1",6013);

InputStream in = sock.getInputStream();

BufferedReader bin = new BufferedReader(new InputStreamReader(in));

/\* read the date from the socket \*/

String line;

while ( (line = bin.readLine()) != null)

System.out.println(line);

/\* close the socket connection\*/

sock.close();

*}*

catch (IOException ioe) *{*

System.err.println(ioe);

*}*

*}*

**Assignment**

Implement an echo server

**Lab Session # 4**

**Threads**

In this lab session, you will learn about java threads.

// Create a second thread by extending Thread

class NewThread extends Thread {

NewThread()

{

// Create a new, second thread

super("Demo Thread");

System.out.println("Child thread: " + this);

start(); // Start the thread

}

// This is the entry point for the second thread.

public void run() {

try {

for(int i = 5; i > 0; i--) {

System.out.println("Child Thread: " + i);

Thread.sleep(500);

}

}

catch (InterruptedException e) {

System.out.println("Child interrupted.");

}

System.out.println("Exiting child thread.");

}

}

class ExtendThread

{

public static void main(String args[]) {

new NewThread(); // create a new thread

try {

for(int i = 5; i > 0; i--) {

System.out.println("Main Thread: " + i);

Thread.sleep(1000);

}

}

catch (InterruptedException e) {

System.out.println("Main thread interrupted.");

}

System.out.println("Main thread exiting.");

}

**Assignment**

Write down a program that creates 3 child threads.

**Lab Session # 5**

**Threads**

In this lab session, you will learn to write a java program for summation of non negative integers.

**Here is the code for creating named pipe**class Sum

{

private int sum;

public int getSum() {

return sum;

}

public void setSum(int sum) {

this.sum = sum;

}

}

class Summation implements Runnable

{

private int upper;

private Sum sumValue;

public Summation(int upper, Sum sumValue) {

this.upper = upper;

this.sumValue = sumValue;

}

public void run() {

int sum = 0;

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

sum += i;

sumValue.setSum(sum);

}

}

public class Driver

{

public static void main(String[] args) {

if (args.length > 0) {

if (Integer.parseInt(args[0]) < 0)

System.err.println(args[0] + " must be >= 0.");

else {

Sum sumObject = new Sum();

int upper = Integer.parseInt(args[0]);

Thread thrd = new Thread(new Summation(upper, sumObject));

thrd.start();

try {

thrd.join();

System.out.println

("The sum of "+upper+" is "+sumObject.getSum());

}

catch (InterruptedException ie) { }

}

}

else

System.err.println("Usage: Summation <integer value>"); }

}

**Practice Question**

**Q1.** Implement the code given above and show the output.

**Q2**. Modify the program so that we have only two threads created. The first should call print\_hello1 and the second should call print\_hello2. Both functions should have for loops for 100 times and should display line “Executing thread no. 1” or “Executing thread no. 2”. Is the output interleaved or not. Why?

**Q3**. Use two threads to show two rotating bars at the two top corners of a screen. Basically each thread will be driving the “rotation” of the bar. (Hint: You can use “\” and “/” to achieve the effect but you will have to figure out how to achieve the effect of “rotation”.)

**Q4**. For the program developed in Q3, modify it so that on the pressing “1” the left bar starts rotating i.e. thread 1 starts executing and on pressing “2”, the left bar stops and the right bar starts executing. If “1” is pressed again then the left bar starts rotating again but the right bar stops rotating. (Hint: Think in terms of suspend and resume)

**Lab Session # 6, 7**

Implementation of CPU Scheduling Algorithms

In this lab session, you will learn implementation of FIFO (non preemptive) CPU scheduling algorithm. The following code reads number of processes their cpu burst and displays average turnaround time and average waiting time. Visual C++ 6.0 IDE is recommended for editing/compiling the program(s).

Implementation of FCFS

#include<iostream>

using namespace std;

const int n=5;

class fifo

{

private:

int cpu[n],arr\_t[n],turn\_tm[n],wait\_tm[n];

int assign[n],comp[n];

float avg\_turn\_aroud\_time,avg\_waiting\_time;

public:

void get()

{

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

{

cout<<"enter the cpu burst of p"<<i<<"\tprocess";

cin>>cpu[i];

cout<<"enter the Arrival time of p"<<i<<"process\t";

cin>>arr\_t[i];

if(i==0)

{

assign[i]=arr\_t[i];

comp[i]=arr\_t[i]+cpu[i];

}

else

{

assign[i]=comp[i-1];

comp[i]=comp[i-1]+cpu[i];

}

}

cout<<"\n==================================================\n";

}

void turn\_around\_time()

{

float sum=0;

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

{

turn\_tm[i]=comp[i]-arr\_t[i];

sum=sum+turn\_tm[i];

}

avg\_turn\_aroud\_time=sum/n;

}

void waiting\_time()

{

float sum=0.0;

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

{

wait\_tm[i]=assign[i]-arr\_t[i];

sum=sum+wait\_tm[i];

}

avg\_waiting\_time=sum/n;

}

void show()

{

cout<<"Process \tcpu burst \tArrival Time\twaiting Time/tturn around time"<<endl;

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

{

cout<<"p"<<i<<"\t\t"<<cpu[i]<<"\t\t"<<arr\_t[i]<<"\t\t"<<wait\_tm[i]<<"\t\t"<<turn\_tm[i]<<endl;

cout<<endl;

}

cout<<"The average waiting time is ="<<avg\_waiting\_time;

cout<<endl;

cout<<"The average turn around time is ="<<avg\_turn\_aroud\_time<<endl;

}

};

int main()

{

fifo f;

f.get();

f.waiting\_time();

f.turn\_around\_time();

f.show();

return 0;

}

**Exercise**

Implement SJF and Priority scheduling algorithm. Read no of processes their arrival time cpu burst and priority from user. The program should compute the average turnaround time and average waiting time

**Lab Session # 8**

Implementation of CPU Scheduling Algorithms

In this lab session, you will learn implementation of Round Robin (preemptive) CPU scheduling algorithm. The following code reads number of processes their cpu burst and displays average turnaround time and average waiting time. Visual C++ 6.0 IDE is recommended for editing/compiling the program(s).

Implementation of Round Robin

#include<iostream>

using namespace std;

struct p

{

int AT;

int processno;

int cpuburst;

int complationtime;

int turnaround;

};

void main()

{

p arr[5];

int i=0;

p newi[5];

int j;

int total=0; // Cpu burst by user......

int x;

{ cout<<endl;

cout<<endl;

cout<<"\t\t..........WELCOME TO ROUND ROBIN ALGO..........."<<endl;

cout<<endl;

cout<<endl;

cout<<endl;

}

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

{

cout<<"\t\tEnter process no "<<i<<" :";

cin>>arr[i].processno;

cout<<"\t\tEnter cpu burst for :";

cin>>arr[i].cpuburst;

cout<<"\t\tEnter arrival time for :";

cin>>arr[i].AT;

cout<<"\t\t.............................................."<<endl;

cout<<endl;

}//end

cout<<endl;

cout<<endl;

cout<<endl;

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

{

total=arr[i].cpuburst+total;

}// Find total cpu burst.....

int complete=0;

newi[0]=arr[0];

newi[1]=arr[0];

i=0; x=0;

{

do

{

for(j=0;j<5;j++)

{

if(newi[j].cpuburst==0)

{

complete=complete+0;

}

else if(newi[j].cpuburst -1==0)

{

newi[j].cpuburst=newi[j].cpuburst-1;

complete=complete+1;

newi[j].complationtime=complete;

}

else if(newi[j].cpuburst -2==0)

{

newi[j].cpuburst=newi[j].cpuburst-2;

complete=complete+2;

newi[j].complationtime=complete;

}

else

{

newi[j].cpuburst=newi[j].cpuburst-3;

complete=complete+3;

newi[j].complationtime=complete;

}

if(i==4)

{

newi[i-4]=arr[x+1];

i++;

}// last cheker......................

if(complete>=arr[x+1].AT && i<4)

{

while(complete>=arr[x+1].AT && i<4)

{

newi[i+1]=arr[x+1];

i++; x++;

}

if(i==4)

{newi[i-4]=arr[x+1];

}

else

{

newi[i+1]=newi[j];

i++;

}

}//end of if/////

}//end for

}while(total>complete);//end of complete while...

}

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

{ newi[i].turnaround=newi[i].complationtime-newi[i].AT;

}

{

cout<<" PROCESS #:"<<"\tCPU BURST"<<"\tARRIVAL TIME"<<"\tTURN ARONUD TIME"<<endl;

}

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

{

cout<<"P"<<newi[i].processno<<":"<<"\t\t"<<newi[i].cpuburst<<"\t\t"<<newi[i].AT<<"\t\t"<<newi[i].turnaround<<endl;

}

}

Implement SRF (preemptive)scheduling algorithm. Read no of processes their arrival time cpu burst and priority from user. The program should compute the average turnaround time and average waiting time

**Lab Session # 9**

Banker’s Algorithm

In this lab session, you will learn the banker’s algorithm to check the safe state and resource allocation algorithm’s implementation. Following code reads the number of processes, required resources, systems maximum resources. Then it gets requirements of process and after checking safe state allocate resources. Visual C++ 6.0 IDE is recommended for editing/compiling the program(s).

#include<iostream>

using namespace std;

int main()

{

int i,j,p,r,all[3][3],max[3][3],avail[3],need[3][3];

cout<<"\t Enter the number of process: ";

cin>>p;

cout<<"\n\t Enter the number of resources: ";

cin>>r;

cout<<"\n\tEnter the allocation table:\n";

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

{

cout<<"\tEnter the detail for process p"<<i+1<<"\n";

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

{

cout<<"\t\tEnter resources R"<<j+1<<"\t";

cin>>all[i][j];

}

}

cout<<"\tEnter the max table: \n";

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

{

cout<<"\tEnter the detail for process p"<<i+1<<"\n";

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

{

cout<<"\t\tEnter resources R"<<j+1<<"\t";

cin>>max[i][j];

}

}

cout<<"\tEnter the available table: \n";

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

{

cout<<"\t\tEnter resources R"<<i+1<<"\t";

cin>>avail[i];

}

cout<<"The Remaining Needs Are :\n";

cout<<"\t";

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

{

cout<<"\tR"<<i+1<<" ";

}

cout<<"\n";

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

{

cout<<" process p"<<i+1<<" ";

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

{

need[i][j]=max[i][j]-all[i][j];

cout<<need[i][j]<<" ";

}

cout<<"\n";

}

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

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

if(need[i][j]<0)

cout<<"\n\n\t\t The Process P "<<i+1<<" is in UnSafe State:"<<endl;

else

cout<<"\n\n\t\t The Process P "<<i+1<<" is in Safe State:"<<endl;

return 0;

}

**Exercise**

Implement the deadlock detection algorithm for n no of processes after reading maximum, allocated resources and remaining needs.

**Lab Session # 11**

**Implementation of Best Fit and First Fit Memory Management Algorithms**

In this lab session, you will learn the implementation of best fit and first fit memory management algorithms. Visual C++ 6.0 IDE is recommended for editing/compiling the program(s).

#include<iostream.h>

#include<conio.h>

#include<string.h>

void main()

{

unsigned int n,j,i,size[10],m,x=0,t;

int cho=1,ch;

clrscr();

cout<<”\t\t STORAGE PLACEMENT STRATEGIES\n”;

cout<<”\tEnter the number of holes:”;

cin>>n;

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

{

cout<<”\n Enter The Size Of Hole “<<i<<”:”;

cin>>size[i];

}

while(cho==1)

{

cout<<”\nEnter the size of the incoming program:”;

cin>>m;

cout<<”\nMenu”;

cout<<”\n1.First Fit Strategy \n2.Best Fit Strategy”;

cout<<”\n Enter your choice:”;

cin>>ch;

x=0;

switch(ch)

{

case 1:

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

{

if(size[i]>=m)

{

cout<<”\nYour program is placed in hole “<<i;

size[i]-=m;

x=i;

break;

}

}

if(x==0) cout<<”There is no room for your program.”;

break;

case 2:

unsigned int temp=0,pos=0,t1;

if(m<=size[1])

{

temp=size[1]-m;

pos=1;

}

else

temp=size[1];

for(i=2;i<=n;i++)

{

if(m<=size[i])

{

t1=size[i]-m;

if(temp>=t1)

{

temp=t1;

pos=i;

}

}

else  temp=size[i];

}

if(pos==0)

cout<<”There is no room for your page.”;

else

{

size[pos]=size[pos]-m;

cout<<”Your program is palced in hole “<<pos;

}

break;

case 4:

return;

}

cout<<”\nFree Storage List”;

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

cout<<”\nHole “<<i<<”\t\t”<<size[i];

cout<<”\n\nPress 1 to continue:”;

cin>>cho;

}

}

**Exercise**

Implement best fit and first fit Memory management algorithme.

**Lab Session # 12**

Page Replacement Algorithm

In this lab session, you will learn the implementation of page replacement algorithms. Following code reads the number of free frames and a reference string. At the end it will display the number of page faults. Visual C++ 6.0 IDE is recommended for editing/compiling the program(s).

**Implementation of LRU Page Replacement Algorithm**

#include<iostream>

#include<iomanip>

using namespace std;

struct PriorityString

{

int PageNo;

PriorityString\* link;

};

struct Frame

{

int PageNo;

Frame\* left, \*right;

};

class LeastRecentlyUsed

{

private:

bool found;

int PageFault;

int TotalFrame;

///////Pointers////////

PriorityString\* head, \*last;

Frame\* RecentlyUsed, \*leastRecentlyUsed;

public:

LeastRecentlyUsed(void);

void Welcome(void);

void TakeFrame(void);

void ShowFault(void);

void FindFault(void);

void moveDownAndReplace(Frame\* FindFrame, PriorityString\* data);

void EnterPriorityString(void);

void ShowPriorityString(void);

virtual ~LeastRecentlyUsed(void);

};

LeastRecentlyUsed::LeastRecentlyUsed(void)

{

found = false;

PageFault = 0;

TotalFrame = 0;

head = NULL;

last = NULL;

RecentlyUsed = NULL;

leastRecentlyUsed = NULL;

}

void LeastRecentlyUsed::moveDownAndReplace(Frame \* FindFrame, PriorityString\* data)

{

Frame\* temp = FindFrame;

Frame\* temp1 = FindFrame->left;

cout << setw(25) << "Recently used" << setw(25) << "Previuosly used" << setw(25) << "Least recently Used" << endl;

cout << setw(25) << RecentlyUsed->PageNo << setw(25) << RecentlyUsed->right->PageNo << setw(25) << leastRecentlyUsed->PageNo << endl;

while(temp1 != NULL)

{

temp->PageNo = temp1->PageNo;

temp = temp1;

temp1 = temp1->left;

}

RecentlyUsed->PageNo = data->PageNo;

cout << setw(25) << RecentlyUsed->PageNo << setw(25) << RecentlyUsed->right->PageNo << setw(25) << leastRecentlyUsed->PageNo << endl;

}

void LeastRecentlyUsed::FindFault(void)

{

PriorityString\* temp = head;

Frame\* FindFrame;

while(head != NULL)

{

temp = head;

head = head->link;

FindFrame = RecentlyUsed;

while(FindFrame->PageNo != -999 && FindFrame->right != NULL)

{

if(FindFrame->PageNo == temp->PageNo)

{

found = true;

break;

}

else

{

FindFrame = FindFrame->right;

}

}//end inner while

if(FindFrame == leastRecentlyUsed && FindFrame->PageNo == temp->PageNo)

{

found = true;

}

moveDownAndReplace(FindFrame,temp);

//RecentlyUsed->PageNo = temp->PageNo;

delete temp;

if(found == false)

{

PageFault++;

}

else

found = false;

}// end outer while

}

void LeastRecentlyUsed::TakeFrame(void)

{

cout << "How many free frame you want to take for process:\_ ";

cin >> TotalFrame;

cout << endl;

Frame \* temp = new Frame;

RecentlyUsed = temp;

RecentlyUsed->PageNo = -999;

leastRecentlyUsed = temp;

temp->left = NULL;

temp->right = NULL;

for(int i = 1; i < TotalFrame; i++)

{

temp = new Frame;

temp->right = NULL;

leastRecentlyUsed->right = temp;

temp->left = leastRecentlyUsed;

leastRecentlyUsed = temp;

leastRecentlyUsed->PageNo = -999;

}

}

void LeastRecentlyUsed::ShowFault(void)

{

cout << "Total page faults are:\_ "

<< PageFault << endl;

}

void LeastRecentlyUsed::EnterPriorityString(void)

{

int length;

cout << "Please specify length of PriorityString..." ;

cin >> length;

cout << endl;

PriorityString\* temp;

int i = 0;

while(i < length )

{

temp = new PriorityString;

cout << "Please enter required page number:\_ ";

cin >> temp->PageNo;

temp->link = NULL;

cout << endl;

if(head == NULL)

{

head = temp;

last = head;

}

else

{

last->link = temp;

last = temp;

}

i++;

}// end while

}

void LeastRecentlyUsed::ShowPriorityString(void)

{

PriorityString\*temp = head;

while(temp!=NULL)

{

cout << temp->PageNo << " ";

temp = temp->link;

}

cout << endl;

}

LeastRecentlyUsed::~LeastRecentlyUsed(void)

{

PriorityString\* temp = head;

while(head != NULL)

{

head = head->link;

delete temp;

temp = head;

}

Frame\* tempframe = RecentlyUsed;

while(tempframe != NULL)

{

RecentlyUsed = RecentlyUsed->right;

if(RecentlyUsed != NULL)

{

RecentlyUsed->left = tempframe->left;

}

delete tempframe;

tempframe = RecentlyUsed;

}

}

int main()

{

LeastRecentlyUsed pro;

pro.EnterPriorityString();

pro.ShowPriorityString();

pro.TakeFrame();

pro.FindFault();

pro.ShowFault();

return 0;

}

**Exercice**

Implement FIFO page replacement algorithm. Code should read the number of free frames and a reference string. At the end it should display the number of page faults.

**Lab Session # 13, 14**

Page Replacement Algorithm

In this lab session, you will learn the implementation of Page Buffering page replacement algorithms. Following code reads the number of free frames and a reference string. At the end it will display the number of page faults. Visual C++ 6.0 IDE is recommended for editing/compiling the program(s).

Implementation of Page Buffering

#include<iostream>

#include<string>

using namespace std;

# define frame 3;

int main()

{

string s;

int array[100],proc[4],buffer,i,len,count,fault=0,j,k,temp;

char cha[5];

cout<<"NO OF FRAMES = 3"<<endl;

cout<<"ENTER STRING for the sequence of pages"<<endl;

getline(cin,s);

len=s.length();

//making ints of string

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

{

cha[0]=s[i];

array[i]=atoi(cha);

cout<<array[i]<<" ";

strset(cha,NULL);

}

cout<<endl;

count=-1;

//processing

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

{

if(count<2)

{

count++;

proc[count]=array[i];

fault++;

}

else

{

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

{

if(array[i]==proc[j])

{

break;

}

}

if(j>2) //page is not in ram

{

if(buffer!=-1)

{

if(array[i]!=buffer)

fault++;

}

buffer=proc[0];

for(k=0;k<2;k++)

{

proc[k]=proc[k+1];

}

proc[2]=array[i];

}

}//else

}

cout<<"\nNo of faults= "<<fault<<endl<<endl;

return 0;

}

**Exercice**

Implementation LFU and MFU page replacement algorithm. Code should read the number of free frames and a reference string. At the end it should display the number of page faults.