Ex. No.: 7

Date: 28/3/25

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 <stdlib.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <unistd.h>
#include <string.h>
#define SHM_SIZE 1024
```

```
int main() {
    key_t key = ftok("shmfile", 65);
    int shmid = shmget(key, SHMSIZE, 0666 | IPC_CREAT);
    if (shmid == -1) {
        perror("shmget failed");
        exit(1);
    }
    char *shmaddr = (char *) shmat(shmid, NULL, 0);
    if (shmaddr == (char *) -1) {
        perror("shmat failed");
        exit(1);
    }
    printf(shmaddr, "Hello from sender process!");
    printf("Sender: Data written to shared memory: %s\n",
                                        shmaddr);
    sleep(10);
    shmdt(shmaddr);
    return 0;
}
```

receiver.c

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <unistd.h>
#define SHM_SIZE 1024
int main() {
    key_t key = ftok("shmfile", 65);
    int shmid = shmget(key, SHM_SIZE, 0666);
    if (shmid == -1) {
        perror("Shmget failed");
        exit(1);
    }
    char *shmaddr = (char*) shmat(shmid, NULL, 0);
    if (shmaddr == (char *) -1) {
        perror("Shmat failed");
        exit(1);
    }
    printf("Receiver : Data read from Shared memory : %s \n",
                                                    shmaddr);
    shmdt(shmaddr);
    shmdt(shmid, IPL, 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 1 :

Sender : Data written to shared memory : Hello from Sender process!

output 2 :

Reciver : Data read from shared memory : Hello from sender process!

Result:

Thus, the program for inter process communication between sender & reciever is executed successfully.