Linux-IPC-Semaphores

Ex05-Linux IPC-Semaphores

AIM:

To Write a C program that implements a producer-consumer system with two processes using Semaphores.

DESIGN STEPS:

Step 1:

Navigate to any Linux environment installed on the system or installed inside a virtual environment like virtual box/vmware or online linux JSLinux (https://bellard.org/jslinux/vm.html?url=alpine-x86.cfg&mem=192) or docker.

Step 2:

Write the C Program using Linux Process API - Sempahores

Step 3:

Execute the C Program for the desired output.

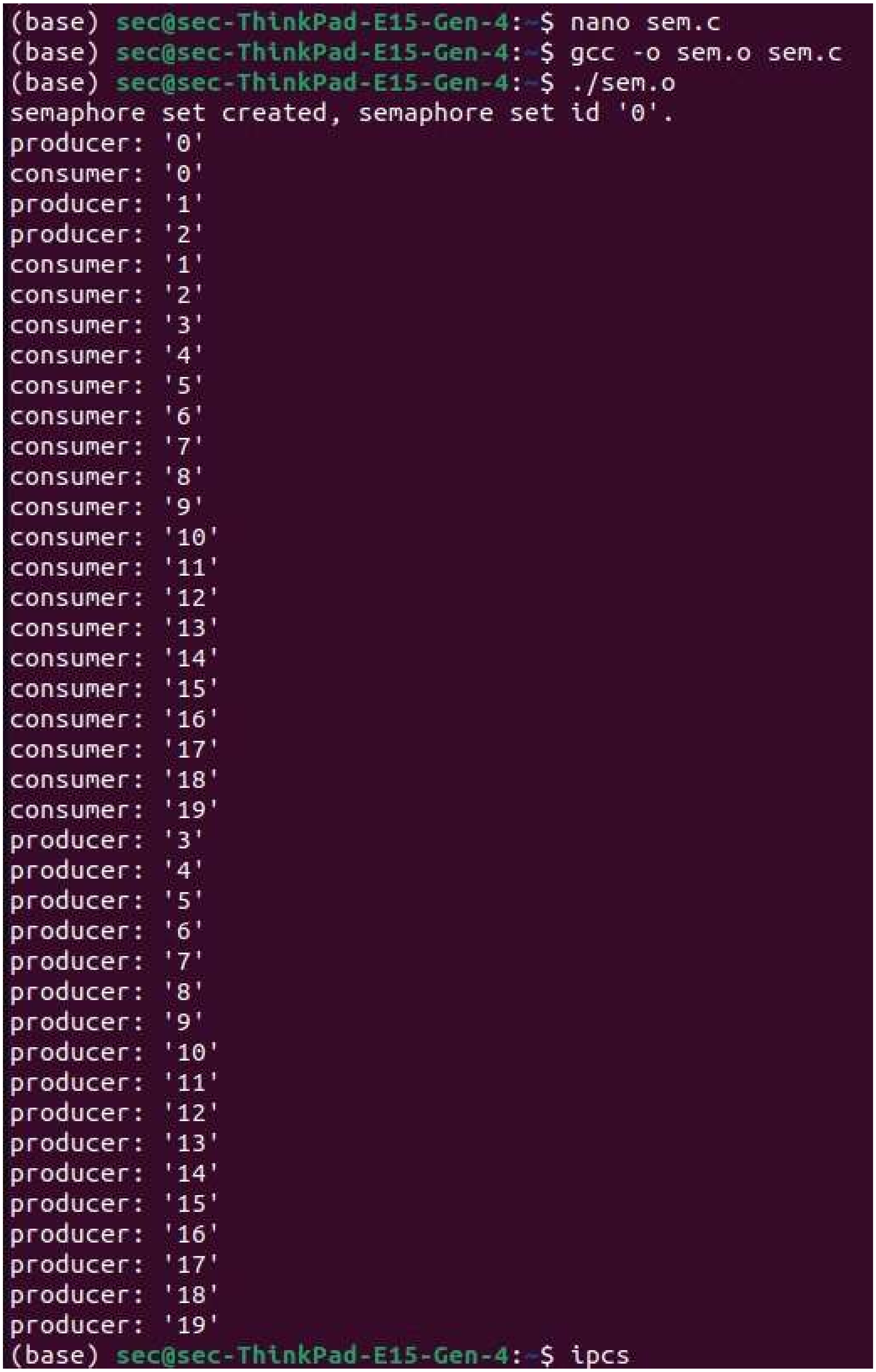
PROGRAM:

Write a C program that implements a producer-consumer system with two processes using Semaphores.

```
/*
 * sem-producer-consumer.c  - demonstrates a basic producer-consumer
 *                            implementation.
 */
#include <stdio.h>       /* standard I/O routines.              */
#include <stdlib.h>      /* rand() and srand() functions        */
#include <unistd.h>      /* fork(), etc.                        */
#include <time.h>        /* nanosleep(), etc.                   */
#include <sys/types.h>   /* various type definitions.           */
#include <sys/ipc.h>     /* general SysV IPC structures         */
#include <sys/sem.h>     /* semaphore functions and structs.    */
#define NUM_LOOPS       20       /* number of loops to perform. */
#if defined(_GNU_LIBRARY_) && !defined(_SEM_SEMUN_UNDEFINED)
/* union semun is defined by including <sys/sem.h> */
#else
/* according to X/OPEN we have to define it ourselves */
union semun {
        int val;                    /* value for SETVAL */
        struct semid_ds buf;       / buffer for IPC_STAT, IPC_SET */
        unsigned short int array;  / array for GETALL, SETALL */
        struct seminfo __buf;      / buffer for IPC_INFO */
};
#endif
int main(int argc, char* argv[])
{
    int sem_set_id;           /* ID of the semaphore set.       */
    union semun sem_val;      /* semaphore value, for semctl(). */
    int child_pid;            /* PID of our child process.      */
    int i;                    /* counter for loop operation.    */
    struct sembuf sem_op;     /* structure for semaphore ops.   */
    int rc;                   /* return value of system calls.  */
    struct timespec delay;    /* used for wasting time.         */
/* create a private semaphore set with one semaphore in it, */
    /* with access only to the owner.                           */
    sem_set_id = semget(IPC_PRIVATE, 1, 0600);
    if (sem_set_id == -1) {
        perror("main: semget");
        exit(1);
    }
    printf("semaphore set created, semaphore set id '%d'.\n", sem_set_id);
    /* intialize the first (and single) semaphore in our set to '0'. */
    sem_val.val = 0;
    rc = semctl(sem_set_id, 0, SETVAL, sem_val);
    /* fork-off a child process, and start a producer/consumer job. */
    child_pid = fork();
    switch (child_pid) {
        case -1:        /* fork() failed */
            perror("fork");
            exit(1);
        case 0:         /* child process here */
            for (i=0; i<NUM_LOOPS; i++) {
                /* block on the semaphore, unless it's value is non-negative. */
                sem_op.sem_num = 0;
                sem_op.sem_op = -1;
                sem_op.sem_flg = 0;
                semop(sem_set_id, &sem_op, 1);
                printf("consumer: '%d'\n", i);
                fflush(stdout);
            }
            break;
        default:        /* parent process here */
            for (i=0; i<NUM_LOOPS; i++) {
                printf("producer: '%d'\n", i);
                fflush(stdout);
                /* increase the value of the semaphore by 1. */
                sem_op.sem_num = 0;
                                        sem_op.sem_op = 1;
                sem_op.sem_flg = 0;
                semop(sem_set_id, &sem_op, 1);
                /* pause execution for a little bit, to allow the */
                /* child process to run and handle some requests. */
                /* this is done about 25% of the time.            */
                if (rand() > 3*(RAND_MAX/4)) {
                    delay.tv_sec = 0;
                    delay.tv_nsec = 10;
                    //nanosleep(&delay, NULL);
                                      sleep(10); }
if(NUM_LOOPS>=10)    {
            semctl(sem_set_id, 0, IPC_RMID, sem_val) ;} // Remove the sem_set_id
            }}
            break;
    }
    return 0;}
```

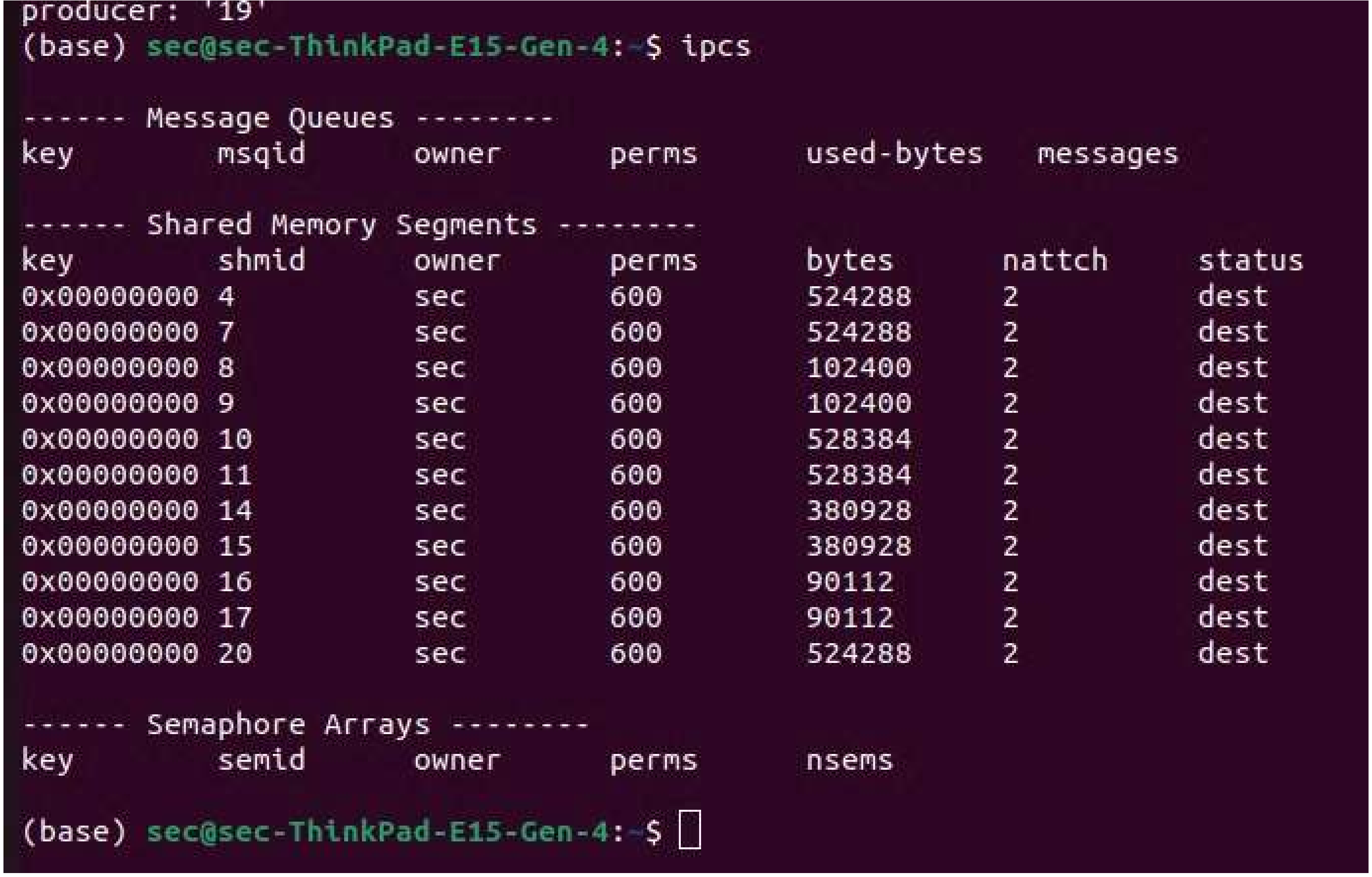
OUTPUT

$ ./sem.o



```
(base) sec@sec-ThinkPad-E15-Gen-4:~$ nano sem.c
(base) sec@sec-ThinkPad-E15-Gen-4:~$ gcc -o sem.o sem.c
(base) sec@sec-ThinkPad-E15-Gen-4:~$ ./sem.o
semaphore set created, semaphore set id '0'.
producer: '0'
consumer: '0'
producer: '1'
producer: '2'
consumer: '1'
consumer: '2'
consumer: '3'
consumer: '4'
consumer: '5'
consumer: '6'
consumer: '7'
consumer: '8'
consumer: '9'
consumer: '10'
consumer: '11'
consumer: '12'
consumer: '13'
consumer: '14'
consumer: '15'
consumer: '16'
consumer: '17'
consumer: '18'
consumer: '19'
producer: '3'
producer: '4'
producer: '5'
producer: '6'
producer: '7'
producer: '8'
producer: '9'
producer: '10'
producer: '11'
producer: '12'
producer: '13'
producer: '14'
producer: '15'
producer: '16'
producer: '17'
producer: '18'
producer: '19'
(base) sec@sec-ThinkPad-E15-Gen-4:~$ ipcs
```

$ ipcs



```
producer: '19'
(base) sec@sec-ThinkPad-E15-Gen-4:~$ ipcs

------ Message Queues --------
key        msqid      owner      perms      used-bytes   messages

------ Shared Memory Segments --------
key        shmid      owner      perms      bytes      nattch     status
0x00000000 4          sec        600        524288     2          dest
0x00000000 7          sec        600        524288     2          dest
0x00000000 8          sec        600        102400     2          dest
0x00000000 9          sec        600        102400     2          dest
0x00000000 10         sec        600        528384     2          dest
0x00000000 11         sec        600        528384     2          dest
0x00000000 14         sec        600        380928     2          dest
0x00000000 15         sec        600        380928     2          dest
0x00000000 16         sec        600        90112      2          dest
0x00000000 17         sec        600        90112      2          dest
0x00000000 20         sec        600        524288     2          dest

------ Semaphore Arrays --------
key        semid      owner      perms      nsems

(base) sec@sec-ThinkPad-E15-Gen-4:~$ 
```

RESULT: The program is executed successfully.

RESULT:

The program is executed successfully.