**Name:-** Dishant Modh

**Roll No:-** IT076

**EXPERIMENT 8**

**Aim:** Study system calls related to semaphore.

Study system calls: sem\_init(), sem\_destroy(), sem\_wait(), sem\_post()

**Problem 1.** Solve producer consumer problem using semaphore. (Single producer and single consumer)

**2.** Solve producer consumer problem (multiple producer multiple consumer) using semaphore. Tools: Linux OS, gcc compiler, pthread.so

**1. Solve producer consumer problem using semaphore. (Single producer and single consumer)**

**Single.c**

#include <stdio.h>

#include <pthread.h>

#include <semaphore.h>

*int* buf[5], f, r;

sem\_t mutex, full, empty;

*void* \*produce(*void* \**arg*)

{

*int* i;

for (i = 0; i < 10; i++)

{

sem\_wait(&empty);

sem\_wait(&mutex);

printf("produced item is %d\n", i);

buf[(++r) % 5] = i;

sleep(1);

sem\_post(&mutex);

sem\_post(&full);

printf("full %u\n", full);

}

}

*void* \*consume(*void* \**arg*)

{

*int* item, i;

for (i = 0; i < 10; i++)

{

sem\_wait(&full);

printf("full %u\n", full);

sem\_wait(&mutex);

item = buf[(++f) % 5];

printf("consumed item is %d\n", item);

sleep(1);

sem\_post(&mutex);

sem\_post(&empty);

}

}

main()

{

pthread\_t tid1, tid2;

sem\_init(&mutex, 0, 1);

sem\_init(&full, 0, 1);

sem\_init(&empty, 0, 5);

pthread\_create(&tid1, NULL, produce, NULL);

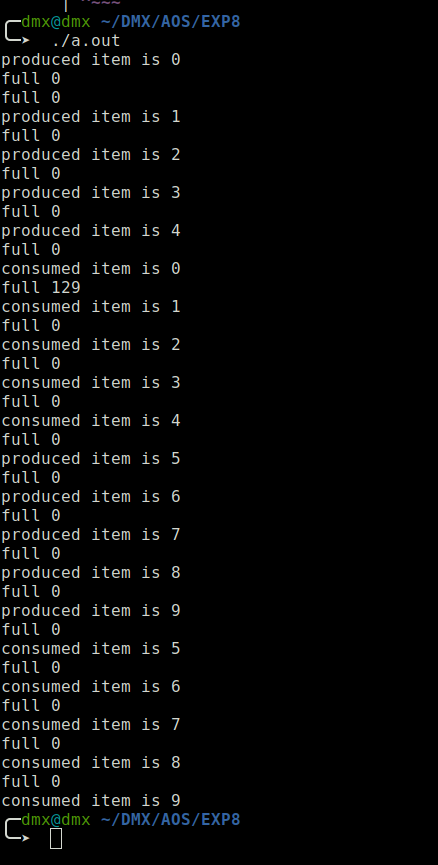
pthread\_create(&tid2, NULL, consume, NULL);

pthread\_join(tid1, NULL);

pthread\_join(tid2, NULL);

}

**Output:-**

****

**2. Solve producer consumer problem (multiple producer multiple consumer) using semaphore.**

**Multi.c**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <pthread.h>

#include <semaphore.h>

#define MAX 10

*int* in, out, itemIn, itemOut, maxToProduce, noOfProducer, noOfConsumer,

buffer[MAX];

sem\_t empty, full, mutex;

*void* \*produceItem(*void* \**pid*)

{

*int* \*pro = (*int* \*)*pid*;

for (*int* i = 0; i < maxToProduce; i++)

{

sem\_wait(&empty);

sem\_wait(&mutex);

buffer[in] = (++itemIn);

printf("Producer %d: Produce item %d at %d.\n", \*pro, itemIn, in);

in = (in + 1) % MAX;

sem\_post(&mutex);

sem\_post(&full);

sleep(noOfProducer);

}

}

*void* \*consumeItem(*void* \**cid*)

{

*int* \*con = (*int* \*)*cid*;

while (1)

{

if (itemOut == (noOfProducer \* maxToProduce))

return NULL;

sem\_wait(&full);

sem\_wait(&mutex);

itemOut = buffer[out];

printf("Consumer %d: Consume item %d from %d.\n", \*con, itemOut,

out);

out = (out + 1) % MAX;

sem\_post(&mutex);

sem\_post(&empty);

sleep(noOfConsumer);

}

}

*int* main()

{

pthread\_t producer[3], consumer[3];

*int* i;

*int* a[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

printf("\nEnter total no of Producer : ");

scanf("%d", &noOfProducer);

printf("Enter total no of Consumer : ");

scanf("%d", &noOfConsumer);

printf("Enter total no item one producer can produce : ");

scanf("%d", &maxToProduce);

sem\_init(&mutex, 0, 1);

sem\_init(&empty, 0, MAX);

sem\_init(&full, 0, 0);

printf("\tSemaphore Created...\n");

printf("\tProducer-Consumer Creating...\n");

for (i = 0; i < noOfProducer; i++)

{

pthread\_create(&producer[i], NULL, produceItem, (*void* \*)&a[i]);

sleep(1);

}

for (i = 0; i < noOfConsumer; i++)

{

pthread\_create(&consumer[i], NULL, consumeItem, (*void* \*)&a[i]);

sleep(1);

}

for (i = 0; i < noOfProducer; i++)

pthread\_join(producer[i], NULL);

for (i = 0; i < noOfConsumer; i++)

pthread\_join(consumer[i], NULL);

printf("\tProducer-Consumer Destroying...\n");

sem\_destroy(&full);

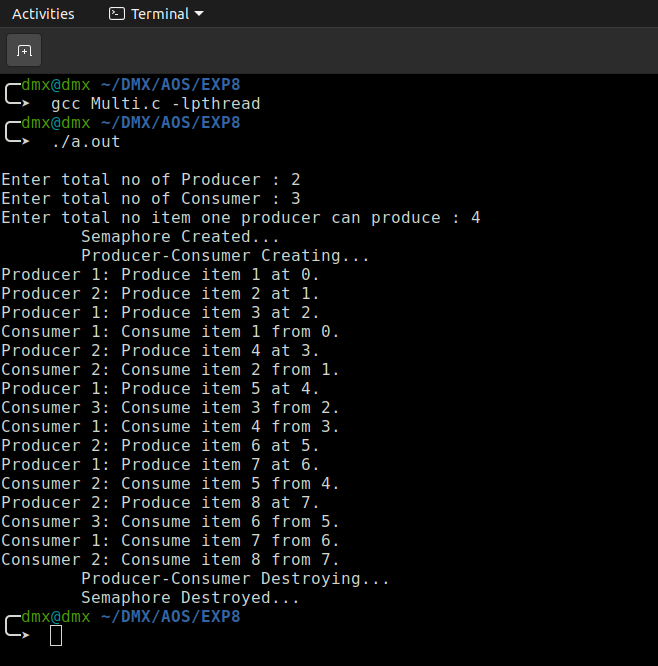
sem\_destroy(&empty);

sem\_destroy(&mutex);

printf("\tSemaphore Destroyed...\n");

return 0;

**Output:-**

****