

Smt. Chandibai Himathmal Mansukhani College

Contents

USCS3P01:USCS303 – Operating System (OS) Practical_04	3
System (OS) Practical – 04	3
Practical-04: Process Communication.	3
Practical Date: 6th August 2021	3
Practical Aim: Producer-Consumer Problem, RPI	3
Process Communication	4
Producer-Consumer Problem	5
a) Using Shared Memory	6
1. Producer-Consumer Solution using Shared Memory	7
A) QUESTION: - Write a java program for producer consumer problem using shared memory.	7
B) SOURCE CODE	7
File Name: P4_PC_SM_Buffer_BL.java	7
File Name: P4_PC_SM_BufferImpl_BL.java	8
File Name: P4_PC_SM_BL.java	10
C) OUTPUT: -	11
Using Message Passing	12
2. Producer-Consumer Solution using Message Passing	13
A) QUESTION: - Write a java program for producer consumer problem using message passing.	13
B) SOURCE CODE: -	13
File Name: P4_PC_MP_Channel_BL.java	13
File Name: P4_PC_MP_MessageQueue_BL.java	14
File Name: P4_PC_MP_BL.java	16
C) OUTPUT: -	18
Remote Procedure Calls	19
3. Remote Method Invocation (RMI) Calculator.	20
A) QUESTION: - Write a java RMI program for adding, subtracting, multiply and dividing two numbers.	20
B) Steps: -	20
C) SOURCE CODE	22
File Name: P4_RMI_CalcServerIntf_BL.java	22
File Name: P4_RMI_CalcServerImpl_BL.java	23
File Name: P4_RMI_CalcServer_BL.java	24

Smt. Chandibai Himathmal Mansukhani College

File Name: P4_RMI_CalcClient_BL.java 26

D) OUTPUT: - 28

CS Dept / Sem III / 2021-2022

USCS3P01:USCS303 – Operating System (OS) Practical_04 **System (OS) Practical – 04**

Practical-04: Process Communication.

Practical Date: 6th August 2021

Practical Aim: Producer-Consumer Problem, RPI

Content:

Solution for Producer - Consumer Problem using shared memory and message passing.

Communication in Client-Server environment using Remote Method Invocation (RMI).

Process:

Producer - Consumer problem without threads and without synchronization.

Implement Remote method invocation (RMI).

Prior Knowledge:

Concept of shared memory, message passing, interfaces and remote method invocation.

Process Communication

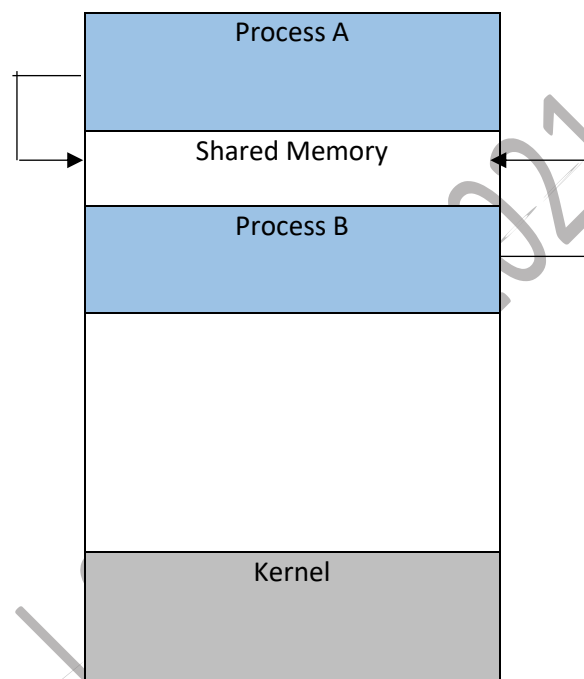
- Processes often need to communicate with each other.
- This is complicated in distributed systems by the fact that the communicating processes may be on different workstations.
- Inter – process communication provides a means for processes to cooperate and compete.
- Message passing and remote procedure calls are the most common methods of inter-process communication in distributed systems.
- A less frequently used but no less valuable method is distributed shared memory.

Producer-Consumer Problem

- In a producer/consumer relationship, the producer portion of an application generates data and stores it in a shared object, and the consumer portion of an application reads data from the shared object.
- One example of a common producer/consumer relationship is print spooling. A word processor spools data to a buffer (typically a file) and that data is subsequently consumed by the printer as it prints the document. Similarly, an application that copies data onto compact discs places data in a fixed-size buffer that is emptied as the CD-RW drive burns the data onto the compact disc.

a) Using Shared Memory

- Shared memory is memory that may be simultaneously accessed by multiple processes with an intent to provide communication among them or avoid redundant copies.
- Shared memory is an efficient means of passing data between processes.



Smt. Chandibai Himathmal Mansukhani College

1. Producer-Consumer Solution using Shared Memory

A) QUESTION: - Write a java program for producer consumer problem using shared memory.

B) SOURCE CODE

File Name: P4_PC_SM_Buffer_AN.java

//Name: ABHISHEK NIKAM

// Batch: B2

// PRN: 2020016400805951

// Date: 6th August,2021

// Prac-04: Process Communication

public interface P4_PC_SM_Buffer_AN

{

//Producers call this method

public void insert(String item);

//Consumers call this method

public String remove();

}//interface ends

Smt. Chandibai Himathmal Mansukhani College

File Name: P4_PC_SM_BufferImpl_AN.java

//Name: ABHISHEK NIKAM

// Batch: B2

// PRN: 2020016400805951

// Date: 6th August,2021

// Prac-04: Process Communication

```
public class P4_PC_SM_BufferImpl_AN implements  
P4_PC_SM_Buffer_AN  
{  
  
    private static final int BUFFER_SIZE = 5;  
    private String[] elements;  
    private int in, out, count;  
  
    public P4_PC_SM_BufferImpl_AN() //constructor initializing the variable to  
initial value  
    {  
  
        count = 0;  
        in = 0;  
        out = 0;  
        elements = new String[BUFFER_SIZE];  
  
        }//constructor ends  
  
        // Producers call this method  
        public void insert(String item)  
        {  
  
            while(count == BUFFER_SIZE)  
  
                ;//do nothing as there is no free space  
  
            //add an item to the buffer  
  
            elements[in] = item;
```


Smt. Chandibai Himathmal Mansukhani College

```
        in = (in + 1) % BUFFER_SIZE;

        ++count;

        System.out.println("Item Produced: " + item + "at position" + in +
"having total items " + count);

    } //insert() ends

//Consumers call this method

public String remove()
{
    String item;
    while(count == 0)
        ; //do nothing as there is nothing to consume

    //remove an item from the buffer
    item = elements[out];
    out = (out + 1) % BUFFER_SIZE;
    --count;

    System.out.println("Item Consumed: " + item + " from position " + out +
"remaining total items" + count);

    return item;
} //remove() ends
} //class ends
```

Smt. Chandibai Himathmal Mansukhani College

File Name: P4_PC_SM_BL.java

//Name: ABHISHEK NIKAM

// Batch: B2

// PRN: 2020016400805951

// Date: 6th August,2021

// Prac-04: Process Communication

public class P4_PC_SM_AN

{

public static void main(String[] args) {

P4_PC_SM_BufferImpl_AN bufobj = new
P4_PC_SM_BufferImpl_AN();

System.out.println("\n=====PRODUCER producing
ITEMS=====\\n");

bufobj.insert("Name: ABHISHEK NIKAM");

bufobj.insert("CHMS: Batch-B2");

bufobj.insert("PRN: 2020016400805951");

bufobj.insert("USCSP301 - USCS303: OS Practical -P4");

System.out.println("\n=====CONSUMER consuming the
ITEMS=====\\n");

String element = bufobj.remove();

System.out.println(element);

System.out.println(bufobj.remove());

System.out.println(bufobj.remove());

System.out.println(bufobj.remove());

} //main ends

} //class ends

C) OUTPUT: -

```
Item Consumed: USCSP301 - USCS303: OS Practical -P4 from position 4remaining total items0
USCSP301 - USCS303: OS Practical -P4
C:\SYCS\USCSP301_USCS303_OS_B2\Prac_04_Bimal_b1_6th july 2021\Q1_PC_SM_B1>
```

Using Message Passing

- Message passing is the basis of most inter-process communication in distributed systems.
- It is at the lowest level of abstraction and requires the application programmer to be able to identify the destination process, the message, the source process and the data types expected from these processes.
- Communication in the message passing paradigm, in its simplest form, is performed using the send() and receive() primitives. The syntax is generally of the form:

Send (receiver, message)

Receive (sender, message)

- The send() primitive requires the name of the destination process and the message data as parameters. The addition of the name of the sender as a parameter for the send() primitive would enable the receiver to acknowledge the message. The receive() primitive requires the name of the anticipated sender and should provide a storage buffer for the message.

2. Producer-Consumer Solution using Message Passing

A) QUESTION: - Write a java program for producer consumer problem using message passing.

B) SOURCE CODE: -

File Name: P4_PC_MP_Channel_AN.java

//Name: ABHISHEK NIKAM

// Batch: B2

// PRN: 2020016400805951

// Date: 6th August,2021

// Prac-04: Process Communication

```
public interface P4_PC_AN_Channel_BL<E>
{
    // Send a message to the channel
    public void send(E item);
    // Receive a message from the channel
    public E receive();
}//interface ends
```

Smt. Chandibai Himathmal Mansukhani College

File Name: P4_PC_MP_MessageQueue_AN.java

//Name: ABHISHEK NIKAM

// Batch: B2

// PRN: 2020016400805951

// Date: 6th August,2021

// Prac-04: Process Communication

import java.util.Vector;

**public class P4_PC_MP_MessageQueue_AN<E> implements
P4_PC_MP_Channel_AN<E>**

{

private Vector<E> queue;

public P4_PC_MP_MessageQueue_AN(){

queue = new Vector<E>();

}

// This implements a nonblocking send

public void send(E item)

{

queue.addElement(item);

} // send() ends

// This implements a nonblocking receive

public E receive()

{

if(queue.size() == 0)

return null;

else

return queue.remove(0);

} // receive() ends

Smt. Chandibai Himathmal Mansukhani College

} // class ends

CS Dept / Sem III / 2021-2022

Smt. Chandibai Himathmal Mansukhani College

File Name: P4_PC_MP_AN.java

//Name: ABHISHEK NIKAM

// Batch: B2

// PRN: 2020016400805951

// Date: 6th August,2021

// Prac-04: Process Communication

import java.util.Date;

public class P4_PC_MP_AN

{

public static void main(String args[])

{

// Producer and Consumer process

**P4_PC_MP_Channel_AN<Date> mailBox = new
P4_PC_MP_MessageQueue_AN<Date>();**

int i=0;

do

{

Date message = new Date();

System.out.println("Producer produced - " + (i+1) + ":" + message);

mailBox.send(message);

Date rightNow = mailBox.receive();

if(rightNow != null)

{

System.out.println("Consumer consumed " + (i+1)+": " +rightNow);

}

i++;

}while (i < 10);

} // main ends

Smt. Chandibai Himathmal Mansukhani College

}// class ends

CS Dept / Sem III / 2021-2022

C) OUTPUT: -

```
Producer produced - 1: Mon Aug 09 16:04:18 IST 2021
Consumer consumed 1: Mon Aug 09 16:04:18 IST 2021
Producer produced - 2: Mon Aug 09 16:04:18 IST 2021
Consumer consumed 2: Mon Aug 09 16:04:18 IST 2021
Producer produced - 3: Mon Aug 09 16:04:18 IST 2021
Consumer consumed 3: Mon Aug 09 16:04:18 IST 2021
Producer produced - 4: Mon Aug 09 16:04:18 IST 2021
Consumer consumed 4: Mon Aug 09 16:04:18 IST 2021
Producer produced - 5: Mon Aug 09 16:04:18 IST 2021
Consumer consumed 5: Mon Aug 09 16:04:18 IST 2021
Producer produced - 6: Mon Aug 09 16:04:18 IST 2021
Consumer consumed 6: Mon Aug 09 16:04:18 IST 2021
Producer produced - 7: Mon Aug 09 16:04:18 IST 2021
Consumer consumed 7: Mon Aug 09 16:04:18 IST 2021
Producer produced - 8: Mon Aug 09 16:04:18 IST 2021
Consumer consumed 8: Mon Aug 09 16:04:18 IST 2021
Producer produced - 9: Mon Aug 09 16:04:18 IST 2021
Consumer consumed 9: Mon Aug 09 16:04:18 IST 2021
Producer produced - 10: Mon Aug 09 16:04:18 IST 2021
Consumer consumed 10: Mon Aug 09 16:04:18 IST 2021
```

Remote Procedure Calls.

- Message passing leaves the programmers with the burden of the explicit control of the movement of data. Remote procedure calls (RPC) relieves this burden by increasing the level of abstraction and providing semantics similar to local procedure call.

- Syntax:

- The syntax of a remote procedure call is generally of the form:

Call procedure_name(value_arguments;result_arguments)

- The client process blocks at the call() until the reply is received
- The remote procedure is the server processes which has already begun executing on a remote machine.
- It blocks at the receive() until it receives a message and parameters from the sender.
- The server then sends a reply() when it has finished its task.
- The syntax is as follows:

receive procedure_name(in value_parameters; out result_parameters)

reply(caller, result_parameters)

- In the simplest case, the execution of call() generates a client stub which marshals the arguments into a message and sends the message to the server machine. On the server machine the server is blocked awaiting the message. On receipt of the message the server stub is generated and extracts the parameters from the message and passes the parameters and control to the procedure. The results are returned to client with the same procedure in reverse.

3. Remote Method Invocation (RMI) Calculator.

A) QUESTION: - Write a java RMI program for adding, subtracting, multiply and dividing two numbers.

B) Steps: -

✓ **Step 1: Creating the Remote Interface**

- This file defines the remote interface that is provided by the server. It contains four methods that accept two **integer** arguments and return their sum, difference, product and quotient. All remote interfaces must extend the **Remote** interface, which is part of **java.rmi**. **Remote** defines no members. Its purpose is simply to indicate that an interface uses remote methods. All remote methods can throw a **RemoteException**.

✓ **Step 2: Implementing the Remote Interface**

- This file implements the remote interface. The implementation of all the four methods is straight forward. All remote methods must extend **UnicastRemoteObject**, which provides functionality that is needed to make objects available from remote machines.

✓ **Step 3: Creating the server**

- This file contains the main program for the server machine. Its primary function is to update the RMI registry on that machine. This is done by using the **rebind()** method of the **Naming** class (found in **java.rmi**), that method associates a name with an object reference. The first argument to the **rebind()** method is a string that names the server. Its second argument is a reference to an instance of **CalcServerImpl**.

✓ **Step 4: Creating the Client**

- This file implements the client side of this distributed application. It accepts three command-line arguments. The first is the IP address or name of the server machine. The second and third arguments are the two numbers that are to be operated.
- The application begins by forming a string that follows the URL syntax. This URL uses the rmi protocol. The string includes the IP address or name of the server and the string "CSB2". The program then invokes the **lookup()** method of the **Naming** class. This method accepts one argument, the rmi URL, and returns a reference to an object of type **CalcServerInf**. All remote method invocations can then be directed to this object.

✓ **Step 5: Manually generate a stub, if required**

- Prior to Java 5, stubs needed to be built manually by using **rmic**. This step is not required for modern versions of Java. However, if we work

in a legacy environment, then we can use the `rmic` compiler, as shown here, to build a stub.

`rmic CalcServerImpl`

✓ **Step 6: Install Files on the Client and Server Machines**

- Copy **`P4_RMI_CalcClient_AN.class`**, **`P4_RMI_CalcServerImpl_AN_Stub.class`** (if needed), and **`P4_RMI_CalcServerIntf_AN.class`** to a directory on the client machine.
- Copy **`CalcServerIntf.class`**, **`P4_RMI_CalcServerImpl_AN.class`**, **`P4_RMI_CalcServerImpl_AN_Stub.class`** (if needed), and **`P4_RMI_CalcServer_AN.class`** to a directory on the server machine.

✓ **Step 7: Start the RMI Registry on the Server Machine**

- The JDK provides a program called `rmiregistry`, which executes on the server machine. It maps names to object references. Start the RMI Registry from the command line, as shown here: **`start rmiregistry`**
- When this command returns, a new window gets created. Leave this window open until we are done experimenting with the RMI example.

✓ **Step 8: Start the server**

- The server code is started from the command line, as shown here:

`java P4_RMI_CalcServer_AN`

✓ **Step 9: Start the client**

- The client code is started from the command line, as shown here:

`java P4_RMI_CalcClient_BL 127.0.0.1 15 5`

❖ **For Execution:**

➤ **Open 2 Command Terminals:**

✦ **Terminal-01:**

- Compile all the .java files
- Command:
`javac *.java`
- Start the RMI registry:
- Command:

`start rmiregistry`

✦ **Terminal-02:**

- **Start the Server:**
- **Command:**
`java P4_RMI_CalcServer_AN`
- **Run the Client:**
- **Command:**
`java P4_RMI_CalcClient_AN`
`127.0.0.1 15 5`

Smt. Chandibai Himathmal Mansukhani College

C) SOURCE CODE

File Name: P4_RMI_CalcServerIntf_AN.java

//Name: ABHISHEK NIKAM

// Batch: B2

// PRN: 2020016400805951

// Date: 6th August,2021

// Prac-04: Process Communication

import java.rmi.*;

public interface P4_RMI_CalcServerIntf_AN extends Remote

{

int add(int a,int b)throws RemoteException ;

int subtract(int a,int b)throws RemoteException ;

int multiply(int a,int b)throws RemoteException ;

int divide(int a,int b)throws RemoteException ;

} // interface ends

Smt. Chandibai Himathmal Mansukhani College

File Name: P4_RMI_CalcServerImpl_BL.java

//Name: ABHISHEK NIKAM

// Batch: B2

// PRN: 2020016400805951

// Date: 6th August,2021

// Prac-04: Process Communication

import java.rmi.*;

import java.rmi.server.*;

public class P4_RMI_CalcServerImpl_AN extends UnicastRemoteObject implements P4_RMI_CalcServerIntf_AN

{

public P4_RMI_CalcServerImpl_AN() throws RemoteException{

}

public int add(int a, int b) throws RemoteException

{

return a + b;

}

public int subtract(int a, int b) throws RemoteException

{

return a - b;

}

public int multiply(int a, int b) throws RemoteException

{

return a *b;

}

public int divide(int a, int b) throws RemoteException

{

return a / b;

Smt. Chandibai Himathmal Mansukhani College

```
    }  
} // class ends
```

File Name: P4_RMI_CalcServer_AN.java

//Name: ABHISHEK NIKAM

// Batch: B2

// PRN: 2020016400805951

// Date: 6th August,2021

// Prac-04: Process Communication

```
import java.net.*;  
import java.rmi.*;  
public class P4_RMI_CalcServer_AN  
{  
    public static void main(String args[])  
    {  
        try  
        {  
            P4_RMI_CalcServerImpl_AN csi = new P4_RMI_CalcServerImpl_AN();  
            Naming.rebind("CSB2", csi);  
        } // try ends  
        catch(Exception e)  
        {  
            System.out.println("Exception: " + e);  
        } // catch ends  
    } // main ends  
} // class ends
```


CS Dept / Sem III / 2021-2022

Smt. Chandibai Himathmal Mansukhani College

File Name: P4_RMI_CalcClient_AN.java

//Name: ABHISHEK NIKAM

// Batch: B2

// PRN: 2020016400805951

// Date: 6th August,2021

// Prac-04: Process Communication

```
import java.rmi.*;

public class P4_RMI_CalcClient_AN
{
    public static void main(String args[])
    {
        try{
            String CSURL="rmi://" + args[0] + "/" + "CSB2";
            P4_RMI_CalcServerIntf_AN CSIntf =
(P4_RMI_CalcServerIntf_BL) Naming.lookup(CSURL);

            System.out.println("The first number is: " + args[1]);
            int x = Integer.parseInt(args[1]);
            System.out.println("The second number is: " + args[2]);
            int y = Integer.parseInt(args[2]);

            System.out.println("=====Arithmetic Operations=====");
            System.out.println("Addition: " + x + " + " + y + " = " + CSIntf.add(x,y));
            System.out.println("Subtraction: " + x + " - " + y + " = " + CSIntf.subtract(x,y));
            System.out.println("Multiplication: " + x + " * " + y + " = "
"+CSIntf.multiply(x,y));
            System.out.println("Division: " + x + " / " + y + " = "
+CSIntf.divide(x,y));
        } // try ends
        catch(Exception e){
            System.out.println("Exception: " + e);
        }
    }
}
```

Smt. Chandibai Himathmal Mansukhani College

}// catch ends

} // main ends

} // class ends

CS Dept / Sem III / 2021-2022

D) OUTPUT: -

```
08/09/2021 04:17 PM <DIR> -
08/09/2021 04:17 PM <DIR> ..
08/09/2021 04:17 PM      1,813 P4_RMI_CalcClient_BL.class
08/09/2021 04:16 PM      1,034 P4_RMI_CalcClient_BL.java
08/09/2021 04:17 PM      583 P4_RMI_CalcServerImpl_BL.class
08/09/2021 04:13 PM      665 P4_RMI_CalcServerImpl_BL.java
08/09/2021 04:17 PM      312 P4_RMI_CalcServerIntf_BL.class
08/09/2021 04:10 PM      426 P4_RMI_CalcServerIntf_BL.java
08/09/2021 04:17 PM      1,115 P4_RMI_CalcServer_BL.class
08/09/2021 04:17 PM      479 P4_RMI_CalcServer_BL.java
                8 File(s)      6,427 bytes
                2 Dir(s)  188,007,886,144 bytes free
```

```
The first number is: 15
The second number is: 5
=====Arithmetic Operations=====
Addition: 15+5=20
Subtraction: 15 - 5 = 10
Multiplication: 15 * 5 = 75
Division: 15/5 = 3
```