**Experiment:9: Illustrate the concept of inter-process communication using shared memory with a C program.**

**Aim:**

The aim of this program is to illustrate **Inter-Process Communication (IPC) using Shared Memory** in C. Shared memory allows two or more processes to communicate by accessing a common memory space. One process writes data to the shared memory, and another process reads the data from it.

**Procedure:**

1. **Create Shared Memory:**
   * Use shmget() to create or access a shared memory segment.
   * shmget() returns a shared memory identifier that can be used to attach the shared memory.
2. **Attach the Shared Memory:**
   * Use shmat() to attach the shared memory segment to the process's address space.
3. **Write to Shared Memory:**
   * One process writes data into the shared memory.
4. **Read from Shared Memory:**
   * Another process reads the data from the shared memory.
5. **Detach and Remove Shared Memory:**
   * After the processes finish using the shared memory, they should detach from it using shmdt() and remove it using shmctl() to free resources.

**Steps in the Program:**

1. A **producer process** will write data into the shared memory.
2. A **consumer process** will read the data from the shared memory.

**C Program Implementation:**

#include <stdio.h>

#include <stdlib.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <string.h>

#define SHM\_KEY 1234 // Shared memory key

int main() {

int shmid;

char \*shared\_mem;

// Create shared memory segment

shmid = shmget(SHM\_KEY, 1024, IPC\_CREAT | 0666);

if (shmid == -1) {

perror("shmget failed");

exit(1);

}

// Attach the shared memory segment

shared\_mem = (char \*)shmat(shmid, NULL, 0);

if (shared\_mem == (char \*)(-1)) {

perror("shmat failed");

exit(1);

}

// Write data to shared memory

printf("Enter data to write to shared memory: ");

fgets(shared\_mem, 1024, stdin); // User input to shared memory

// Detach the shared memory segment

if (shmdt(shared\_mem) == -1) {

perror("shmdt failed");

exit(1);

}

printf("Data written to shared memory successfully!\n");

return 0;

}

Output:

