# Bansilal Ramnath Agarwal Charitable Trust’s

Vishwakarma Institute of Technology, Pune-37

*(Anautonomous Institute of Savitribai Phule Pune University)*



**Department of Multidisciplinary Engineering**

|  |  |
| --- | --- |
| **Division** | **CS-A** |
| **Batch** | **B1** |
| **Roll no.** | **90** |
| **Name** | **Aditya Shrinivas Kurapati** |
| **Subject** | **OS** |

1. Producer Consumer PROBLEM USING C PROGRAM

#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#define BUFFER\_SIZE 5

typedef struct {

int buf[BUFFER\_SIZE];

size\_t len;

pthread\_mutex\_t mutex;

pthread\_cond\_t can\_produce;

pthread\_cond\_t can\_consume;

} buffer\_t;

void \*producer(void \*arg) {

buffer\_t \*buffer = (buffer\_t \*)arg;

while (1) {

#ifdef UNDERFLOW

sleep(5);

#endif

pthread\_mutex\_lock(&buffer->mutex);

if (buffer->len == BUFFER\_SIZE) {

pthread\_cond\_wait(&buffer->can\_produce, &buffer->mutex);

}

int t = rand();

printf("Produced %d\n ", t);

buffer->buf[buffer->len] = t;

++buffer->len;

pthread\_cond\_signal(&buffer->can\_consume);

pthread\_mutex\_unlock(&buffer->mutex);

printf("\nFULL: %ld\n", buffer->len);

printf("\nEmpty: %ld\n", (BUFFER\_SIZE - buffer->len));

}

return NULL;

}

void \*consumer(void \*arg) {

buffer\_t \*buffer = (buffer\_t \*)arg;

while (1) {

#ifdef OVERFLOW

sleep(5);

#endif

pthread\_mutex\_lock(&buffer->mutex);

if (buffer->len == 0) {

pthread\_cond\_wait(&buffer->can\_consume, &buffer->mutex);

}

--buffer->len;

printf("Consumed %d\n", buffer->buf[buffer->len]);

pthread\_cond\_signal(&buffer->can\_produce);

pthread\_mutex\_unlock(&buffer->mutex);

printf("\nFULL: %ld\n", buffer->len);

printf("\nEmpty: %ld\n", (BUFFER\_SIZE - buffer->len));

}

return NULL;

}

int main(int argc, char \*argv[]) {

buffer\_t buffer = {.len = 0,

.mutex = PTHREAD\_MUTEX\_INITIALIZER,

.can\_produce = PTHREAD\_COND\_INITIALIZER,

.can\_consume = PTHREAD\_COND\_INITIALIZER};

pthread\_t prod, cons;

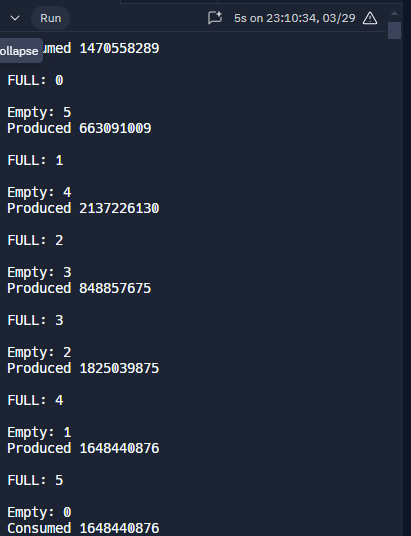
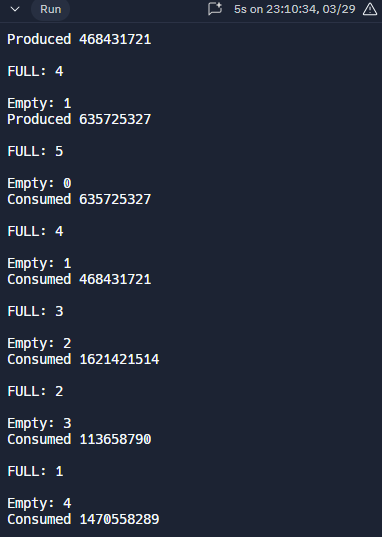
pthread\_create(&prod, NULL, producer, (void \*)&buffer);

pthread\_create(&cons, NULL, consumer, (void \*)&buffer);

pthread\_join(prod, NULL);

pthread\_join(cons, NULL);

return 0;

}  
  
OUTPUT:-  
  
  
  
  
2. READER\_WRITER PROBLEM USING C PROGRAM

#include <pthread.h>

#include <semaphore.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

sem\_t mutex, writeblock;

int data = 0, rcount = 0;

void \*reader(void \*arg) {

int f;

f = \*((int \*)arg);

sem\_wait(&mutex);

rcount++;

if (rcount == 1) {

sem\_wait(&writeblock);

}

sem\_post(&mutex);

printf("Reader %d is reading data %d\n", f, data);

sleep(2);

sem\_wait(&mutex);

rcount--;

if (rcount == 0) {

sem\_post(&writeblock);

}

sem\_post(&mutex);

return NULL;

}

void \*writer(void \*arg) {

int f;

f = \*((int \*)arg);

sem\_wait(&writeblock);

data++;

printf("Data Written By The Writer Is %d\n", data);

sleep(1);

sem\_post(&writeblock);

return NULL;

}

int main() {

int i;

pthread\_t rtid[3], wtid[3];

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

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

while (1) {

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

pthread\_create(&wtid[i], NULL, writer, &i);

pthread\_create(&rtid[i], NULL, reader, &i);

}

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

pthread\_join(wtid[i], NULL);

pthread\_join(rtid[i], NULL);

}

sleep(5); // Delay for 5 seconds before running the loop again

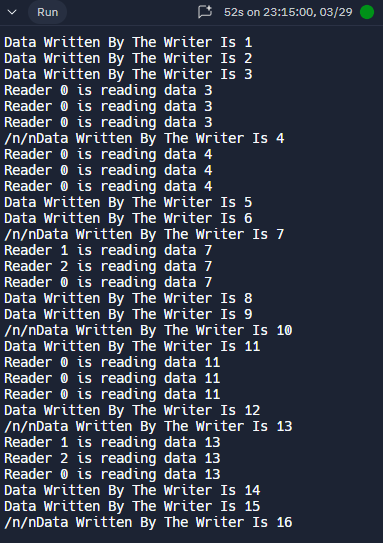
printf("/n/n");

}

return 0;

}

OUTPUT:-



**3.Dinning Philosophers Problem**

#include <pthread.h>

#include <semaphore.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#define MIN(a, b) (a < b ? a : b)

#define MAX(a, b) (a > b ? a : b)

sem\_t chopsticks[5];

void \*eat(void \*arg) {

int i = (int)arg;

int count = 2;

while (count > 0) {

sem\_wait(&chopsticks[MIN(i, (i + 1) % 5)]);

sem\_wait(&chopsticks[MAX(i, (i + 1) % 5)]);

printf("Philosopher %d is eating\n", i);

sem\_post(&chopsticks[MIN(i, (i + 1) % 5)]);

sem\_post(&chopsticks[MAX(i, (i + 1) % 5)]);

count--;

}

return NULL;

}

int main() {

pthread\_t philosophers[5];

int i;

for (i = 0; i < 5; i++) {

sem\_init(&chopsticks[i], 0, 1);

}

for (i = 0; i < 5; i++) {

pthread\_create(&philosophers[i], NULL, eat, (void \*)i);

}

for (i = 0; i < 5; i++) {

pthread\_join(philosophers[i], NULL);

}

return 0;

}

