Distributed Computing

Experiment – 1

Program to demonstrate Inter- Process communication using Java	
Learning Objective:	To understand basic underlying concepts of forming distributed systems.
Learning Outcome:	students will be able to demonstrate Inter-Process communication.
Course Outcome:	**CSL801.1**
Program (Outcome:	(PO1) Engineering knowledge : Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems. (PO2) Problem Analysis : Identify, Formulate, review research literature and analyze complex engineering problems reaching substantiated conclusion using first principles of mathematics, natural science and engineering sciences.
Bloom's Taxonomy Level:	Analysis, Create
Theory:	**IPC** IPC, or Inter-Process Communication, refers to the mechanisms and techniques used by various processes or programs to communicate with each other in a computing environment. IPC is essential for coordinating and exchanging information between different processes running concurrently on a computer system. Here are some key points about IPC, along with its advantages and disadvantages: **Advantages of IPC** • **Data Sharing**: IPC allows processes to share data, enabling collaboration and information exchange between different parts of a system. • **Resource Sharing**: Processes can share resources such as files, devices, and memory, leading to more efficient utilization of system resources.

- **Concurrency**: IPC enables concurrent execution of processes, allowing multiple tasks to be performed simultaneously and improving system efficiency.

- **Modularity**: Processes can be designed as modular components, with IPC facilitating communication between these modules. This promotes code organization and maintainability.

- **Fault Isolation**: IPC can help isolate faults or errors in one process from affecting others, contributing to system stability and reliability.

Disadvantages of IPC

- **Complexity**: Implementing IPC mechanisms can add complexity to the design and development of a system. This complexity can lead to more challenging debugging and maintenance.

- **Performance Overhead**: IPC may introduce performance overhead due to the need for data copying, synchronization, and communication protocols. This overhead can impact the overall system performance.

- **Security Concerns**: In some cases, IPC may introduce security vulnerabilities if not implemented carefully. Unauthorized access to shared resources or sensitive information can be a concern.

- **Synchronization Issues**: Coordinating communication between processes may require synchronization mechanisms to prevent race conditions and ensure data consistency. Improper synchronization can lead to issues like deadlocks or data corruption.

Types of IPC

1. **Message Passing**:

Advantages: Explicit communication, easier to implement and understand, supports different data types.
Disadvantages: Overhead due to message creation and copying, potential for message delays.

2. **Shared Memory**:

Advantages: Faster communication as data is directly shared, efficient for large data sets.

Disadvantages: Requires synchronization mechanisms, potential for data inconsistency, more complex to implement.

3. **Pipes and FIFOs (First In, First Out)**:

Advantages: Simple and efficient for communication between related processes.
Disadvantages: Limited to communication between related processes, unidirectional communication.

4. **Sockets**:

Advantages: Network-capable communication, suitable for inter-process communication on different machines.
Disadvantages: Overhead due to network communication, more complex setup.

5. **Signals**:

Advantages: Simple and lightweight for certain types of communication.
Disadvantages: Limited in terms of data size, less suitable for complex communication.

6. **Remote Procedure Calls (RPC)**:

Advantages: Abstraction of remote communication, similar to local function calls.
Disadvantages: Overhead due to marshaling and network communication, potential for network failures.

Outcome :

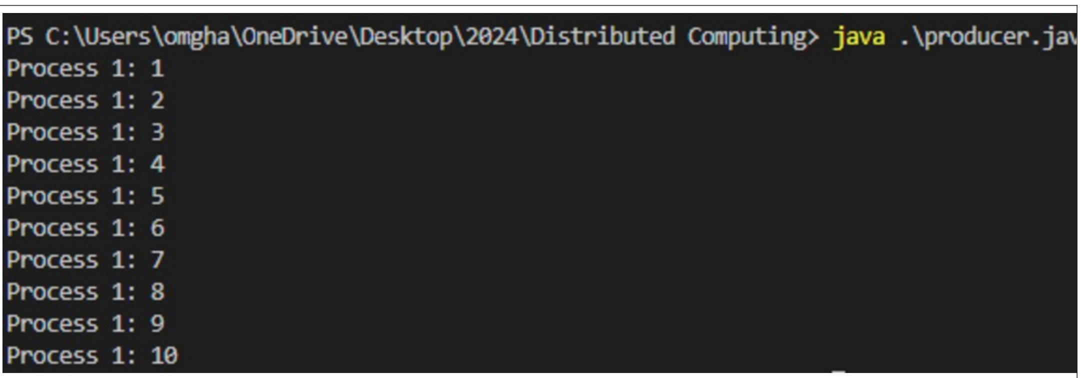
Producer Consumer Problem -

Producer -

Program -

```
import java.io.IOException;

import java.io.RandomAccessFile;

import java.nio.MappedByteBuffer;

import java.nio.channels.FileChannel;



/**

 * Interprocess communication in Java using a memory-mapped file
 */
public class producer {
   public static void main(String args[]) throws IOException, InterruptedException {
       RandomAccessFile rd = new
RandomAccessFile("C:/Users/omgha/OneDrive/Desktop/2024/Distributed
Computing/mapped.txt", "rw");
       FileChannel fc = rd.getChannel();
       MappedByteBuffer mem = fc.map(FileChannel.MapMode.READ_WRITE, 0, 1000);

       try {
          Thread.sleep(10000);
       } catch (InterruptedException e) {
          e.printStackTrace();
       }

       for (int i = 1; i <= 10; i++) {
          mem.put((byte) i);
          System.out.println("Process 1: " + (byte) i);
          Thread.sleep(1); // time to allow CPU cache refreshed
       }

       // Close resources
       fc.close();
       rd.close();
   }
}
```

Output -



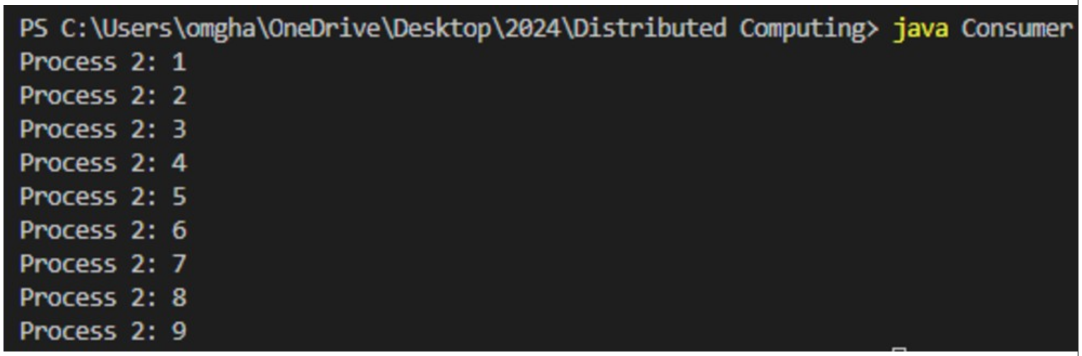
```
PS C:\Users\omgha\OneDrive\Desktop\2024\Distributed Computing> java .\producer.jav
Process 1: 1
Process 1: 2
Process 1: 3
Process 1: 4
Process 1: 5
Process 1: 6
Process 1: 7
Process 1: 8
Process 1: 9
Process 1: 10
```

Consumer -

Program -

```
import java.io.IOException;

import java.io.RandomAccessFile;

import java.nio.MappedByteBuffer;

import java.nio.channels.FileChannel;


/**

 * Consumer process reading data from the memory-mapped file

 */

public class Consumer {

   public static void main(String args[]) throws IOException, InterruptedException {

       RandomAccessFile rd = new
RandomAccessFile("C:/Users/omgha/OneDrive/Desktop/2024/Distributed
Computing/mapped.txt", "r");

       FileChannel fc = rd.getChannel();

       MappedByteBuffer mem = fc.map(FileChannel.MapMode.READ_ONLY, 0, 1000);


       // Assuming that the producer has already written the data

       for (int i = 0; i < 9; i++) {
            byte value = mem.get();

            System.out.println("Process 2: " + value);

        }


        // Close resources

        fc.close();

        rd.close();

    }

}
```

Output -



```
PS C:\Users\omgha\OneDrive\Desktop\2024\Distributed Computing> java Consumer
Process 2: 1
Process 2: 2
Process 2: 3
Process 2: 4
Process 2: 5
Process 2: 6
Process 2: 7
Process 2: 8
Process 2: 9
```

Conclusion :	Thus we have studied the basics of creating distributed systems, learning how to build strong and efficient networks. Understanding these principles helps us tackle challenges in distributed computing, setting the groundwork for creative system designs.
References:	https://www.geeksforgeeks.org/interprocess-communication-in-distributed-systems/ https://en.wikipedia.org/wiki/Inter-process_communication https://www.baeldung.com/cs/inter-process-communication

Rubrics for Assessment

Timely Submission	Submitted after 2 weeks 0	Submitted after deadline 1	On time Submission 2
Understanding	Student is confused about the concept 0	Students has justifiably understood the concept 2	Students is very clear about the concepts 3
Performance	Students has not performed the Experiment 0	Student has performed with help 2	Student has independently performed the experiment 3
Development	Student struggles to write code 0	Student can write code the requirement stated 1	Student can write exceptional code with his own ideas 2