НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Департамент программной инженерии
Дисциплина: «Архитектура вычислительных систем»

Микропроект по дисциплине

«Архитектура вычислительных систем»

На тему:

«Задача о магазине - 2 (забывчивые покупатели)»

Пояснительная записка

Выполнил:
Моторкин Владимир,
студент гр. БПИ198.

Москва
2020

Содержание

1. Текст задания

Вариант 21.

Задача о магазине - 2 (забывчивые покупатели). В магазине работают два отдела, каждый отдел обладает уникальным ассортиментом. В каждом отделе работает один продавец. В магазин ходят исключительно забывчивые покупатели, поэтому каждый покупатель носит с собой список товаров, которые желает купить. Покупатель приобретает товары точно в том порядке, в каком они записаны в его списке. Продавец может обслужить только одного покупателя за раз. Покупатель, вставший в очередь, засыпает пока не дойдет до продавца. Продавец засыпает, если в его отделе нет покупателей, и просыпается, если появится хотя бы один. Создать многопоточное приложение, моделирующее работу магазина.

2. Применяемые расчетные методы

2.1. Теория решения задания

В данном задании генерация списка товаров зависит от количества покупателей и максимально возможного количества товаров в списке. Общее количество товаров равно (количество покупателей) * (максимальная длина списка покупок). При этом в первый магазин попадает (количество покупателей) / 2 товаров (деление нацело), во второй остальные.

Используются коллекции разных потоков: 2 магазина и покупатели, их количество вводится с клавиатуры, и оно не менее двух. Для сообщения о том, что покупатель попал в магазин и для оповещения покупателя о завершении покупки используются семафоры, по одному на магазин и покупателя.

Каждый магазин обслуживает покупателя 2 секунды.

3. Тестирование программы

3.1. Корректные значения

```
Input count of the customers >=2:
2
Input max count of products in the list:
3
list of products:
        product #0 shop #0
        product #1 shop #0
        product #2 shop #0
        product #3 shop #1
        product #4 shop #1
        product #5 shop #1

Customer #0: my shopping list is 4 1
Customer #1: my shopping list is 4 3
Customer #0: I have enqueued into the shop #0
Customer #1: I have enqueued into the shop #1
The Shop #0 is serving the customer #0
The Shop #1 is serving the customer #1
The Shop #0 has served the customer #0
The Shop #1 has served the customer #1
Customer #1: I have bought product #3 in shop #1
Customer #1: I have enqueued into the shop #1
Customer #0: I have bought product #1 in shop #0
Customer #0: I have enqueued into the shop #1
The Shop #1 is serving the customer #1
The Shop #1 has served the customer #1
The Shop #1 is serving the customer #0
Customer #1: I have bought product #4 in shop #1
Customer #1: I have bought everything I wanted!
Served customers: 1
The Shop #1 has served the customer #0
Customer #0: I have bought product #4 in shop #1
Customer #0: I have bought everything I wanted!
Served customers: 2
```

```
Input count of the customers >=2:
2
Input max count of products in the list:
1
list of products:
        product #0 shop #0
        product #1 shop #1

Customer #0: my shopping list is 1
Customer #1: my shopping list is 1
Customer #0: I have enqueued into the shop #1
Customer #1: I have enqueued into the shop #1
The Shop #1 is serving the customer #0
The Shop #1 has served the customer #0
The Shop #1 is serving the customer #1
Customer #0: I have bought product #1 in shop #1
Customer #0: I have bought everything I wanted!
Served customers: 1
The Shop #1 has served the customer #1
Customer #1: I have bought product #1 in shop #1
Customer #1: I have bought everything I wanted!
Served customers: 2
```

```
Input count of the customers >=2:
4
Input max count of products in the list:
1
list of products:
        product #0 shop #0
        product #1 shop #0
        product #2 shop #1
        product #3 shop #1

Customer #0: my shopping list is 1
Customer #0: I have enqueued into the shop #0
Customer #2: my shopping list is 3
Customer #2: I have enqueued into the shop #1
Customer #1: my shopping list is 2
Customer #1: I have enqueued into the shop #1
Customer #3: my shopping list is 1
Customer #3: I have enqueued into the shop #0
The Shop #0 is serving the customer #0
The Shop #1 is serving the customer #2
The Shop #0 has served the customer #0
The Shop #0 is serving the customer #3
The Shop #1 has served the customer #2
Customer #0: I have bought product #1 in shop #0
Customer #0: I have bought everything I wanted!
Served customers: 1
Customer #2: I have bought product #3 in shop #1
Customer #2: I have bought everything I wanted!
Served customers: 2
The Shop #1 is serving the customer #1
The Shop #1 has served the customer #1
Customer #3: I have bought product #1 in shop #0
Customer #3: I have bought everything I wanted!
Served customers: 3
The Shop #0 has served the customer #3
Customer #1: I have bought product #2 in shop #1
Customer #1: I have bought everything I wanted!
Served customers: 4
```

3.2. Некорректные значения

```
Input count of the customers >=2:
2
Input max count of products in the list:
1
list of products:
        product #0 shop #0
        product #1 shop #1

Customer #0: my shopping list is 1
Customer #1: my shopping list is 1
Customer #0: I have enqueued into the shop #1
Customer #1: I have enqueued into the shop #1
The Shop #1 is serving the customer #0
The Shop #1 has served the customer #0
The Shop #1 is serving the customer #1
Customer #0: I have bought product #1 in shop #1
Customer #0: I have bought everything I wanted!
Served customers: 1
The Shop #1 has served the customer #1
Customer #1: I have bought product #1 in shop #1
Customer #1: I have bought everything I wanted!
Served customers: 2
```

ПРИЛОЖЕНИЕ 1

Список литературы

1. Задание. [Электронный ресурс] // URL: http://softcraft.ru/edu/comparch/tasks/mp02/mp02.pdf / (дата обращения: 10.12.2020)
2. Описание потоков в c++. [Электронный ресурс] // URL: https://en.cppreference.com/w/cpp/thread (дата обращения: 10.12.2020)
3. Потоки, блокировки и условные переменные в C++11. [Электронный ресурс] // URL: https://habr.com/ru/post/182610/ (дата обращения: 10.12.2020)

ПРИЛОЖЕНИЕ 2

Код программы

```
1.  #include <iostream>
2.  #include <vector>
3.  #include <queue>
4.  #include <mutex>
5.  #include <semaphore.h>
6.  #include <thread>
7.
8.  using namespace std;
9.
10.struct Product{
11.    int id;
12.    int shop;
13.};
14.
15.vector<queue<int>> queues(2);
16.vector<Product> allProducts;
17.
18.int servedCustomers = 0;
19.int globalId = 0;
20.int globalIdShop = 0;
21.
22.// Generating products with different id.
23.void generateProducts(int countOfCustomers, int maxCountOfProducts){
24.    int id;
25.    for (int i = 0; i < countOfCustomers / 2; ++i) {
26.        for (int j = 0; j < maxCountOfProducts; ++j) {
```

```
27.             Product product;
28.             product.id = id;
29.             product.shop = 0;
30.             allProducts.push_back(product);
31.             id++;
32.         }
33.     }
34.     for (int i = countOfCustomers / 2; i < countOfCustomers; ++i) {
35.         for (int j = 0; j < maxCountOfProducts; ++j) {
36.             Product product;
37.             product.id = id;
38.             product.shop = 1;
39.             allProducts.push_back(product);
40.             id++;
41.         }
42.     }
43.     cout << "list of products:" << endl;
44.     for (auto& product : allProducts)
45.         cout << "\tproduct #" << product.id << " shop #" << product.shop << endl;
46.     cout << endl;
47. }
48.
49.
50. void customerFunction(vector<sem_t> *shop_sems, int maxCountOfProducts,
    sem_t *customer_semaphore,
51.                     mutex *generation_mutex, mutex *queue_mutex, mutex
    *console_mutex, mutex *served_mutex){
52.     vector<Product> shoppingList;
53.     generation_mutex->lock();
```

```
54.    int myId = globalId++;
55.    srand(static_cast<unsigned>(myId * myId +
       static_cast<unsigned>(time(nullptr))));
56.    int countOfProducts = rand() % maxCountOfProducts + 1;
57.    shoppingList.reserve(countOfProducts);
58.    for (int i = 0; i < countOfProducts; ++i) {
59.        shoppingList.push_back(allProducts.at(rand() % allProducts.size()));
60.    }
61.    generation_mutex->unlock();
62.    console_mutex->lock();
63.    cout << "Customer #" << myId << ": my shopping list is ";
64.    for (int i = 0; i < countOfProducts; ++i) {
65.        cout << shoppingList[i].id << " ";
66.    }
67.    cout << endl;
68.    console_mutex->unlock();
69.
70.    while (!shoppingList.empty()){
71.        queue_mutex->lock();
72.        queues[shoppingList.back().shop].push(myId);
73.
74.        console_mutex->lock();
75.        cout << "Customer #" << myId << ": I have enqueued into the shop #" <<
       shoppingList.back().shop << endl;
76.        console_mutex->unlock();
77.
78.        queue_mutex->unlock();
79.
80.        sem_post(&shop_sems->at(shoppingList.back().shop));
```

```
81.         sem_wait(customer_semaphore);
82.
83.         console_mutex->lock();
84.         cout << "Customer #" << myId << ": I have bought product #" <<
    shoppingList.back().id <<
85.              " in shop #" << shoppingList.back().shop << endl;
86.         console_mutex->unlock();
87.
88.         shoppingList.pop_back();
89.     }
90.     served_mutex->lock();
91.     servedCustomers++;
92.     served_mutex->unlock();
93.
94.     console_mutex->lock();
95.     cout << "Customer #" << myId << ": I have bought everything I wanted!" <<
    endl;
96.     cout << "Served customers: " << servedCustomers << endl;
97.     console_mutex->unlock();
98.
99.     return;
100.    }
101.
102.    void shopFunction(vector<sem_t> *shop_sems, int countOfCustomers,
    vector<sem_t> *customer_sems, mutex *vector_mutex,
103.                      mutex *console_mutex, mutex *gen_shop_mutex, mutex
    *served_mutex){
104.
105.        gen_shop_mutex->lock();
```

```
106.        int myid = globalIdShop++;
107.        srand(static_cast<unsigned>(myid * myid +
    static_cast<unsigned>(time(nullptr))));
108.        gen_shop_mutex->unlock();
109.
110.        int myServedCustomers = 0;
111.        int customerId;
112.
113.        while(myServedCustomers != countOfCustomers){
114.            sem_wait(&shop_sems->at(myid));
115.
116.            vector_mutex->lock();
117.            customerId = queues[myid].front();
118.            queues[myid].pop();
119.            vector_mutex->unlock();
120.
121.            console_mutex->lock();
122.            cout << "The Shop #" << myid
123.                << " is serving the customer #" << customerId << endl;
124.            console_mutex->unlock();
125.
126.            this_thread::sleep_for(chrono::seconds(2));
127.
128.            console_mutex->lock();
129.            cout << "The Shop #" << myid
130.                << " has served the customer #" << customerId << endl;
131.            console_mutex->unlock();
132.
```

```
133.            sem_post(&customer_sems->at(customerId));
134.
135.            served_mutex->lock();
136.            myServedCustomers = servedCustomers;
137.            served_mutex->unlock();
138.        }
139.        return;
140.    }
141.
142.    int main() {
143.        int countOfCustomers;
144.        int maxCountOfProducts;
145.        mutex queue_mutex, served_mutex, vector_mutex, console_mutex,
    generation_mutex, gen_shop_mutex;
146.        // Input
147.        cout << "Input count of the customers >=2:" << endl;
148.        cin >> countOfCustomers;
149.        if (countOfCustomers < 2){
150.            cout << "count must be >= 2!";
151.            return -1;
152.        }
153.        cout << "Input max count of products in the list:" << endl;
154.        cin >> maxCountOfProducts;
155.        // Customer semaphores.
156.        vector<sem_t> customer_semaphores(countOfCustomers);
157.        generateProducts(countOfCustomers, maxCountOfProducts);
158.        for (uint64_t i = 0; i < countOfCustomers; ++i) {
159.            sem_init(&customer_semaphores[i], 0, 0);
```

```
160.    }
161.    pthread_barrier_t customer_barrier;
162.    pthread_barrier_init(&customer_barrier, nullptr, countOfCustomers + 2);
163.    // Shop semaphores.
164.    vector<sem_t> shop_semaphores(2);
165.    for (auto &sem : shop_semaphores) {
166.        sem_init(&sem, 0, 0);
167.    }
168.
169.    vector<thread> customer_threads;
170.    for (uint64_t i = 0; i < countOfCustomers; ++i) {
171.        customer_threads.emplace_back(customerFunction, &shop_semaphores,
    maxCountOfProducts, &customer_semaphores[i],
172.                                      &generation_mutex, &queue_mutex, &console_mutex,
    &served_mutex);
173.    }
174.
175.    vector<std::thread> shop_threads;
176.    for (int i = 0; i < 2; ++i) {
177.        shop_threads.emplace_back(shopFunction, &shop_semaphores,
    countOfCustomers, &customer_semaphores, &vector_mutex,
178.                                  &console_mutex, &gen_shop_mutex, &served_mutex);
179.    }
180.
181.    for (uint64_t i = 0; i < 2; ++i) {
182.        shop_threads[i].detach();
183.    }
184.    for (uint64_t i = 0; i < countOfCustomers; ++i) {
185.        customer_threads[i].join();
```

```
186.        }
187.
188.        system ("pause");
189.        return 0;
    }
```