Ex. no: 7

Name : Ashish P Shaji

Roll NO : 230701041

IPC USING SHARED MEMORY

Aim:

To write a C program to do Inter Process Communication (IPC) using shared memory between sender process and receiver process.

Algorithm:

sender

1. Set the size of the shared memory segment
2. Allocate the shared memory segment using shmget
3. Attach the shared memory segment using shmat
4. Write a string to the shared memory segment using sprintf
5. Set delay using sleep
6. Detach shared memory segment using shmdt

receiver

1. Set the size of the shared memory segment
2. Allocate the shared memory segment using shmget
3. Attach the shared memory segment using shmat
4. Print the shared memory contents sent by the sender process.
5. Detach shared memory segment using shmdt

Program Code:

sender.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <string.h>
#include <unistd.h>

#define SHM_SIZE 1024  // Size of the shared memory segment

int main() {
    key_t key = ftok("shmfile", 65);  // Generate a unique key
    int shmid = shmget(key, SHM_SIZE, 0666 | IPC_CREAT);  // Allocate shared memory segment

    if (shmid == -1) {
        perror("shmget failed");
        exit(1);
    }

    char *str = (char *)shmat(shmid, NULL, 0);  // Attach to shared memory

    if (str == (char *)(-1)) {
        perror("shmat failed");
        exit(1);
    }

    printf("Writing to shared memory...\n");

    sprintf(str, "Welcome to Shared Memory");  // Write message to shared memory
    sleep(2);  // Sleep to give time for receiver to read

    shmdt(str);  // Detach from shared memory
    return 0;
}
```

receiver.c:

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <string.h>

#define SHM_SIZE 1024  // Size of the shared memory segment

int main() {
    key_t key = ftok("shmfile", 65);  // Generate a unique key
    int shmid = shmget(key, SHM_SIZE, 0666 | IPC_CREAT);  // Allocate shared memory segment

    if (shmid == -1) {
        perror("shmget failed");
        exit(1);
    }

    char *str = (char *)shmat(shmid, NULL, 0);  // Attach to shared memory

    if (str == (char *)(-1)) {
        perror("shmat failed");
        exit(1);
    }

    printf("Message Received: %s\n", str);  // Read and print the message from shared memory

    shmdt(str);  // Detach from shared memory
    shmctl(shmid, IPC_RMID, NULL);  // Remove shared memory segment
    return 0;
}
```

```c
    printf("Message Received: %s\n", str);  // Read and print the message from shared memory

    shmdt(str);  // Detach from shared memory
    shmctl(shmid, IPC_RMID, NULL);  // Remove shared memory segment
    return 0;
}
```

OUTPUT :

```
liveuser@localhost-live:~$ gcc sender.c -o sender  # Compile sender.c
./sender                 # Run sender process
Writing to shared memory...
liveuser@localhost-live:~$ gcc receiver.c -o receiver  # Compile receiver.c
./receiver                   # Run receiver process
Message Received: Welcome to Shared Memory
```