Thread Synchronization Using Counting Semaphore(Producer , Consumer Problem)

#include<stdio.h>

#include<semaphore.h>

#include<sys/types.h>

#include<pthread.h>

#include<unistd.h>

#include<stdlib.h>

#define BUFFER\_SIZE

10 pthread\_mutex\_t mutex; sem\_t empty,full; int buffer[BUFFER\_SIZE];

int counter; pthread\_t

tid;

void \*producer(); void \*consumer(); void insert\_item(int);

int remove\_item(); void initilize()

{

pthread\_mutex\_init(&mutex,NULL); sem\_init(&full,0,0); sem\_init(&empty,0,BUFFER\_SIZE);

}

void \*producer()

{

int item,wait\_time; wait\_time=rand()%5; sleep(wait\_time)%5; item=rand()%10; sem\_wait(&empty); pthread\_mutex\_lock(&mutex); printf("Producer produce

%d\n\n",item); insert\_item(item); pthread\_mutex\_unlock(&mutex);

sem\_post(&full);

}

void \*consumer()

{

int item,wait\_time;

wait\_time=rand()%5;

sleep(wait\_time); sem\_wait(&full);

pthread\_mutex\_lock(&mutex);

item=remove\_item(); printf("Consumer consume %d\n\n",item); pthread\_mutex\_unlock(&mutex); sem\_post(&empty);

}

void insert\_item(int item)

{

buffer[counter++]=item;

}

int remove\_item()

{

return buffer[--counter];

}

int main()

{

int n1,n2; int i; printf("Enter number of Producers: "); scanf("%d",&n1); printf("Enter number of Consumers: ");

scanf("%d",&n2);

initilize(); for(i=0;i<n1;i++)

pthread\_create(&tid,NULL,producer,NULL); for(i=0;i<n2;i++)

pthread\_create(&tid,NULL,consumer,NULL); sleep(5);

exit(0);

}