

## Practical – 7

### Aim: -

Implement TWO-WAY Inter-process communication using Shared Memory between independent processes. This communication must continue till a specific key is pressed or a STOP message is sent by any one of the processes.

### Theory: -

Inter Process Communication through shared memory is a concept where two or more process can access the common memory. And communication is done via this shared memory where changes made by one process can be viewed by another process.

The problem with pipes, fifo and message queue – is that for two process to exchange information. The information has to go through the kernel.

- Server reads from the input file.
- The server writes this data in a message using either a pipe, fifo or message queue.
- The client reads the data from the IPC channel, again requiring the data to be copied from kernel's IPC buffer to the client's buffer.
- Finally, the data is copied from the client's buffer.

A total of four copies of data are required (2 read and 2 write). So, shared memory provides a way by letting two or more processes share a memory segment. With Shared Memory the data is only copied twice – from input file into shared memory and from shared memory to the output file.

Source Code: - **C Language**

#### **Server.c**

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

```
#define SHMSZ 27
```

```
int main()
{
```

```
int shmid,i;
key_t key;
char *shm, *s;
key = 1000;
if ((shmid = shmget(key, SHMSZ, IPC_CREAT | 0666)) < 0) {
    perror("shmget");
    exit(1);
}
if ((shm = shmat(shmid, NULL, 0)) == (char *) -1) {
    perror("shmat");
    exit(1);
}
*shm = '!';
while(*shm!='*')
{
    char a[100];
    printf("mesaage to send (*to end):");
    gets(a);
    if(a[0]!='*'){
        s = shm+1;
        for (i=0;i<strlen(a);i++)
            *s++ = a[i];
        *s = NULL;
        *shm = '@';

        while(*shm!='#'&&*shm!='*')
            sleep(1);

        if(*shm!='*'){
            printf("client:");
            for (s = shm+1; *s != NULL; s++)
                putchar(*s);
            putchar('\n');
        }else
            printf("connection closed by client");
    }
    else
        *shm = '*';
}
shmid = shmdt(shm);
exit(0);
}
```

### Client.c

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

#define SHMSZ 27

int main()
{
    int shmid,i;
    key_t key;
    char *shm, *s;
    key = 1000; //key for data segment created by server, must be same
    as that of server

    if ((shmid = shmget(key, SHMSZ, 0666)) < 0) { //locate data
    segment using key and get its id
        perror("shmget");
        exit(1);
    }
    if ((shm = shmat(shmid, NULL, 0)) == (char *) -1) {
        perror("shmat");
        exit(1);
    }

    *shm = '!';
    while(*shm!='*')
    {
        while(*shm!='@'&&*shm!='*')
            sleep(1);
        if(*shm!='*'){
            printf("server:");
            for (s = shm+1; *s != NULL; s++)
                putchar(*s);
            putchar('\n');

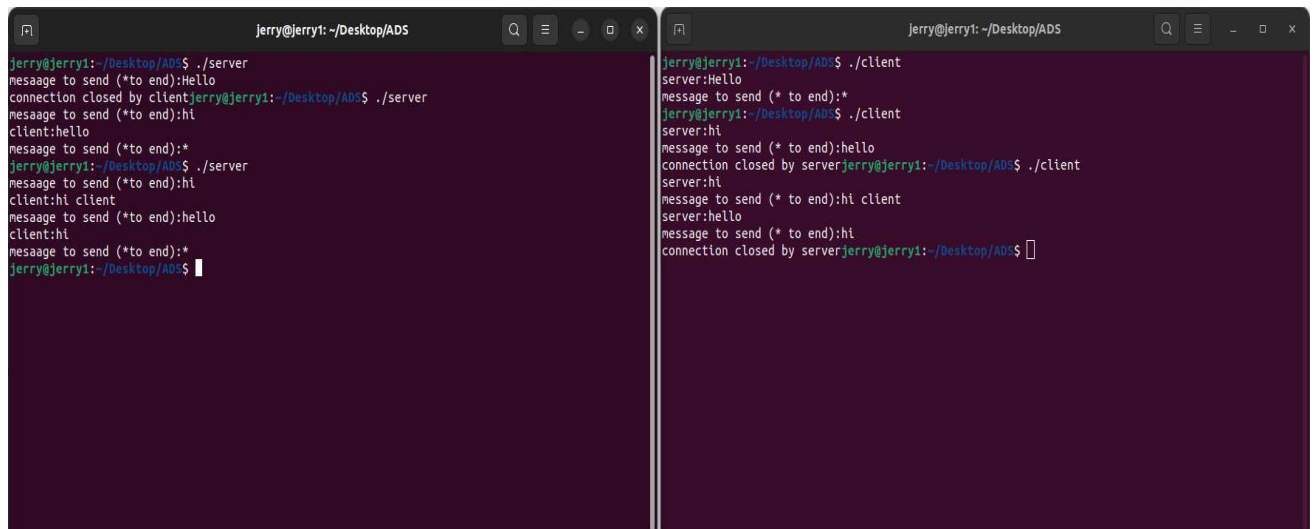
            char a[100];
            printf("message to send (* to end):");
            gets(a);
```

```
        if(a[0]!='*'){
            s=shm+1;
            for (i=0;i<strlen(a);i++)
                *s++ = a[i];
            *s = NULL;
            *shm = '#';
        }
        else
            *shm = '*';
    }
    else
        printf("connection closed by server");
}

shmidx = shmidx(shm);

exit(0);
}
```

**Output: -**



```
jerry@jerry1: ~/Desktop/ADS
jerry@jerry1:~/Desktop/ADS$ ./server
message to send (*to end):Hello
connection closed by clientjerry@jerry1:~/Desktop/ADS$ ./server
message to send (*to end):hi
client:hello
message to send (*to end):*
jerry@jerry1:~/Desktop/ADS$ ./server
message to send (*to end):hi
client:hi client
message to send (*to end):hello
client:hi
message to send (*to end):*
jerry@jerry1:~/Desktop/ADS$
```

```
jerry@jerry1:~/Desktop/ADS$ ./client
server:Hello
message to send (* to end):*
jerry@jerry1:~/Desktop/ADS$ ./client
server:hi
message to send (* to end):hello
connection closed by serverjerry@jerry1:~/Desktop/ADS$ ./client
server:hi
message to send (* to end):hi client
server:hello
message to send (* to end):hi
connection closed by serverjerry@jerry1:~/Desktop/ADS$
```