Ex. No: 5 IMPLEMENTATION OF SHARED MEMORY AND IPC

AIM

To write a C program to illustrate interprocess communication using Shared Memory system calls.

ALGORITHM

1. Create shared memory using shmget( ) system call
2. If successfull it returns positive value
3. Attach the created shared memory using shmat( ) systemcall.
4. Write to shared memory using shmsnd( ) system call
5. Read the contents from shared memory using shmrcv( ) systemcall

PROGRAM

```
#include <sys/types.h>
#include <sys/shm.h>
#include <sys/ipc.h>
#include <stdio.h>
#include <stdlib.h>
int main()
{
        int shmid;
        key_t key = 0 * 10;
        shmid = shmget(key, 100, IPC_CREAT | 0666);
        if (shmid < 0)
                printf("First SHMID failed\n\n");
        else
                printf("First SHMID succeded id=%d\n\n", shmid);
        shmid = shmget(key, 101, IPC_CREAT | 0666);
        if (shmid < 0)
                printf("Second SHMID failed\n\n");
        else
                printf("Secondt SHMID succeded id=%d\n\n", shmid);
        shmid = shmget(key, 90, IPC_CREAT | 0666);
        if (shmid < 0)
                printf("Third SHMID failed\n\n");
        else
                printf("Third SHMID succeded id=%d\n\n", shmid);
}
```

Ex.No: 6 **IMPLEMENTATION OF SEMAPHORE**

AIM

To implement Producer Consumer problem using Semaphore.

ALGORITHM

1. Union a variable “mysemun” as integer type to implement semaphore.
2. A buffer “sembuf” variable for producer consumer is written
3. In producer, buffer “sembuf” full condition is checked and if it is full then it is displayed.
4. In consumer, buffer “sembuf” empty condition is checked and a message displayed.
5. Producer produces an element sequentially from main() and a message displayed.
6. Consumer consumes an element sequentially from main() and a message displayed.
7. The producer consumer produces messages are displayed till the user defines in the program.

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>

int mutex = 1, full = 0, empty = 3, x = 0;

void producer();
void consumer();
int wait(int);
int signal(int);

int main() {
   int n;
   printf("\n1.PRODUCER\n2.CONSUMER\n3.EXIT\n");

   while (1) {
      printf("\nENTER YOUR CHOICE : ");
      scanf("%d", &n);

      if (mutex == 1) {
         switch (n) {
            case 1:
               if (empty != 0) {
                  producer();
               } else {
                  printf("BUFFER IS FULL\n");
               }
               break;
            case 2:
               if (full != 0) {
```

```
                consumer();
            } else {
                printf("BUFFER IS EMPTY\n");
            }
            break;
        case 3:
            exit(1);
        }
    }
  }
}

int wait(int s) {
  return --s;
}

int signal(int s) {
  return ++s;
}

void producer() {
  mutex = wait(mutex);
  full = signal(full);
  empty = wait(empty);
  x++;
  printf("Producer produces the item %d\n", x);
  mutex = signal(mutex);
}

void consumer() {
  mutex = wait(mutex);
  full = wait(full);
  empty = signal(empty);
  printf("Consumer consumes item %d\n", x);
  x--;
  mutex = signal(mutex);
}
```