

Ex. No.: 7

Date:

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 <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#define SharedMemSize 50
void main()
{
    char c;
    int shmid;
    key_t key;
```

```
char * shared-memory;
```

```
key = 5677;
```

```
key = 5677;
```

```
if (shmids = shmget (key, SharedMemSize, IPC_CREAT|0666)) < 0
```

```
{
```

```
    perror ("shmget");
```

```
    exit(1);
```

```
}
```

```
if ((shared-memory = shmat (shmids, NULL, 0)) == (char *) -1) {
```

```
    perror ("shmat");
```

```
    exit(1);
```

```
}
```

```
printf (shared-memory, "Welcome to Shared Memory");
```

```
sleep(2);
```

```
exit(0);
```

```
}
```



receiver.c

```
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdio.h>
#include <stdlib.h>
#define SharedMemSize 50
```

```
void main()
```

```
{
```

```
    int shmid
```

```
    key_t key;
```

```
    char * shared_memory
```

```
    key = 5677
```

```
    if ((shmid = shmget(key, SharedMemSize, 0666)) < 0) {
        perror("shmget");
        exit(1);
    }
```

```
}
```

```
if ((shared_memory = shmat(shmid, NULL, 0)) == (char *) -1) {
    perror("shmat");
    exit(1);
}
```

```
printf("Message received : %s\n", shared_memory);
exit(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]#
```

shared memory attached at address: 0x7f8c5a3b6000
Message written to shared memory: Welcome to shared memory

shared memory attached at address: 0x7f9d01b6000
Message ~~written~~ read from shared memory: Welcome to shared memory



Result:

Thus the program for inter process communication between sender & receiver is executed successfully