Vidya Vikas Education Trust's
Universal College of Engineering, Kaman Road, Vasai – 401208
Accredited A Grade by NAAC

Experiment 1

Name of the Student: - Rutuja Rajesh Kini
Roll No:-53
Date of Practical Performed: -8/1/25 Staff Signature with Date & Marks

Aim: Write a program to perform Inter-process communication

Theory:

Inter-process Communication (IPC) refers to the mechanisms that allow processes (programs in execution) to communicate with each other and synchronize their actions. In a distributed system or multiprocessing environment, different processes often need to exchange data or signals to coordinate their tasks or share resources. IPC is crucial for ensuring that processes can work together efficiently and safely.

Types of Inter-Process Communication (IPC):

1. **Message Passing:** Processes communicate by sending and receiving messages. The message can be passed through direct communication (point-to-point) or indirect communication (via a message queue).
2. **Shared Memory:** Multiple processes access the same region of memory for communication. One process writes to the memory, and other processes read from it.
3. **Message-Oriented Middleware (MOM):** Middleware that allows processes to send messages asynchronously through a messaging system. MOM provides reliable, scalable, and decoupled communication.
4. **Remote Procedure Call (RPC):** A protocol that allows a program to execute code on a remote machine as if it were a local function call. This is often used for distributed systems where processes on different machines need to interact.
5. **Distributed Shared Memory (DSM):** A system where multiple machines or processes share the same memory space, making it appear as though they have a global memory. It allows processes to communicate as if they were sharing memory.

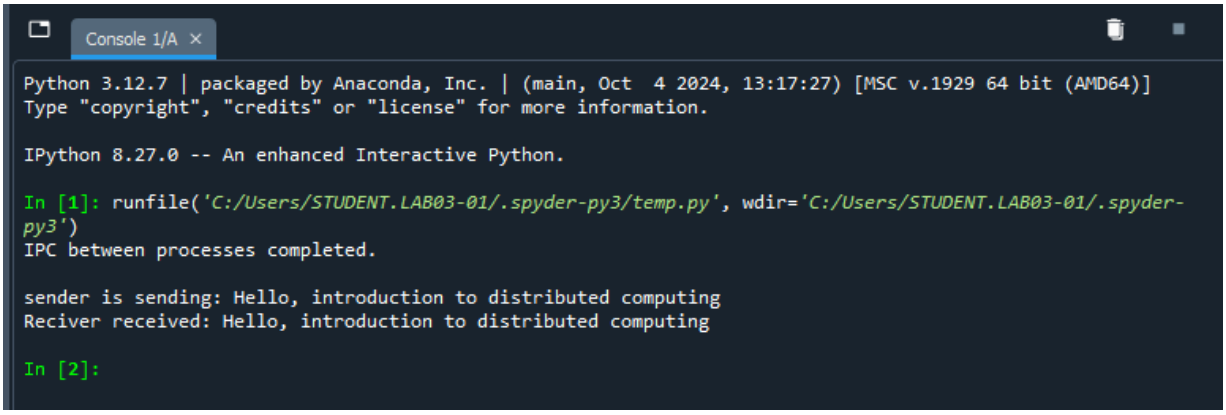
Example Use Cases for IPC:

- **Client-Server Applications**: Clients may communicate with servers using sockets, RPCs, or message queues.
- **Multi-threaded Programs**: Different threads within a process might use shared memory or semaphores to communicate.
- **Distributed Systems**: Different nodes in a distributed system might use sockets, message queues, or MOM to exchange information.
- **Real-Time Systems**: Using signals or semaphores for process synchronization in time-sensitive applications.

Code:

```
import multiprocessing
import time

def sender(q):
    time.sleep(2)
    message = "Hello, introduction to distributed computing "
    print(f"sender is sending: {message}")
    q.put(message)

def receiver(q):
    message = q.get()
    print(f"Reciver received: {message}")


def run_ipc():
    q = multiprocessing.Queue()


    sender_process = multiprocessing.Process(target=sender, args=(q,))
    receiver_process = multiprocessing.Process(target=receiver, args=(q,))


    sender_process.start()
    receiver_process.start()


    sender_process.join()
    receiver_process.join()

    print("IPC between processes completed.")

if __name__=="__main__":
    run_ipc()
```

Output:



Conclusion: Hence, we have successfully performed Inter-process communication.