Distributed Computing

(CDCSC15)

PRACTICAL FILE

Submitted by:
Vyom Kaushik

2020UCD2106

CSDS-1(Semester V)

INDEX

Practical 1

Aim: Program to implement Lamport's Logical Clock

Code:

```
#include <bits/stdc++.h>
using namespace std;

int max1(int a, int b)
{
    if(a>b){
        return a;
    }
    return b;
}

void display(int e1, int e2,
             int p1[5], int p2[3])
{
    int i;

    cout <<endl<<"The time stamps of events in P1:"<<endl;

    for (i = 0; i < e1; i++) {
        cout << p1[i] << " ";
    }

    cout <<endl<< "The time stamps of events in P2:"<<endl;

    for (i = 0; i < e2; i++)
        cout << p2[i] << " ";
}

void lamportLogicalClock(int e1, int e2,
                         int m[5][3])
{
    int i, j, k, p1[e1], p2[e2];

    // Initialize p1[] and p2[]
    for (i = 0; i < e1; i++)
        p1[i] = i + 1;

    for (i = 0; i < e2; i++)
        p2[i] = i + 1;
    cout << "\t";
    for (i = 0; i < e2; i++)
        cout << "\te2" << i + 1;

    for (i = 0; i < e1; i++) {

        cout << endl<<" e1" << i + 1 << "\t";

        for (j = 0; j < e2; j++)
            cout << m[i][j] << "\t";
    }

    for (i = 0; i < e1; i++) {
        for (j = 0; j < e2; j++) {

            if (m[i][j] == 1) {
                p2[j] = max1(p2[j], p1[i] + 1);
                for (k = j + 1; k < e2; k++)
                    p2[k] = p2[k - 1] + 1;
            }
```

```
                if (m[i][j] == -1) {
                    p1[i] = max1(p1[i], p2[j] + 1);
                    for (k = i + 1; k < e1; k++)
                        p1[k] = p1[k - 1] + 1;
                }
            }
        }

        // Function Call
        display(e1, e2, p1, p2);
    }


    int main()
    {
        int e1 = 5, e2 = 3, m[5][3];

        m[0][0] = 0;
        m[0][1] = 0;
        m[0][2] = 0;
        m[1][0] = 0;
        m[1][1] = 0;
        m[1][2] = 1;
        m[2][0] = 0;
        m[2][1] = 0;
        m[2][2] = 0;
        m[3][0] = 0;
        m[3][1] = 0;
        m[3][2] = 0;
        m[4][0] = 0;
        m[4][1] = -1;
        m[4][2] = 0;

        // Function Call
        lamportLogicalClock(e1, e2, m);

        return 0;
    }
```

OUTPUT

```
        e21    e22    e23
 e11    0      0      0
 e12    0      0      1
 e13    0      0      0
 e14    0      0      0
 e15    0      -1     0
The time stamps of events in P1:
1 2 3 4 5
The time stamps of events in P2:
1 2 3
```

Practical 2

Aim: Lamport's Vector Clock

Code:

```
#include<iostream>
#include<stdio.h>
#define SIZE 10
using namespace std;

class node {
	public:
	int data[SIZE];
	node *next;

		node() {
			for(int p=0; p<SIZE; p++) {
				data[p] = 0;
			}
			next = NULL;
		}

		node(int v[], int n1) {
			for(int s = 0; s < n1; s++) {
				data[s] = v[s];
			}
			next = NULL;
		}

		friend class process;
}*start=NULL;

int main() {
	int n, events, sent, receive, sentE, recE, commLines = 0;
	node *temp;
	node *proc[SIZE];   //array of processes
	cout<<"Enter no. of processes: ";
```

```
cin>>n;
int vector[n];      //representation of data
for(int i=0;i<n;i++)vector[i]=0;
/*----------------INITIALIZATION LOOP-------------------------*/
for(int i = 0; i < n; i++) {      //number of processes
       for(int v = 0; v < n; v++) {
              vector[v] = 0;
       }

       cout<<"Enter no. of events in process "<<i+1<<": ";
       cin>>events;

       for(int j = 1; j <= events; j++) {
              vector[i] = j;
              node *newnode = new node(vector,n);
              if(start == NULL) {
                     start = newnode;
                     temp = start;
              } else {
                     temp->next = newnode;
                     temp = temp->next;
              }
       }
       proc[i] = start;
       start = NULL;
}

/*-------------------DATA GATHERING--------------------*/
cout<<"\nEnter the number of communication lines: ";
cin>>commLines;
node *tempS, *tempR;

for(int i = 0; i < commLines; i++) {
       cout<<"\nEnter the sending process: ";
       cin>>sent;
       cout<<"\nEnter the receiving process: ";
       cin>>receive;
```

```
            cout<<"\nEnter the sending event number: ";
            cin>>sentE;
            cout<<"\nEnter the receiving event number: ";
            cin>>recE;

            tempS = proc[sent - 1];
            tempR = proc[receive - 1];

            for(int j = 1; j < sentE; j++)
                  tempS = tempS->next;

            for(int j = 1; j < recE; j++)
                  tempR = tempR->next;

            for(int j = 0; j < n; j++) {
                  tempR->data[j] = (tempR->data[j] < tempS->data[j]) ? tempS-
>data[j] : tempR->data[j];
            }
      }

      cout<<"\nThe resulting vectors are:"<<endl<<endl;
      for(int k = 0; k < n; k++) {
            cout<<"Process "<<k + 1<<": ";

            node *temp1 = proc[k];
            while(temp1) {
                  cout<<"(";
                  for(int f = 0; f < n - 1; f++)
                        cout<<temp1->data[f]<<",";

                  cout<<temp1->data[n-1];
                  cout<<")";
                  temp1 = temp1->next;
            }
            cout<<endl;
      }
```

```
        return 0;
}
```

OUTPUT

```
Enter no. of processes: 3
Enter no. of events in process 1: 2
Enter no. of events in process 2: 3
Enter no. of events in process 3: 1

Enter the number of communication lines: 3

Enter the sending process: 3

Enter the receiving process: 2

Enter the sending event number: 1

Enter the receiving event number: 1

Enter the sending process: 1

Enter the receiving process: 2

Enter the sending event number: 1

Enter the receiving event number: 2

Enter the sending process: 2

Enter the receiving process: 1

Enter the sending event number: 3

Enter the receiving event number: 2

The resulting vectors are:

Process 1: (1,0,0)(2,3,0)
Process 2: (0,1,1)(1,2,0)(0,3,0)
Process 3: (0,0,1)
```

Practical 3

Aim: Distributed Mutual Exclusion (Non-Token Based)

Code:

```
#include<bits/stdc++.h>
using namespace std;

int main()
{
    int ns,ncs,timestamp,site;
    cout<<"Enter number of sites :";
    cin>>ns;
    cout<<"Enter number of sites which want to enter critical section:";
    cin>>ncs;
    vector<int> ts(ns,0);
    vector<int> request;
    map <int,int> mp;
    for(int i=0;i<ncs;i++)
    {
        cout<<"\nEnter timestamp:";
        cin>>timestamp;
        cout<<"Enter site number:";
        cin>>site;
        ts[site-1]=timestamp;
        request.push_back(site);
        mp[timestamp]=site;
    }

    cout<<"\nSites and Timestamp:\n";
    for(int i=0;i<ts.size();i++)
    {
        cout<<i+1<<" "<<ts[i]<<endl;
    }

    for(int i=0;i<request.size();i++)
    {
        cout<<"\n Request from site:"<<request[i]<<endl;
        for(int j=0;j<ts.size();j++)
        {
            if(request[i]!=(j+1))
            {
                if(ts[j]>ts[request[i]-1] || ts[j]==0)
                    cout<<j+1<<" Replied\n";
                else
                    cout<<j+1<<" Deferred\n";
            }
        }
    }

    cout<<endl;
    map<int,int>:: iterator it;
    it=mp.begin();
    int c=0;
    for(it=mp.begin();it!=mp.end();it++)
    {
        cout<<"Site "<<it->second<<" entered Critical Section \n";
        if(c==0)
            cout<<"\nSimilarly,\n\n";
        c++;
    }
    return 0;
}
```

OUTPUT

```
Enter number of sites :5
Enter number of sites which want to enter critical section:3

Enter timestamp:2
Enter site number:2

Enter timestamp:3
Enter site number:3

Enter timestamp:1
Enter site number:4

Sites and Timestamp:
1 0
2 2
3 3
4 1
5 0

 Request from site:2
1 Replied
3 Replied
4 Deferred
5 Replied

 Request from site:3
1 Replied
2 Deferred
4 Deferred
5 Replied

 Request from site:4
1 Replied
2 Replied
3 Replied
5 Replied

Site 4 entered Critical Section

Similarly,

Site 2 entered Critical Section
Site 3 entered Critical Section
```

Practical 4

Aim: Remote Procedure Call (RPC)

Code:

IDL.x

```
struct values{
	float num1;
	float num2;
	char operation;
};
program COMPUTE {
	version COMPUTE_VERS {
		float ADD(values) = 1;
		float SUB(values) = 2;
	} = 3;
} = 4;
```

IDL_client.c

```
/*
* This is sample code generated by rpcgen.
* These are only templates and you can use them
* as a guideline for developing your own functions.
*/
#include "IDL.h"
float compute_3(char *host, float a, float b, char op)
{
	CLIENT *clnt;
	float *result_1;
	values add_3_arg;
	float *result_2;
	values sub_3_arg;
	if(op == '+') {
		add_3_arg.num1 = a;
		add_3_arg.num2 = b;
		add_3_arg.operation = op;
```

```
		clnt = clnt_create (host, COMPUTE, COMPUTE_VERS, "udp");
		if (clnt == NULL) {
			clnt_pcreateerror (host);
			exit (1);
		}
		result_1 = add_3(&add_3_arg, clnt);
		if (result_1 == (float *) NULL) {

		}
		clnt_destroy (clnt);

		return (*result_1);
	}
	else if (op == '-') {
		sub_3_arg.num1 = a;
		sub_3_arg.num2 = b;
		sub_3_arg.operation = op;
		clnt = clnt_create (host, COMPUTE, COMPUTE_VERS, "udp");
		if (clnt == NULL) {
			clnt_pcreateerror (host);
			exit (1);
		}
		result_2 = sub_3(&sub_3_arg, clnt);
		if (result_2 == (float *) NULL) {
			clnt_perror (clnt, "call failed");
		}
		clnt_destroy (clnt);
		return (*result_2);
	}
}
int main (int argc, char *argv[])
{
	char *host;
	float number1, number2;
	char oper;
	printf("Enter the first number: ");
	scanf("%f", &number1);
```

```
        printf("Enter the second number: ");
        scanf("%f", &number2);
        printf("Enter the operator: ");
        scanf("%s", &oper);
        host = argv[1];
        printf("Answer = %f\n", compute_3(host, number1, number2, oper));
        exit (0);
}
```

IDL_server.c

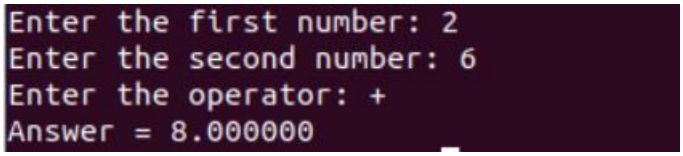
```
/*
* This is sample code generated by rpcgen.
* These are only templates and you can use them
* as a guideline for developing your own functions.
*/
#include "IDL.h"
float *
add_3_svc(values *argp, struct svc_req *rqstp)
{
        static float result;
        result = argp->num1 + argp->num2;
        return &result;
}
float *
sub_3_svc(values *argp, struct svc_req *rqstp)
{
        static float result;
        result = argp->num1 - argp->num2;
        return &result;
}
```

IDL_server.c

```
/*
* This is sample code generated by rpcgen.
* These are only templates and you can use them
* as a guideline for developing your own functions.
```

```
*/
#include "IDL.h"
float *
add_3_svc(values *argp, struct svc_req *rqstp)
{
	static float result;
	result = argp->num1 + argp->num2;
	return &result;
}
float *
sub_3_svc(values *argp, struct svc_req *rqstp)
{
	static float result;
	result = argp->num1 - argp->num2;
	return &result;
}
```

OUTPUT



Practical 5

Aim: Remote Method Invocation (RMI)

Code:

We have implemented a Calculator using RMI. The implementation of this program consists of four Java files, namely CalculatorImpl.java, CalculatorServer.java, Calculator.java and CalculatorClient.java.

Calculator.java:

```
// This is an interface

import java.rmi.*;

public interface Calculator extends Remote {

	public long add(long a, long b) throws RemoteException;

	public long sub(long a, long b) throws RemoteException;

	public long mul(long a, long b) throws RemoteException;

	public long div(long a, long b) throws RemoteException;

}
```

CalculatorImpl.java:

```
// This is the implementation file of calculator methods

public class CalculatorImpl extends java.rmi.server.UnicastRemoteObject implements

Calculator {

public CalculatorImpl() throws java.rmi.RemoteException {

	super();

}

public long add(long a, long b) {

	return a + b;

}

public long sub(long a, long b) {

	return a - b;

}

public long mul(long a, long b) {

	return a * b;

}

public long div(long a, long b) {

	return a / b;

}

}
```

CalculatorServer.java

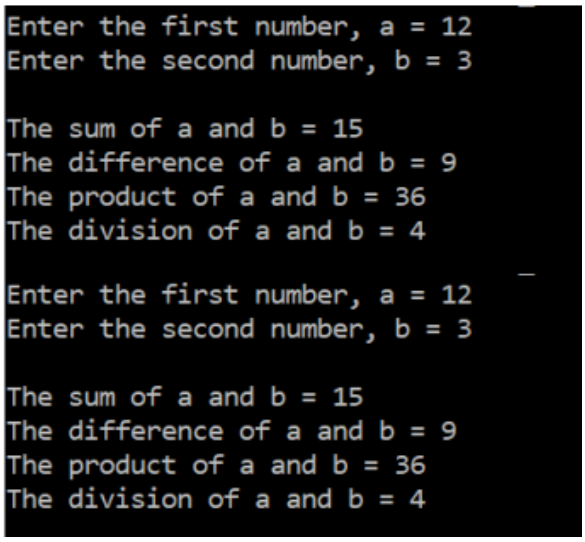
```
// This is the server implementing the calculator
import java.rmi.Naming;
public class CalculatorServer {
	public CalculatorServer() {
		try {
			Calculator c = new CalculatorImpl();
			Naming.rebind("rmi://localhost:1099/CalculatorService", c);
		} catch (Exception e) {
			System.out.println("Trouble: " + e);
		}
}
public static void main(String args[]) {
	new CalculatorServer();
	}
}
```

CalculatorClient.java

```
// This is the actual client program using Calculator interface
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.net.MalformedURLException;
import java.rmi.NotBoundException;
import java.util.*;
public class CalculatorClient {
	public static void main(String[] args) {
		try {
				Calculator c = (Calculator)
Naming.lookup("rmi://localhost/CalculatorService");
				Scanner sc = new Scanner(System.in);
				System.out.print("Enter the first number, a = ");
				int a = sc.nextInt();
				System.out.print("Enter the second number, b = ");
				int b = sc.nextInt();
				sc.close();
				System.out.println();
				System.out.print("The sum of a and b = ");
				System.out.println(c.add(a, b));
```

```
                        System.out.print("The difference of a and b = ");
                        System.out.println(c.sub(a, b));
                        System.out.print("The product of a and b = ");
                        System.out.println(c.mul(a, b));
                        System.out.print("The division of a and b = ");
                        System.out.println(c.div(a, b));
                    }
                    catch (MalformedURLException murle) {
                        System.out.println();
                        System.out.println("MalformedURLException");
                        System.out.println(murle);
                    }
                    catch (RemoteException re) {
                        System.out.println();
                        System.out.println("Remote Exception");
                        System.out.println(re);
                    }
                    catch (NotBoundException nbe) {
                        System.out.println();
                        System.out.println("NotBoundException");
                        System.out.println(nbe);
                    }
                    catch (java.lang.ArithmeticException ae) {
                        System.out.println();
                        System.out.println("java.lang.ArithmeticException");
                        System.out.println(ae);
                }
            }
        }
```

OUTPUT


Enter the first number, a = 12
Enter the second number, b = 3
The sum of a and b = 15
The difference of a and b = 9
The product of a and b = 36
The division of a and b = 4
Enter the first number, a = 12
Enter the second number, b = 3
The sum of a and b = 15
The difference of a and b = 9
The product of a and b = 36
The division of a and b = 4

Practical 6

Aim: Remote Deadlock Detection

Code:

```
#include <bits/stdc++.h>

using namespace std;

bool deadlock(int start, int current, vector<vector<bool>> &depends, vector<bool>
&visited, vector<int> &site)

{

     if (visited[current])

         return true;

         visited[current] = true;

         for (int i = 0; i < depends[current].size(); i++)

         {

             if (!depends[current][i])

                continue;

             if (i == current)

                return true;

             if (site[current] != site[i])

             cout << "Probe is sent: (" << start + 1 << "," << current + 1 << ","
<< i + 1 << ")" << endl;

             return deadlock(start, i, depends, visited, site);


     }

     return false;

}

int main()

{

     int sites;

     vector<int> site;

     cout << "Enter number of sites: " << endl;

     cin >> sites;

     int total_no_of_events = 0;

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

     {

         int events;
```

```
        cout << "Enter number of events in site " << i + 1 << ": " << endl;
        cin >> events;
        for (int j = 0; j < events; j++)
            site.push_back(i);
        total_no_of_events += events;
    }
    cout << "So, we have " << sites << " sites and " << total_no_of_events << "
events numbered " << endl;
    for (int i = 1; i <= total_no_of_events; i++)
       cout << i << " ";
    cout << endl;
    vector<vector<bool>> depends(total_no_of_events,
vector<bool>(total_no_of_events, false));
    int m;
    cout << "Enter the no. of dependencies: " << endl;
    cin >> m;
    for (int i = 0; i < m; i++)
    {
        int a, b;
        cout << "Enter the Dependencies (If event 1 depends on event 2, enter 1
2):" << endl;
        cin >> a >> b;
        depends[a - 1][b - 1] = true;
    }
    cout << "Enter the Node to Start Probe: " << endl;
    int start;
    cin >> start;
    start--;
    vector<bool> visited(total_no_of_events, false);
    if (deadlock(start, start, depends, visited, site))
        cout << "A Deadlock exists" << endl;
    else
       cout << "No Deadlock doesn't exist" << endl;
    return 0;
}
```

OUTPUT

```
Enter number of sites:
2
Enter number of events in site 1:
3
Enter number of events in site 2:
2
So, we have 2 sites and 5 events numbered
1 2 3 4 5
Enter the no. of dependencies:
6
Enter the Dependencies (If event 1 depends on event 2, enter 1 2):
1 2
Enter the Dependencies (If event 1 depends on event 2, enter 1 2):
2 3
Enter the Dependencies (If event 1 depends on event 2, enter 1 2):
3 4
Enter the Dependencies (If event 1 depends on event 2, enter 1 2):
2 4
Enter the Dependencies (If event 1 depends on event 2, enter 1 2):
4 5
Enter the Dependencies (If event 1 depends on event 2, enter 1 2):
5 1
Enter the Node to Start Probe:
1
Probe is sent: (1,3,4)
Probe is sent: (1,5,1)
A Deadlock exists
```

Practical 7

Aim: Locking Algorithm

Code:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
     int lock = 0;
     bool locked_T1 = false, locked_T2 = false;
     while (1)
     {
          int choice;
          if (!locked_T1)
          {
               cout << "Transaction T1 wants to Lock Data Object ?" << endl;
               cout << "1. Yes" << endl;
               cout << "2. No" << endl;
               cin >> choice;
               if (!lock && choice == 1)
               {
                    lock = 1;
                    locked_T1 = true;
                    cout << "T1 has been given the Lock for the Data Object" << endl;
               }
               else if (lock)
                    cout << "\nData Object is Already Locked by " << (locked_T1 ? "T1"
: "T2") << endl
               << endl;
          }
          if (!locked_T2)
          {
               cout << "Transaction T2 wants to Lock Data Object ?" << endl;
               cout << "1. Yes" << endl;
               cout << "2. No" << endl;
               cin >> choice;
               if (!lock && choice == 1)
```

```
            {
                lock = 1;
                locked_T2 = true;
                cout << "T2 has been given the Lock for the Data Object" << endl;
            }
            else if (lock)
               cout << "\nData Object is Already Locked by " << (locked_T1 ? "T1"
: "T2") << endl << endl;
         }
         if (lock)
         {
            if (locked_T1)
            {
                cout << "Transaction T1 wants to Release the Lock on Data Object?"
<< endl;
                cout << "1. Yes" << endl;
                cout << "2. No" << endl;
                cin >> choice;
                if (choice == 1)
                {
                    lock = 0, locked_T1 = false;
                    cout << "The Lock on Data Object has been released by T1 !" <<
endl;
                }
            }
            else
            {
                cout << "Transaction T2 wants to Release the Lock on Data Object?"
<< endl;
                cout << "1. Yes" << endl;
                cout << "2. No" << endl;
                cin >> choice;
                if (choice == 1)
                {
                    lock = 0, locked_T2 = false;
                    cout << "The Lock on Data Object has been released by T2 !" <<
endl;
                }
            }
```

```
        }
    }
    return 0;
}
```

OUTPUT

```
Transaction T1 wants to Lock Data Object ?
1. Yes
2. No
1
T1 has been given the Lock for the Data Object
Transaction T2 wants to Lock Data Object ?
1. Yes
2. No
1

Data Object is Already Locked by T1

Transaction T1 wants to Release the Lock on Data Object?
1. Yes
2. No
2
Transaction T2 wants to Lock Data Object ?
1. Yes
2. No
1

Data Object is Already Locked by T1

Transaction T1 wants to Release the Lock on Data Object?
1. Yes
2. No
1
The Lock on Data Object has been released by T1 !
Transaction T1 wants to Lock Data Object ?
1. Yes
2. No
2
Transaction T2 wants to Lock Data Object ?
1. Yes
2. No
1
T2 has been given the Lock for the Data Object
Transaction T2 wants to Release the Lock on Data Object?
1. Yes
2. No
1
The Lock on Data Object has been released by T2 !
Transaction T1 wants to Lock Data Object ?
1. Yes
2. No
1
T1 has been given the Lock for the Data Object
```