НИУ ВШЭ, Департамент программной инженерии

Микропроект №2

Вариант 11

Карякин Алексей

БПИ192

1. Текст задачи

Задача о магазине. В магазине работают три отдела, каждый отдел обслуживает один продавец. Покупатель, зайдя в магазин, делает покупки в произвольных отделах, и если в выбранном отделе продавец не свободен, покупатель становится в очередь и засыпает, пока продавец не освободится. Создать многопоточное приложение, моделирующее рабочий день магазина.

2. Текст программы

```
#include <pthread.h>
#include <semaphore.h>
#include <stdlib.h>
#include <stdio.h>
#include <chrono>
#include <iostream>
#include <thread>
#include <queue>

pthread_mutex_t mutex[3]; //отдельные мьютексы для каждого отдела

pthread_cond_t empty[3]; //потоки-продавцы блокируются этой переменной, когда в очереди к продавцу никого нет (отдельная переменная для каждого отдела).
pthread_cond_t not_empty[3]; //потоки-покупатели блокируются этой переменной, когда к интересующему продавцу выстроилась очередь (отдельная переменная для каждого отдела).

std::queue<int> firstDep; //очередь для первого отдела.
std::queue<int> secondDep; //очередь для второго отдела.
std::queue<int> thirdDep; //очередь для третьего отдела.

int values[3] = { 0, 0, 0 }; //массив для хранения последних значений сумм товаров.

//стартовая функция потоков – продавцов
void* Salesman(void* param) {
    srand(time(NULL)); //инициализатор генератора случайных чисел, зависящий от текущего времени.
    int pNum = *((int*)param); //номер продавца
    int price; //цена, за которую покупатель купил товар
    while (1) {
        pthread_mutex_lock(&mutex[pNum - 1]); //защита операции чтения

        //заснуть, если в очереди нет покупателей.
        while (pNum == 1 && firstDep.size() == 0 || pNum == 2 && secondDep.size() == 0 || pNum == 3 && thirdDep.size() == 0) {
            pthread_cond_wait(&empty[pNum - 1], &mutex[pNum - 1]);
        }

        price = values[pNum - 1]; //получение значения цены товара.

        printf("Department %d got %d from customer\n", pNum, price);

```

```
40.         //удаление покупателя из очереди после покупки.
41.         switch (pNum)
42.         {
43.         case 1:
44.             firstDep.pop();
45.             break;
46.         case 2:
47.             secondDep.pop();
48.             break;
49.         case 3:
50.             thirdDep.pop();
51.             break;
52.         default:
53.             break;
54.         }
55.
56.         //конец критической секции
57.         pthread_mutex_unlock(&mutex[pNum - 1]);
58.         //разбудить потоки-покупатели после продажи товара клиенту.
59.         pthread_cond_broadcast(&not_empty[pNum - 1]);
60.
61.         std::this_thread::sleep_for(std::chrono::milliseconds(1000)); //заснуть
    текущему потоку на секунду.
62.     }
63.     return NULL;
64. }
65.
66. //стартовая функция потоков – покупателей.
67. void* Customer(void* param) {
68.     srand(time(NULL)); //инициализатор генератора случайных чисел, зависящий от
    текущего времени.
69.     int cNum = *((int*)param); //номер покупателя.
70.
71.     int price; //цена товара
72.     int depNum; //номер отдела, в который хочет пойти покупатель.
73.     while (1) {
74.         depNum = rand() % 3;
75.
76.         price = rand() % 1000;
77.
78.         pthread_mutex_lock(&mutex[depNum]); //защита операции записи.
79.
80.         printf("Consumer %d decided to go to %d department\n", cNum, depNum + 1);
81.
82.         //добавление покупателя в очередь.
83.         switch (depNum)
84.         {
85.         case 0:
86.             firstDep.push(cNum);
87.             break;
88.         case 1:
89.             secondDep.push(cNum);
90.             break;
91.         case 2:
92.             thirdDep.push(cNum);
93.             break;
94.         default:
95.             break;
96.         }
97.
98.         //Покупатель спит, пока не окажется первым в очереди.
99.         while (depNum == 0 && firstDep.front() != cNum || depNum == 1 &&
    secondDep.front() != cNum || depNum == 2 && thirdDep.front() != cNum) {
100.                pthread_cond_wait(&not_empty[depNum], &mutex[depNum]);
101.            }
```

```
102.
103.                 values[depNum] = price; //запись цены товара.
104.
105.                 printf("Consumer %d bought something with price %d in %d
    department\n", cNum, price, depNum + 1);
106.
107.                 //конец критической секции.
108.                 pthread_mutex_unlock(&mutex[depNum]);
109.                 //разбудить потоки-продавцы.
110.                 pthread_cond_broadcast(&empty[depNum]);
111.
112.                 std::this_thread::sleep_for(std::chrono::milliseconds(1000));
    //заснуть текущему потоку на секунду.
113.             }
114.             return NULL;
115.         }
116.
117.         int main() {
118.             int i;
119.
120.             //инициализация мьютексов и усл. переменных.
121.             for (i = 0; i < 3; i++) {
122.                 pthread_mutex_init(&mutex[i], NULL);
123.                 pthread_cond_init(&empty[i], NULL);
124.                 pthread_cond_init(&not_empty[i], NULL);
125.             }
126.
127.
128.             pthread_t threadS[3];
129.             int salesmans[3];
130.             for (i = 0; i < 3; i++) {
131.                 salesmans[i] = i + 1;
132.                 pthread_create(&threadS[i], NULL, Salesman, (void*)(salesmans +
    i));
133.             }
134.
135.             //запуск потребителей
136.             pthread_t threadC[4];
137.             int customers[4];
138.             for (i = 0; i < 4; i++) {
139.                 customers[i] = i + 1;
140.                 pthread_create(&threadC[i], NULL, Customer, (void*)(customers +
    i));
141.             }
142.
143.             //Мэйн становится потоком-покупателем.
144.             int mNum = 0;
145.             Customer((void*)&mNum);
146.             return 0;
147.     }
```

3. Описание работы программы

Потоки-покупатели каждую итерацию цикла while(1) генерируют номер отдела, в который он пойдёт покупать товар, цену которого он также генерирует. Затем, с помощью мьютекса (отдельный для каждого отдела)

защищаем операции записи/добавления в очередь, и, в случае если покупатель не первый в очереди, отправляем его спать. Когда покупатель оказывается первым, записываем в массив с последними значениями товаров цену приобретённого покупателем товара. После этого критическая секция заканчивается, потоки покупатели будятся.

Потоки-продавцы каждую итерацию цикла, начинают с критической секции, которая защищает операцию чтения. Продавцы засыпают, если в очереди никого нет. Когда в очереди появляется человек, продавец считывает цену товара, и после этого обслуженный покупатель удаляется из очереди, и затем критическая секция заканчивается, потоки-покупатели будятся.

4. Скриншоты работы программы