**Musa Haji  
230701391**

**Ex No.7  
 Interprocess Communication**

Aim:

To write a C program to implement Inter-Process Communication (IPC) using shared memory between sender and receiver processes.

Algorithm:

Sender Process:

Set the size of the shared memory segment.

Allocate the shared memory segment using shmget.

Attach the shared memory segment using shmat.

Write a string to the shared memory segment using sprintf.

Set delay using sleep.

Detach the shared memory segment using shmdt.

Receiver Process:

Set the size of the shared memory segment.

Allocate the shared memory segment using shmget.

Attach the shared memory segment using shmat.

Print the shared memory contents sent by the sender process.

Detach the shared memory segment using shmdt.

Program Code:

sender.c

#include <stdio.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <string.h>

#include <unistd.h>

#define SHMSIZE 1024

int main() {

key\_t key = 1234; // Shared memory key

int shmid;

char \*shm\_ptr;

// Create shared memory segment

shmid = shmget(key, SHMSIZE, 0666 | IPC\_CREAT);

if (shmid == -1) {

perror("shmget failed");

return 1;

}

// Attach shared memory segment

shm\_ptr = (char \*)shmat(shmid, NULL, 0);

if (shm\_ptr == (char \*)-1) {

perror("shmat failed");

return 1;

}

// Write to shared memory

sprintf(shm\_ptr, "Welcome to Shared Memory");

// Sleep to simulate delay

sleep(2);

// Detach shared memory segment

shmdt(shm\_ptr);

return 0;

}

receiver.c

#include <stdio.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <string.h>

#define SHMSIZE 1024

int main() {

key\_t key = 1234; // Shared memory key

int shmid;

char \*shm\_ptr;

// Get shared memory segment

shmid = shmget(key, SHMSIZE, 0666);

if (shmid == -1) {

perror("shmget failed");

return 1;

}

// Attach shared memory segment

shm\_ptr = (char \*)shmat(shmid, NULL, 0);

if (shm\_ptr == (char \*)-1) {

perror("shmat failed");

return 1;

}

// Print the message from shared memory

printf("Message Received: %s\n", shm\_ptr);

// Detach shared memory segment

shmdt(shm\_ptr);

return 0;

}

Sample Output:

Terminal 1 (Sender):

[root@localhost student]# gcc sender.c -o sender

[root@localhost student]# ./sender

Terminal 2 (Receiver):

[root@localhost student]# gcc receiver.c -o receiver

[root@localhost student]# ./receiver

Message Received: Welcome to Shared Memory

[root@localhost student]#

Result:

The program successfully demonstrates Inter-Process Communication (IPC) using shared memory. The sender process writes a message to the shared memory, and the receiver process reads and prints the message.