Assignment 4

Name: Vivek Sapkal   Roll No.: B22AI066

Inter-Process Communication

- Here is a python implementation of the Inter-Process Communication(IPC) using shared memory along with two two processes that communicate with each other using this shared memory. Shared memory is a commonly used mechanism for IPC, where multiple processes can access the same region of memory concurrently.

➢ Code:

```
import multiprocessing
from multiprocessing import shared_memory
import time

# counter which increments every 5 seconds
def process_1(shared_mem):
    try:
        while True:
            time.sleep(5)
            current_value = shared_mem.buf[0]
            current_value += 1
            shared_mem.buf[0] = current_value
            print("Process-1: Counter incremented to", current_value)
    except KeyboardInterrupt:
        print("Process-1 Terminated")

# takes the value from counter and computes the square of that value
def process_2(shared_mem):
    try:
        while True:
            time.sleep(1)  # Check every second
            current_value = shared_mem.buf[0]
            square_value = current_value ** 2
            print("Process-2: Square of counter value", current_value,
"is", square_value)
    except KeyboardInterrupt:
        print("Process-2 Terminated")

if __name__ == "__main__":
    # Create shared memory
    shared_mem =
multiprocessing.shared_memory.SharedMemory(create=True, size=4)

    # Set the initial value in shared memory
    shared_mem.buf[0] = 0

    # Create processes
    p1 =multiprocessing.Process(target=process_1,args=(shared_mem,))
    p2 =multiprocessing.Process(target=process_2,args=(shared_mem,))

    # Start processes
    p1.start()
    p2.start()

    try:
        # Join processes
        p1.join()
        p2.join()
    except KeyboardInterrupt:
        # Close shared memory
        shared_mem.close()
        shared_mem.unlink()
        print("Processes Terminated !")
```

➢ Description:

- Implementation is done in python using the multiprocessing module which allows to create processes and shared memory which the processes will access to communicate with each other.
- Two processes are defined, Process-1 for accessing the value from the shared memory and incrementing it every 5 seconds and Process-2 for accessing the current value from shared memory and computing its square.
- Operations of both processes are written in try except blocks to handle Keyboard Interrupts to stop execution and exit gracefully.
- The shared memory segment of size 4 bytes(since we only want to store an integer) is created using the SharedMemory class from the multiprocessing.shared_memory module and the initial value is set to zero.
- Two multiprocessing processes (p1 and p2) are created using the Process class from the multiprocessing module.
- Each process is assigned a target function (process_1 or process_2) to execute, along with the shared memory object as an argument.
- The main program ensures that the processes are started and joined properly, allowing them to execute asynchronously and synchronize their termination.
- In the try block in the main program the p1.join() and p2.join() wait for the processes p1 and p2 to finish.
- If a KeyboardInterrupt exception occurs, the except block is executed. Within the except block, shared memory is closed using shared_mem.close() and the shared memory object is unlinked from the system using shared_mem.unlink() and then the program terminates gracefully.