Ex. No.: 7

Date: 26/03/2025

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

```
# include <stdio.h>
# include <sys/ipc.h>
# include <sys/shm.h>
# include <unistd.h>
int main(){
        int size = 1024;
        key_t key = ftok("shmfile", 65);
        int shmid = shmget(key, size, 0666 | IPC_CREAT
        char* shared memory = (char*) shmat(shmid,
                                                NULL, 0);
sprintf (shared_memory, "Hello from the sender
                                        Process!");
printf ("sender : Message written to shared
                        memory %s\n", shared_memory);
sleep(5);
Shmdt (shared_memory);
return 0;
}
```

receiver.c

```
# include < stdio.h >
# include < sys | ipc.h >
# include < sys | shm.h >
int main(){
    int size = 1024 ;
    Key_t key = ftok ("shmfile", 65);
    int shmid = shmget(key, size , 0666 | IPC_
                                        CREAT);

    char* shared_memory = (char*) shmat(
                          shmid, NULL, 0);
    printf ("Receiver : Message read from
             shared memory: %s\n", shared_memory);
   shmdt ( shared_memory)
   shmctl ( shmid, IPC_RMID, NULL).
   return 0 ;
}
```

Sample Output

Terminal 1

```
[root@localhost student]# gcc sender.c -o sender
[root@localhost student]# ./sender
```

Terminal 2

```
[root@localhost student]# gcc receiver.c -o receiver
[root@localhost student]# ./receiver
Message Received: Welcome to Shared Memory
[root@localhost student]#
```

OUTPUT :

Sender : Message written to Shared memory :
Hello How are you?

Receiver : Message read from shared memory :
I am fine

Result:

Hence the Inter Process Communication using shared memory has been implemented and executed successfully.