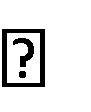
Name : Andhale Nitin Parasram

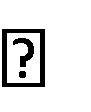
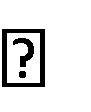
Div- 1

Roll no – I3111

ASSIGNMENTNO: 4 (A)

THREAD SYNCHRONIZATION

AIM: Thread synchronization using counting semaphores and mutual exclusion using mutex. Application to demonstrate: Producer Consumer problem with counting semaphores and mutex. OBJECTIVES : To study Semaphores Mutex Producer Consumer Problem



CODE :

#include<stdio.h>

#include<sys/syscall.h>

#include<pthread.h>

#include<semaphore.h>

#include<stdlib.h>

#include<unistd.h>

#include<sys/types.h>

void \*producer(); void \*consumer();

typedef struct {

int buffer[20]; sem\_t full,empty; } shared;

shared sh; int item; int in=0 , out=0; sem\_t mutex;

int main() { pthread\_t ptid1, ptid2, ctid1; sem\_init(&sh.empty,0,20); sem\_init(&sh.full,0,0); sem\_init(&mutex,0,1);

pthread\_create(&ptid1 ,NULL ,producer ,NULL); pthread\_create(&ptid2 ,NULL ,producer ,NULL); pthread\_create(&ctid1 ,NULL ,consumer ,NULL);

pthread\_join(ptid1 , NULL); pthread\_join(ptid2 , NULL); pthread\_join(ctid1 , NULL);

return 0;

}

void \*producer() {

// pthread\_t item; int ptid; while(1) { item=in; sem\_wait(&sh.empty); sem\_wait(&mutex);

sh.buffer[in++]=item;

printf("PRODUCER Thread id = %ld and Producer Item = %d \n",pthread\_self(),item);

sem\_post(&mutex); sem\_post(&sh.full); sleep(3);

}

} void \*consumer() { int ctid; while(out<=19 && in<=19) { sem\_wait(&sh.full); sem\_wait(&mutex);

item=sh.buffer[out++];

printf("\tCONSUMER Thread id = %ld and Consumer Item = %d \n",syscall(SYS\_gettid,ctid),item);

sem\_post(&mutex); sem\_post(&sh.empty); sleep(1);

}

}

OUTPUT :

# PRODUCER Thread id = 140193825486592 and Producer Item = 0

PRODUCER Thread id = 140193817093888 and Producer Item = 1

CONSUMER Thread id = 706 and Consumer Item = 0

CONSUMER Thread id = 706 and Consumer Item = 1

PRODUCER Thread id = 140193817093888 and Producer Item = 2

PRODUCER Thread id = 140193825486592 and Producer Item = 2

CONSUMER Thread id = 706 and Consumer Item = 2

CONSUMER Thread id = 706 and Consumer Item = 2

PRODUCER Thread id = 140193817093888 and Producer Item = 4

PRODUCER Thread id = 140193825486592 and Producer Item = 5

## CONSUMER Thread id = 706 and Consumer Item = 4

CONSUMER Thread id = 706 and Consumer Item = 5

# PRODUCER Thread id = 140193817093888 and Producer Item = 6

CONSUMER Thread id = 706 and Consumer Item = 6

# PRODUCER Thread id = 140193825486592 and Producer Item = 7

## CONSUMER Thread id = 706 and Consumer Item = 7

PRODUCER Thread id = 140193817093888 and Producer Item = 8

## CONSUMER Thread id = 706 and Consumer Item = 8

PRODUCER Thread id = 140193825486592 and Producer Item = 9

CONSUMER Thread id = 706 and Consumer Item = 9

# PRODUCER Thread id = 140193817093888 and Producer Item = 10

CONSUMER Thread id = 706 and Consumer Item = 10

# PRODUCER Thread id = 140193825486592 and Producer Item = 11

CONSUMER Thread id = 706 and Consumer Item = 11 PRODUCER Thread id = 140193817093888 and Producer Item = 12

CONSUMER Thread id = 706 and Consumer Item = 12

PRODUCER Thread id = 140193825486592 and Producer Item = 13

CONSUMER Thread id = 706 and Consumer Item = 13

PRODUCER Thread id = 140193817093888 and Producer Item = 14

CONSUMER Thread id = 706 and Consumer Item = 14

PRODUCER Thread id = 140193825486592 and Producer Item = 15

CONSUMER Thread id = 706 and Consumer Item = 15