// Implementation of classical problems ( producer, consumer) using threads & mutex

#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#define BUFFER\_SIZE 5

int buffer[BUFFER\_SIZE];

int count = 0;

pthread\_mutex\_t mutex;

pthread\_cond\_t not\_full, not\_empty;

void\* producer(void\* arg) {

int item;

for (int i = 0; i < 10; i++) {

item = rand() % 100;

pthread\_mutex\_lock(&mutex);

while (count == BUFFER\_SIZE)

pthread\_cond\_wait(&not\_full, &mutex);

buffer[count++] = item;

printf("Producer produced: %d\n", item);

pthread\_cond\_signal(&not\_empty);

pthread\_mutex\_unlock(&mutex);

sleep(1);

}

return NULL;

}

void\* consumer(void\* arg) {

int item;

for (int i = 0; i < 10; i++) {

pthread\_mutex\_lock(&mutex);

while (count == 0)

pthread\_cond\_wait(&not\_empty, &mutex);

item = buffer[--count];

printf("Consumer consumed: %d\n", item);

pthread\_cond\_signal(&not\_full);

pthread\_mutex\_unlock(&mutex);

sleep(1);

}

return NULL;

}

int main() {

pthread\_t prod\_thread, cons\_thread;

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

pthread\_cond\_init(&not\_full, NULL);

pthread\_cond\_init(&not\_empty, NULL);

pthread\_create(&prod\_thread, NULL, producer, NULL);

pthread\_create(&cons\_thread, NULL, consumer, NULL);

pthread\_join(prod\_thread, NULL);

pthread\_join(cons\_thread, NULL);

pthread\_mutex\_destroy(&mutex);

pthread\_cond\_destroy(&not\_full);

pthread\_cond\_destroy(&not\_empty);

return 0;

}