Ex. No.: 7 **IPC USING SHARED MEMORY**

Date:19.02.25

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 <unistd.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
```

```
#include <string.h>

#define SIZE 5

int main() {
    int shm_id;
    key_t key;
    char *shm_mem;
    char buffer[SIZE + 2];

    key = 5677;

    if ((shm_id = shmget(key, SIZE + 1, IPC_CREAT | 0666)) == -1) {
        perror("shmget");
        exit(1);
    }

    if ((shm_mem = (char *)shmat(shm_id, NULL, 0)) == (char *)-1) {
        perror("shmat");
        exit(1);
    }

    printf("Enter data (max %d characters): ", SIZE);
    fgets(buffer, sizeof(buffer), stdin);
    buffer[strcspn(buffer, "\n")] = 0;

    int len = strlen(buffer);
    if (len == 0) {
        printf("Sender: Buffer empty!\n");
        sprintf(shm_mem, "EMPTY");
    } else if (len > SIZE) {
        printf("Sender: Buffer full! Only %d characters allowed.\n", SIZE);
        sprintf(shm_mem, "FULL");
    } else {
        strcpy(shm_mem, buffer);
        printf("Sender: Data written to shared memory: %s\n", buffer);
    }

    sleep(5);

    if (shmdt(shm_mem) == -1) {
        perror("shmdt");
        exit(1);
    }
```

```
    printf("Sender: Shared memory detached\n");

    return 0;
}
```

receiver.c

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

#define SIZE 5

int main() {
    int shm_id;
    key_t key;
    char *shm_mem;

    key = 5677;

    if ((shm_id = shmget(key, SIZE + 1, 0666)) == -1) {
        perror("shmget");
        exit(1);
    }

    if ((shm_mem = (char *)shmat(shm_id, NULL, 0)) == (char *)-1) {
        perror("shmat");
        exit(1);
    }

    if (strcmp(shm_mem, "EMPTY") == 0) {
        printf("Buffer is empty!\n");
    } else if (strcmp(shm_mem, "FULL") == 0) {
        printf("Buffer was full in sender!\n");
    } else {
        printf("Data read from shared memory: %s\n", shm_mem);
    }
```

```
    if (shmdt(shm_mem) == -1) {
        perror("shmdt");
        exit(1);
    }

    if (shmctl(shm_id, IPC_RMID, NULL) == -1) {
        perror("shmctl");
        exit(1);
    }

    printf("Shared memory removed successfully\n");

    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]#
```

```
└─$ vi sender.c


┌──(student㉿kali)-[~]
└─$ gcc sender.c -o sender



┌──(student㉿kali)-[~]
└─$ ./sender
Enter data (max 5 characters): abcdefg
Sender: Buffer full! Only 5 characters allowed.
Sender: Shared memory detached

┌──(student㉿kali)-[~]
└─$ ./sender
Enter data (max 5 characters): abcde
Sender: Data written to shared memory: abcde
Sender: Shared memory detached

┌──(student㉿kali)-[~]
└─$ ./sender
Enter data (max 5 characters):
Sender: Buffer empty!
Sender: Shared memory detached
```

Result:

Thus ,the program to do Inter Process Communication (IPC) using shared memory between sender process and receiver process has been executed successfully.