Experiment 03 - Interprocess Communication

Learning Objective: Student should be able to write program to implement interprocess communication

Tools: Python

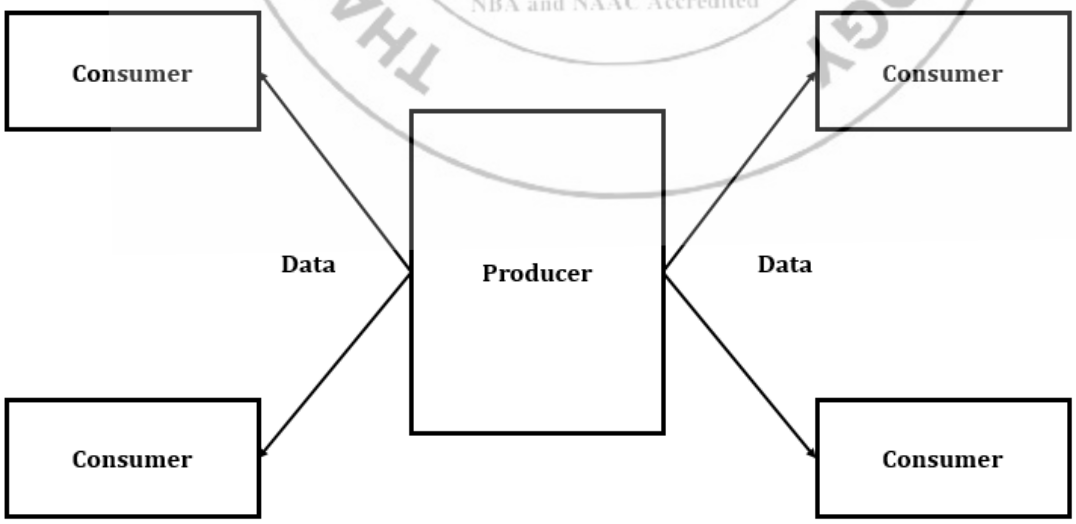
Theory:

Inter Process Communication (IPC)

Inter process communication (IPC) allows different programs or processes running on a computer to share information with each other. IPC allows processes to communicate by using different techniques like sharing memory, sending messages, or using files. It ensures that processes can work together without interfering with each other. Cooperating processes require an Inter Process Communication (IPC) mechanism that will allow them to exchange data and information.

Working of IPC

1. Inter-Process Communication (IPC) is the activity of sharing data across multiple and commonly specialized processes using communication protocols.
2. Typically, applications using IPC are categorized as clients and servers.
3. Client requests data and the Server responds to client requests.
4. IPC has ability to communicate between two cooperating process.
5. IPC is used in many contexts, such as a producer – consumer problem.



Methods of IPC

I) Pipes:

- It is simplest of all the IPCs.
- A pipe enables one way communication between a processes.
- When process terminates, pipe is removed automatically.

II) FIFO:

- FIFO Stands for First In First Out.
- It is also called as Named Pipes.
- A FIFO enables unidirectional communication between a processes.

III) Message Queue:

- It is used in implementation of Message Passing IPC.
- Message Queue will remain, even if the process have existed.
- It can be unidirectional or bidirectional.

IV) Shared Memory:

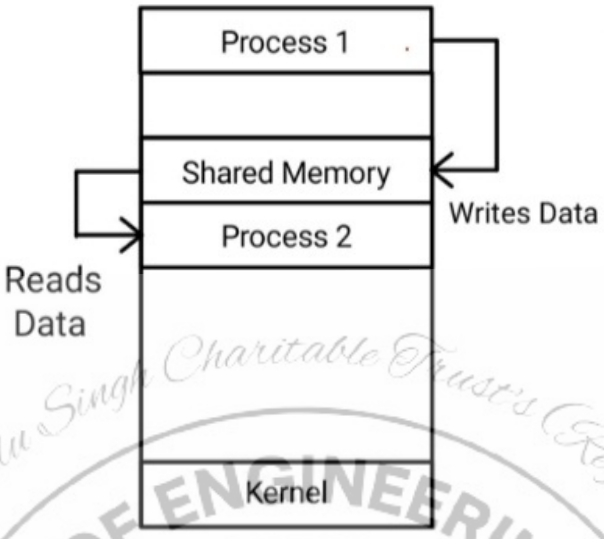
- It includes a memory segment which is shared by two or more processes.
- When a process creates a memory segment for IPC from its logical address space.
- Other process can attach that available space for communication.
- Shared Memory is one of the most simplest & Logical Implementation of IPC.

Types of IPC

1. Shared Memory

In this method, the multiple processes can communicate with each other simultaneously by accessing or sharing the memory. The multiple processes can exchange the data using read/write to the shared memory because the OS creates a common memory in RAM for sharing. It requires protection by the synchronization of access of all multiple processes in the communication.

It is a more efficient method and shares the data very quickly. To understand this, consider the 2 processes as process 1 and process 2. If one process creates memory, then the other process will access it by means of sharing. The communication between the processes in IPC using the shared memory method is shown in the figure below.



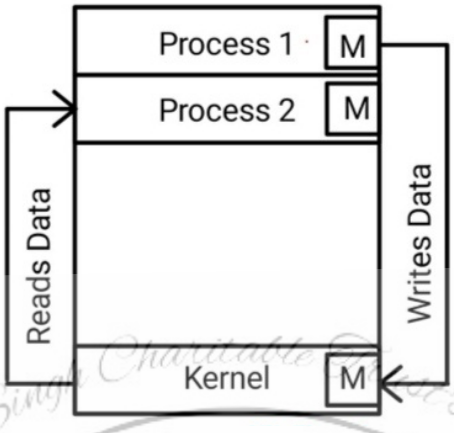
Shared Memory Type IPC

The information generated by process 1 about a particular resource and stored the record in shared memory. If process 2 wants to use this information, it checks the stored record in the shared memory, takes the information generated by process 1, and processes it accordingly. Thus, processes 1 and 2 can utilize shared memory to retrieve information such as records from other processes and transfer certain information to other processes. Examples are Posix and Windows operating systems.

2. Message Passing

It is the other method of inter-process communication for synchronization and communication between the processes. It is very slow and easy to implement using system calls compared to the shared memory method. The processes can exchange the data with each other without using any shared resources or variables.

In message passing, the communication between the processes is performed through the communication link. The operations performed by the processes are, sending the message and receiving the message. Since the message might be variable or fixed type.



Message Passing Inter Process Communication

From the above figure, process 1 sends (writes) the message (M) to the kernel. While process 2 receives (reads) the message from the kernel sent by process 1.

Suppose, process 1 and process 2 communicated together by establishing the communication link as shown in the figure above. After that, they exchange the message through send and receive operations. The message is transferred in the First In First Out (FIFO) style. The standard message exchanged by the processes contains 2 parts. Such as

Header: It contains the type of message, source and destination identities, length of the message, and control information like priority, sequence number, and low disk space actions.
Body: It contains the original message that is to be exchanged.

The communication link can be established in different types. They are,

- Direct communication.
- Indirect communication.
- Synchronous message passing.
- Asynchronous message passing.
- Buffering.

Advantages of IPC

- Interprocess communication allows one application to manage another and enables glitch-free data sharing.
- Interprocess communication helps send messages efficiently between processes.
- The program is easy to maintain and debug because it is divided into different sections of code that work separately.

- Programmers can perform a variety of other tasks at the same time, including Editing, listening to music, compiling, etc.
- Data can be shared between different programs at the same time.
- Tasks can be subdivided and run on special types of processors. You can then exchange data via IPC.

Disadvantages of IPC

- The program cannot write to similar locations.
- Processes or programs that use the shared memory model must make sure that they are not writing to similar memory locations.
- The shared storage model can cause problems such as storage synchronization and protection that need to be addressed.
- It's slower than a direct function call.

Code:

```
#!/usr/bin/env python3

import logging
import os
import time
from multiprocessing import Process, Array

def process1(shared):
    process1_logger = logging.getLogger('process1')

    process1_logger.info(f"Pid:{os.getpid()}")

    for i in range(1,11):
        while True:
            try:
                process1_logger.info(f"Writing {int(i)}")
                shared[i-1] = i
                if i % 6 == 0:
                    process1_logger.info("Intentionally sleeping for 5 seconds")
                    time.sleep(5)
                break
            except Exception as e:
                print(str(e))
                pass

    process1_logger.info("Finished process 1")

def process2(shared):
    process2_logger = logging.getLogger('process2')

    process2_logger.info(f"Pid:{os.getpid()}")

    for i in range(10):
        while True:
            try:
```

```
            line = shared[i]
            if line == -1:
                process2_logger.info("Data not available sleeping for 1 second before retrying")
                time.sleep(1)
                raise Exception('pending')
                process2_logger.info(f"Read: {int(line)}")
            break
        except Exception:
            pass

    process2_logger.info("Finished process 2")

def main():
                parent_logger = logging.getLogger('parent')
    parent_logger.info(f"Pid:{os.getpid()}")

    arr = Array('i', [-1] * 10)

            procs = [Process(target=process1, args=(arr,)), Process(target=process2, args=(arr,))]

    for proc in procs:
        proc.start()

    for proc in procs:
        proc.join()

logging.basicConfig(level=logging.INFO)

if __name__ == '__main__':
    main()
```

Output:

```
INFO:parent:Pid:16740
INFO:process1:Pid:2796
INFO:process1:Writing 1
INFO:process1:Writing 2
INFO:process1:Writing 3
INFO:process1:Writing 4
INFO:process1:Writing 5
INFO:process1:Writing 6
INFO:process1:Intentionally sleeping for 5 seconds
INFO:process2:Pid:12832
INFO:process2:Read: 1
INFO:process2:Read: 2
INFO:process2:Read: 3
INFO:process2:Read: 4
INFO:process2:Read: 5
INFO:process2:Read: 6
INFO:process2:Data not available sleeping for 1 second before retrying
INFO:process2:Data not available sleeping for 1 second before retrying
INFO:process2:Data not available sleeping for 1 second before retrying
INFO:process2:Data not available sleeping for 1 second before retrying
INFO:process2:Data not available sleeping for 1 second before retrying
INFO:process1:Writing 7
INFO:process1:Writing 8
INFO:process1:Writing 9
INFO:process1:Writing 10
INFO:process1:Finished process 1
INFO:process2:Read: 7
INFO:process2:Read: 8
INFO:process2:Read: 9
INFO:process2:Read: 10
INFO:process2:Finished process 2
```

Learning Outcomes: The student should have the ability to
LO1: Describe the protocol for Inter process communication.
LO 2: justify the client server are managed properly by the inter process communication

Course Outcomes: Upon completion of the course students will be able to understand interprocess communication.

Conclusion:

For Faculty Use

Correction Parameters	Formative Assessment [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	Total
Marks Obtained				