Experiment Number: 7

Problem Statement:

**a)  Implementation of Classical problems Producer Consumer using Threads and Semaphore.**

**b)  Implementation of Classical problems Producer Consumer using Threads and Mutex**

NAME: **Manoj Dhanraj Mule** ROLL NO: **71**

CLASS: **IT-B**  BATCH: **B3**

DATE OF PERFORMANCE: **26-09-2024**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **Implementation of Classical problems Producer Consumer using Threads and Semaphore.**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <semaphore.h>

#include <unistd.h>

#define BUFFER\_SIZE 5

int buffer[BUFFER\_SIZE];

int in = 0, out = 0;

sem\_t empty;

sem\_t full;

sem\_t mutex;

void\* producer(void\* arg) {

int item;

int producer\_id = \*((int\*)arg);

while (1) {

item = rand() % 100;

sem\_wait(&empty);

sem\_wait(&mutex);

buffer[in] = item;

printf("Producer %d: Inserted Item %d at %d\n", producer\_id, item, in);

in = (in + 1) % BUFFER\_SIZE;

sem\_post(&mutex);

sem\_post(&full);

sleep(1);

}

return NULL;

}

void\* consumer(void\* arg) {

int item;

int consumer\_id = \*((int\*)arg);

while (1) {

sem\_wait(&full);

sem\_wait(&mutex);

item = buffer[out];

printf("Consumer %d: Removed Item %d from %d\n", consumer\_id, item, out);

out = (out + 1) % BUFFER\_SIZE;

sem\_post(&mutex);

sem\_post(&empty);

sleep(1);

}

return NULL;

}

int main() {

pthread\_t prod\_thread[3], cons\_thread[3];

sem\_init(&empty, 0, BUFFER\_SIZE);

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

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

int producer\_ids[3] = {1, 2, 3};

int consumer\_ids[3] = {1, 2, 3};

for (int i = 0; i < 3; i++) {

pthread\_create(&prod\_thread[i], NULL, producer, &producer\_ids[i]);

pthread\_create(&cons\_thread[i], NULL, consumer, &consumer\_ids[i]);

}

for (int i = 0; i < 3; i++) {

pthread\_join(prod\_thread[i], NULL);

pthread\_join(cons\_thread[i], NULL);

}

sem\_destroy(&empty);

sem\_destroy(&full);

sem\_destroy(&mutex);

return 0;

}

**Output:-**

manoj@manoj-VirtualBox:~/shellscriptprogram$ gcc lab7c.c -o lab7c -pthread

manoj@manoj-VirtualBox:~/shellscriptprogram$ ./lab7c

Producer 3: Inserted Item 83 at 0

Consumer 3: Removed Item 83 from 0

Producer 2: Inserted Item 86 at 1

Consumer 2: Removed Item 86 from 1

Producer 1: Inserted Item 77 at 2

Consumer 1: Removed Item 77 from 2

Producer 3: Inserted Item 15 at 3

Producer 2: Inserted Item 93 at 4

Producer 1: Inserted Item 35 at 0

Consumer 3: Removed Item 15 from 3

Consumer 2: Removed Item 93 from 4

Consumer 1: Removed Item 35 from 0

Producer 3: Inserted Item 86 at 1

Consumer 3: Removed Item 86 from 1

Producer 1: Inserted Item 49 at 2

Consumer 2: Removed Item 49 from 2

Producer 2: Inserted Item 92 at 3

**b)  Implementation of Classical problems Producer Consumder using Threads andMutex.**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <unistd.h>

#define BUFFER\_SIZE 5

int buffer[BUFFER\_SIZE];

int in = 0, out = 0;

int count = 0;

pthread\_mutex\_t mutex;

pthread\_cond\_t cond\_empty;

pthread\_cond\_t cond\_full;

void\* producer(void\* arg) {

int item;

int producer\_id = \*((int\*)arg);

while (1) {

item = rand() % 100;

pthread\_mutex\_lock(&mutex);

while (count == BUFFER\_SIZE) {

pthread\_cond\_wait(&cond\_empty, &mutex);

}

buffer[in] = item;

printf("Producer %d: Inserted Item %d at %d\n", producer\_id, item, in);

in = (in + 1) % BUFFER\_SIZE;

count++;

pthread\_cond\_signal(&cond\_full);

pthread\_mutex\_unlock(&mutex);

sleep(1);

}

return NULL;

}

void\* consumer(void\* arg) {

int item;

int consumer\_id = \*((int\*)arg);

while (1) {

pthread\_mutex\_lock(&mutex);

while (count == 0) {

pthread\_cond\_wait(&cond\_full, &mutex);

}

item = buffer[out];

printf("Consumer %d: Removed Item %d from %d\n", consumer\_id, item, out);

out = (out + 1) % BUFFER\_SIZE;

count--;

pthread\_cond\_signal(&cond\_empty); // Signal that buffer has an empty slot

pthread\_mutex\_unlock(&mutex);

sleep(1);

}

return NULL;

}

int main() {

pthread\_t prod\_thread[3], cons\_thread[3];

pthread\_mutex\_init(&mutex, NULL);

pthread\_cond\_init(&cond\_empty, NULL);

pthread\_cond\_init(&cond\_full, NULL);

int producer\_ids[3] = {1, 2, 3};

int consumer\_ids[3] = {1, 2, 3};

for (int i = 0; i < 3; i++) {

pthread\_create(&prod\_thread[i], NULL, producer, &producer\_ids[i]);

pthread\_create(&cons\_thread[i], NULL, consumer, &consumer\_ids[i]);

for (int i = 0; i < 3; i++) {

pthread\_join(prod\_thread[i], NULL);

pthread\_join(cons\_thread[i], NULL);

}

pthread\_mutex\_destroy(&mutex);

pthread\_cond\_destroy(&cond\_empty);

pthread\_cond\_destroy(&cond\_full);

return 0;

}

**Output:**

manoj@manoj-VirtualBox:~/shellscriptprogram$ gcc lab7d.c -o lab7d -pthread

manoj@manoj-VirtualBox:~/shellscriptprogram$ ./lab7d

Producer 3: Inserted Item 83 at 0

Consumer 3: Removed Item 83 from 0

Producer 2: Inserted Item 86 at 1

Consumer 2: Removed Item 86 from 1

Producer 1: Inserted Item 77 at 2

Consumer 1: Removed Item 77 from 2

Producer 3: Inserted Item 15 at 3

Consumer 3: Removed Item 15 from 3

Producer 2: Inserted Item 93 at 4

Consumer 2: Removed Item 93 from 4

Producer 1: Inserted Item 35 at 0

Consumer 1: Removed Item 35 from 0

Producer 3: Inserted Item 86 at 1

Consumer 3: Removed Item 86 from 1

Producer 2: Inserted Item 92 at 2