**EXPERIMENT 9**

**OBJECTIVE**:-

Program to design consumer producer problem using semaphores.

**CODE**:-

//------------------------------------------------------------------------------

//

// Desc: consumer\_producer.c

//

// Date: 18/4/2018

//

// Author: VU DUY DU - 2k16/CO/364

//

//------------------------------------------------------------------------------

#include<iostream>

#include<pthread.h>

#include<semaphore.h>

#include<unistd.h>

#include<stdio.h>

#include<string>

#define LOOP\_NUM 3

sem\_t semaphore;

int item;

void\* consumer\_thread(void\*arg){

printf("\nConsumer Thread Entered\n\n");

for(int i = 0; i < LOOP\_NUM; i++){

//wait

sem\_wait(&semaphore);

printf("\nConsumer: Entered...\n");

//critical Section

sleep(2);

//item --;

std::cout<<"Consumer: Accessed item "<<item<<"\n";

//signal

printf("Consumer: Just Exiting...\n");

sem\_post(&semaphore);

sleep(1);

}

return NULL;

}

void\* producer\_thread(void\*arg){

printf("\nProducer thread Entered\n\n");

for(int i = 0; i < LOOP\_NUM; i++){

//wait

sem\_wait(&semaphore);

printf("\nProducer: Entered...\n");

//critical Section

sleep(2);

item ++;

std::cout<<"Producer: Producted item "<<item<<"\n";

//signal

printf("Producer: Just Exiting...\n");

sem\_post(&semaphore);

sleep(1);

}

return NULL;

}

int main(){

printf("hello semaphore ");

std::cout<<sem\_init(&semaphore, 0,1);

//create 2 threads for consumer and producer

pthread\_t t1,t2;

pthread\_create(&t1,NULL,producer\_thread,NULL);

pthread\_create(&t2,NULL,consumer\_thread,NULL);

//threads are done their jobs

pthread\_join(t1,NULL);

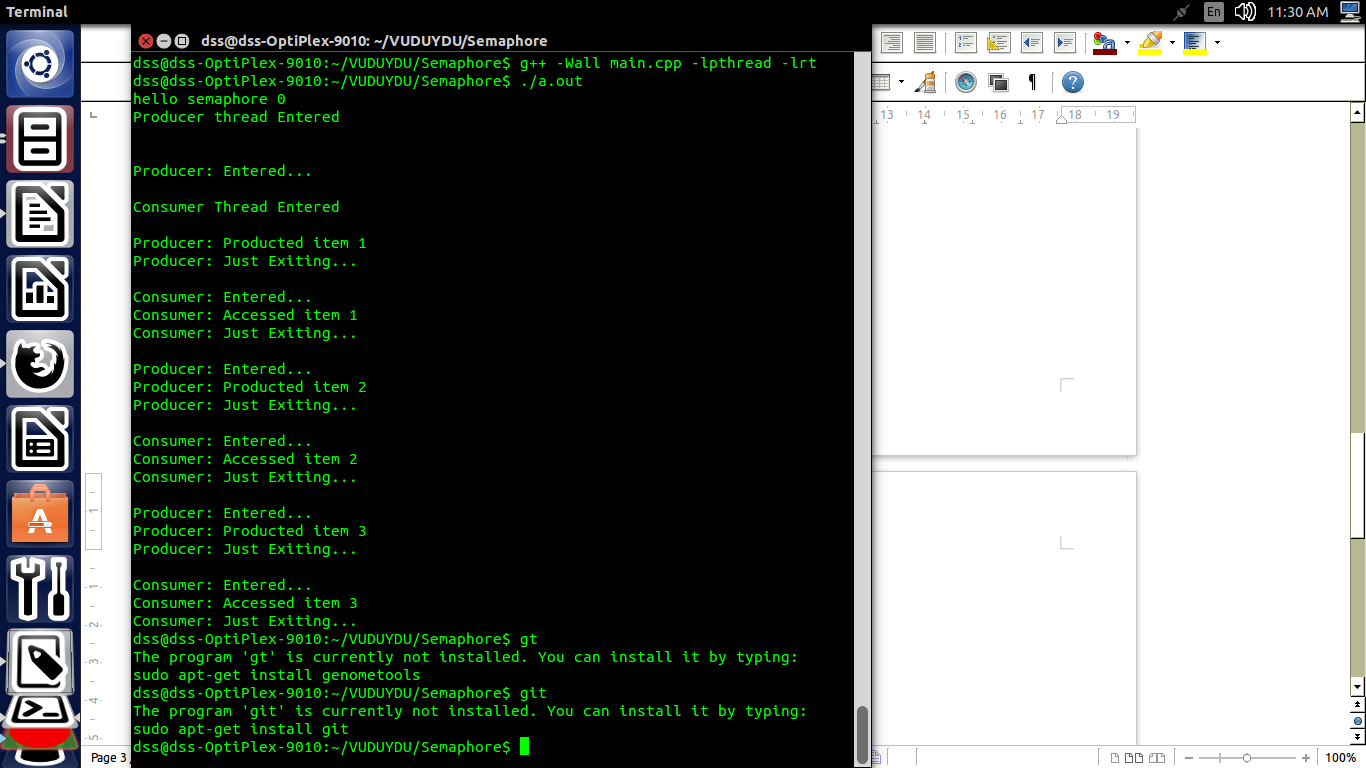
pthread\_join(t2,NULL);

//destroy the semaphore

sem\_destroy(&semaphore);

return 0;

}

**OUTPUT:-**

**DISCUSSION:-**

Producer consumer problem is also known as bounded buffer problem. In this problem we have two processes, producer and consumer, who share a fixed size buffer. Producer work is to produce data or items and put in buffer. Consumer work is to remove data from buffer and consume it. We have to make sure that producer do not produce data when buffer is full and consumer do not remove data when buffer is empty.

*Source code of this experiment can be found here:*

*https://github.com/Sieunguoimay/OSLab-4thsem-2018/tree/master/Exp9*