**Ex. No: 11**

**Date:**

**IPC USING SHARED MEMORY**

**Problem Statement:**

To implement inter process communication using shared memory.

**Problem Description:**

To create a client and server based inter process communication program where server writes A-Z to the shared memory. Client should read the shared memory and display it. When user enters exit in client both client and server should terminate.

**Algorithm:**

1. Create a shared memory with size 100 name sm2
2. Create a buffer shm\_server
3. Display the data in shared memory
4. If data in buffer = exit terminate
5. Write A-Z to the buffer
6. Create a shared memory client instance and buffer
7. Read the data from the shared memory
8. Ask user for input
9. Update the data to the memory
10. If data in buffer = exit terminate
11. Close and unlink the shared memory

**Code:**

**Server**

import time

import multiprocessing.shared\_memory as shared\_memory

# Server code

shm\_server = shared\_memory.SharedMemory(create=True, size=100, name='sm2')

buffer = shm\_server.buf

try:

    while True:

        # Display the data from shared memory

        server\_data = bytes(buffer[:100]).decode('utf-8')

        print("Server data in memory:", server\_data)

        # Check if the data in shared memory equals "exit" and terminate if true

        if buffer[:4] == b'exit':

            shm\_server.close()

            shm\_server.unlink()

            break

        message1 = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

        buffer[:100] = b'\x00' \* 100

        message\_bytes1 = message1.encode('utf-8')

        buffer[:len(message\_bytes1)] = message\_bytes1

        time.sleep(5)

finally:

    # Clean up

    shm\_server.close()

    shm\_server.unlink()

**Client:**

import multiprocessing.shared\_memory as shared\_memory

# Client code

shm\_client = shared\_memory.SharedMemory(name="sm2")

buffer = shm\_client.buf

try:

    while True:

        server\_data = bytes(buffer[:100]).decode('utf-8')

        print("Server says:", server\_data)

        if server\_data.strip() == 'exit':

            # Set the "exit" flag in the shared memory to signal the server to terminate

            buffer[:4] = b'exit'

            break

        user\_input = input("Enter 'exit' to quit: ")

        buffer[:100] = b'\x00' \* 100

        message\_bytes = user\_input.encode('utf-8')

        buffer[:len(message\_bytes)] = message\_bytes

        if user\_input == 'exit':

            # Set the "exit" flag in the shared memory to signal the server to terminate

            buffer[:4] = b'exit'

            shm\_client.close()

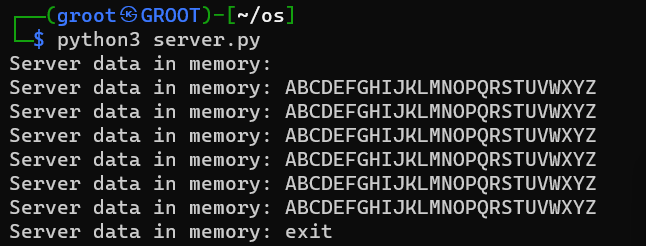
            shm\_client.unlink()

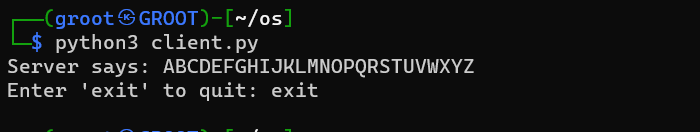
            break

finally:

    shm\_client.close()

**Output:**

****

****

**Result:**

Thus, IPC using shared memory has been implemented successfully.