Name : Sarthak Pagar

Roll No. : 34

Class : TE (IT)

Practical : 4A

Statement : [Thread synchronization using counting semaphores. Application to demonstrate: producerconsumer problem with counting semaphores and mutex.](https://drive.google.com/file/d/1SKXLYzJBhiUIXql_AKVrRccwrFA6bIUo/view?usp=sharing)

#include <stdio.h>

#include <sys/syscall.h>

#include <pthread.h>

#include <semaphore.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#define BUFFER\_SIZE 20

void \*producer(void \*arg);

void \*consumer(void \*arg);

typedef struct {

int buffer[BUFFER\_SIZE];

sem\_t full, empty;

int in, out; // Indices for the buffer

} shared\_data;

shared\_data sh;

sem\_t mutex;

int main() {

pthread\_t ptid1, ptid2, ctid1;

sh.in = 0;

sh.out = 0;

sem\_init(&sh.empty, 0, BUFFER\_SIZE); // Initially, all slots are empty

sem\_init(&sh.full, 0, 0); // Initially, no slots are filled

sem\_init(&mutex, 0, 1); // Mutex for critical section

// Create producer and consumer threads

pthread\_create(&ptid1, NULL, producer, NULL);

pthread\_create(&ptid2, NULL, producer, NULL);

pthread\_create(&ctid1, NULL, consumer, NULL);

// Wait for threads to finish

pthread\_join(ptid1, NULL);

pthread\_join(ptid2, NULL);

pthread\_join(ctid1, NULL);

return 0;

}

void \*producer(void \*arg) {

int item;

while (1) {

item = rand() % 100; // Produce a random item between 0 and 99

sem\_wait(&sh.empty); // Wait for an empty slot

sem\_wait(&mutex); // Lock the critical section

// Insert item into buffer

sh.buffer[sh.in] = item;

printf("PRODUCER Thread id = %ld produced Item = %d at index %d\n", pthread\_self(), item, sh.in);

sh.in = (sh.in + 1) % BUFFER\_SIZE; // Wrap around the index

sem\_post(&mutex); // Unlock the critical section

sem\_post(&sh.full); // Signal that a new item has been produced

sleep(1); // Simulate time taken to produce an item

}

}

void \*consumer(void \*arg) {

int item;

while (1) {

sem\_wait(&sh.full); // Wait for a filled slot

sem\_wait(&mutex); // Lock the critical section

// Remove item from buffer

item = sh.buffer[sh.out];

printf("\tCONSUMER Thread id = %ld consumed Item = %d from index %d\n", pthread\_self(), item, sh.out);

sh.out = (sh.out + 1) % BUFFER\_SIZE; // Wrap around the index

sem\_post(&mutex); // Unlock the critical section

sem\_post(&sh.empty); // Signal that an item has been consumed

sleep(2); // Simulate time taken to consume an item

}

}

Output :-

sarthak1594@UbuntuInWin:~/OS Practical/4$ gcc producer-consumer.c

sarthak1594@UbuntuInWin:~/OS Practical/4$ ./a.out

PRODUCER Thread id = 129509741299264 produced Item = 83 at index 0

CONSUMER Thread id = 129509720327744 consumed Item = 83 from index 0

PRODUCER Thread id = 129509730813504 produced Item = 86 at index 1

PRODUCER Thread id = 129509741299264 produced Item = 77 at index 2

PRODUCER Thread id = 129509730813504 produced Item = 15 at index 3

CONSUMER Thread id = 129509720327744 consumed Item = 86 from index 1

PRODUCER Thread id = 129509730813504 produced Item = 35 at index 4

PRODUCER Thread id = 129509741299264 produced Item = 93 at index 5

PRODUCER Thread id = 129509730813504 produced Item = 86 at index 6

PRODUCER Thread id = 129509741299264 produced Item = 92 at index 7

PRODUCER Thread id = 129509741299264 produced Item = 49 at index 8

CONSUMER Thread id = 129509720327744 consumed Item = 77 from index 2

PRODUCER Thread id = 129509730813504 produced Item = 21 at index 9

PRODUCER Thread id = 129509741299264 produced Item = 62 at index 10

PRODUCER Thread id = 129509730813504 produced Item = 27 at index 11

CONSUMER Thread id = 129509720327744 consumed Item = 15 from index 3

PRODUCER Thread id = 129509741299264 produced Item = 90 at index 12

PRODUCER Thread id = 129509730813504 produced Item = 59 at index 13

PRODUCER Thread id = 129509741299264 produced Item = 63 at index 14

PRODUCER Thread id = 129509730813504 produced Item = 26 at index 15

CONSUMER Thread id = 129509720327744 consumed Item = 35 from index 4

PRODUCER Thread id = 129509741299264 produced Item = 40 at index 16

PRODUCER Thread id = 129509730813504 produced Item = 26 at index 17

PRODUCER Thread id = 129509741299264 produced Item = 72 at index 18

PRODUCER Thread id = 129509730813504 produced Item = 36 at index 19

CONSUMER Thread id = 129509720327744 consumed Item = 93 from index 5

PRODUCER Thread id = 129509741299264 produced Item = 11 at index 0

PRODUCER Thread id = 129509730813504 produced Item = 68 at index 1

PRODUCER Thread id = 129509741299264 produced Item = 67 at index 2

PRODUCER Thread id = 129509730813504 produced Item = 29 at index 3

CONSUMER Thread id = 129509720327744 consumed Item = 86 from index 6

PRODUCER Thread id = 129509741299264 produced Item = 82 at index 4

PRODUCER Thread id = 129509730813504 produced Item = 30 at index 5

PRODUCER Thread id = 129509741299264 produced Item = 62 at index 6

CONSUMER Thread id = 129509720327744 consumed Item = 92 from index 7

PRODUCER Thread id = 129509730813504 produced Item = 23 at index 7

CONSUMER Thread id = 129509720327744 consumed Item = 49 from index 8

PRODUCER Thread id = 129509741299264 produced Item = 67 at index 8

CONSUMER Thread id = 129509720327744 consumed Item = 21 from index 9

PRODUCER Thread id = 129509730813504 produced Item = 35 at index 9

CONSUMER Thread id = 129509720327744 consumed Item = 62 from index 10

PRODUCER Thread id = 129509741299264 produced Item = 29 at index 10

CONSUMER Thread id = 129509720327744 consumed Item = 27 from index 11

PRODUCER Thread id = 129509730813504 produced Item = 2 at index 11

CONSUMER Thread id = 129509720327744 consumed Item = 90 from index 12

PRODUCER Thread id = 129509741299264 produced Item = 22 at index 12

CONSUMER Thread id = 129509720327744 consumed Item = 59 from index 13

PRODUCER Thread id = 129509730813504 produced Item = 58 at index 13

CONSUMER Thread id = 129509720327744 consumed Item = 63 from index 14

PRODUCER Thread id = 129509741299264 produced Item = 69 at index 14

CONSUMER Thread id = 129509720327744 consumed Item = 26 from index 15

PRODUCER Thread id = 129509730813504 produced Item = 67 at index 15

CONSUMER Thread id = 129509720327744 consumed Item = 40 from index 16

PRODUCER Thread id = 129509741299264 produced Item = 93 at index 16

CONSUMER Thread id = 129509720327744 consumed Item = 26 from index 17

PRODUCER Thread id = 129509730813504 produced Item = 56 at index 17

CONSUMER Thread id = 129509720327744 consumed Item = 72 from index 18

PRODUCER Thread id = 129509741299264 produced Item = 11 at index 18

CONSUMER Thread id = 129509720327744 consumed Item = 36 from index 19

PRODUCER Thread id = 129509730813504 produced Item = 42 at index 19

CONSUMER Thread id = 129509720327744 consumed Item = 11 from index 0

PRODUCER Thread id = 129509741299264 produced Item = 29 at index 0