Ex. No.: 7

Date: 28/3/2025

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 < stdlio .h>
# include < staio .h>
# include < ~~stdio~~ sys/shm.h >
# include < unistd .h>
# include < string .h >
# define SHM_SIZE 1024
```

```
int maind)
Key_t Key= stake ("Shmfile"_65);
int shmid = shmget (Key, SHM_Siye}, 0666/ JMC CREATE)
if (shmid==-1){
        Perror ("shmget failed");
        exit(1);
 }
chat * shmaddr=(char+)Shmat(Shmid, NULL, 0);
    if (shmaddr==(char+)-1){
        perror("Smat failed")
        exit(1);
    }
Printf(shmaddr,"Hello from sender process!");
Print("Sender: Data written to shared memory:
                                   %. 81h", 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 = fork("shmfile", 65);
      int shmid = shmget(Key, SHM_SIZE, 0666);
      if (shmid == -1) {
              ~~Potter~~ Perror("shmget failed");
              exit(1);

      }
char* shmaddr = (char*) shmat(shmid, NULL, 0);

if (shmaddr == (char*)-1) {
   Perror ("shmat failed");
   exit(1);
 }
Print("Reciver :Data read from shared memory: %s\n
                                          shmaddr);

shmdt(shmaddr);
Shmdtl(shmid, IPL, RMID, NULL);
return 0;
}
```

<u>Sample Output</u>

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 return to shared memory: Hello
from sender process!

Output 2 :
Reciver : data read from shared memory
hello from sender process!

Result: Thus program for inter process communication between sender & reciver is executed successfully