ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Департамент программной инженерии

Отчет по микропроекту №2 по дисциплине "Архитектура вычислительных систем"

Пояснительная Записка

Исполнитель:
Студент группы БПИ195(1)
Ни Эдуард
13.12.2020 г.

Москва 2020

Условие задания

ВАРИАНТ1. *Задача о парикмахере.* В тихом городке есть парикмахерская. Салон парикмахерской мал, ходить там может только парикмахер и один посетитель. Парикмахер всю жизнь обслуживает посетителей. Когда в салоне никого нет, он спит в кресле. Когда посетитель приходит и видит спящего парикмахера, он будет его, садится в кресло и спит, пока парикмахер занят стрижкой. Если посетитель приходит, а парикмахер занят, то он встает в очередь и засыпает. После стрижки парикмахер сам провожает посетителя. Если есть ожидающие посетители, то парикмахер будит одного из них и ждет пока тот сядет в кресло парикмахера и начинает стрижку. Если никого нет, он снова садится в свое кресло и засыпает до прихода посетителя. Создать многопоточное приложение, моделирующее рабочий день парикмахерской

Решение

Для решения задачи будем пользоваться мутексами для синхронизации работы потоков, у которых есть доступ к общему ресурсу.

По условию, парикмахер спит, когда нет клиентов, клиент приходит и будит его. Когда парикмахер постриг посетителя, он его провожает до выхода, будит следующего и стрижет. В решении это регулируется с помощью рандома, который выставляет время ожидания для задачи – 0 или 4 секунды.

Составление программы

Для работы программы будем использовать библиотеки для языка C++: iostream, unistd.h, pthread.h.

Наша программа будет работать, пока не обслужат 10 посетителей.

Москва 2020

Описание программы

1. Глобальные переменные.

```
pthread_mutex_t custom_m;  //Мутекс для блокировки очереди.
pthread_mutex_t cut_m;  //Мутекс для блокировки стрижки.

int customersCnt = 10;   //Количество посетителей.
int cameCustomers = 0;  //Пришедшие посетители - количество людей в очереди.
int waitOrNot = 0;

bool isExit = true; //Проводил ли парикмахер посетителя.
bool isWait = false;    //Ожидает ли кто-то из пришедших.
bool isCutted = false;  //Пострижен или нет.
```

Рисунок 1 – Глобальные переменные

Создаем два мутекса для блокировки очереди и стрижки. Переменная customersCnt – количество посетителей, которых нужно постричь. cameCustomers – посетители в очереди. waitOrNot – для решения задачи, будет ли образовываться очередь или нет. isExit – ушел ли посетитель из парикмахерской, isWait – есть ли кто, isCutted – постригли или нет.

2. Метод Hairdresser.

```
void *Hairdresser(void *num) {
    const char *pName = (*((std::string *) num)).c_str();
    pthread_mutex_unlock(&cut_m);    //Открываем мутекс.
    //Пока есть запланированные посетители.
    while (customersCnt > 0) {
        //Если парикмахер проводил посетителя и некого ждать.
        while (isExit && !isWait && cameCustomers <= 0) {
            printf( format: "Hairdresser %s: Zzzz... Waiting for a Customer..\n", pName);
            isExit = false;
            while (!isWait);
        }
        //Обслуживаем всю очередь.
        while (cameCustomers > 0) {
            isCutted = true;
            printf( format: "Hairdresser %s: Some Customer came. I am making a haircut.\n", pName);
            sleep( seconds: 2);   //Засыпаем на время стрижки.
            isCutted = false;
            --cameCustomers;
            --customersCnt;
        }
        isWait = false;
    }
    pthread_mutex_lock(&cut_m);    //Закрываем мутекс.
}
```

Рисунок 2 – Метод парикмахера

Москва 2020

Метод представляет парикмахера. В аргументах метода переменная void *num нужно для идентификации парикмахера. Через разыменование получаем строку – имя. Открываем в данном случае мутекс, т. к. он был закрыт в стрижке. В цикле, пока не обслужены все клиенты, есть два вложенных цикла. Первый – цикл ожидания нового посетителя, второй – обслуживает всю очередь. В конце мутекс закрывается.

3. Метод потребителя Costumer.

```
void *Customer(void *num) {
    int pNum = *((int *) num);
    //Имитируем очередь.
    pthread_mutex_lock(&custom_m);    //Закрываем мутекс.
    sleep( seconds: waitOrNot == 0 ? 0 : 4);//Засыпаем на ожидание.
    ++cameCustomers;    //Увеличиваем количество пришедших.
    printf( format: "Customer %d: I came to get a haircut. Hairdresser, wake up!\n", pNum);
    isCutted = true;
    isWait = true;
    pthread_mutex_unlock(&custom_m);    //Открываем мутекс.
    pthread_mutex_lock(&cut_m);
    while (isCutted);   //Ожидание, пока идет стрижка.
    printf( format: "Customer %d: What a great haircut! Thank you.\n", pNum);
    isExit = true;
    pthread_mutex_unlock(&cut_m);
}
```

Рисунок 3 – Метод Costumer

Метод представляет посетителя. В аргументах метода переменная void *num нужно для идентификации потребителя. Через разыменование получаем номер посетителя. Синхронизируем блок и засыпаем на 0 или 4 секунды – генерируется рандомом. 0 – очередь образуется, 4 – нет и тогда парикмахер сам будит посетителя. передаем isCutted = true, чтобы парикмахер начал стрижку и isWait – чтобы создалось ожидание. Блокируем мутекс для того, чтобы посетитель ушел из парикмахерской.

4. Метод main. Точка входа.

```
int main() {
    //Генерируем тип задачи: будет ли ждать или нет.
    srand( seed: time( timer: nullptr));
    waitOrNot = rand() % 2;
    //Инициализируем мутексы.
    pthread_mutex_init(&custom_m,  mutexattr: nullptr);
    pthread_mutex_init(&cut_m,  mutexattr: nullptr);

    //Парикмахер.
    pthread_t pthhairdresser;
    //Массив потоков-посетителей.
    pthread_t pthread[customersCnt];
    //Массив номеров посетителей.
    int pthint[customersCnt];

    //Выделяем поток для парикмахера.
    std::string hName = "Нечепорчук";
    pthread_create(&pthhairdresser,  attr: nullptr, Hairdresser, (void *) (&hName));
    for (int i = 0; i < customersCnt; ++i) {
        pthint[i] = i + 1;
        pthread_create(&pthread[i],  attr: nullptr, Customer,  arg: (void *) (pthint + i));
    }
    //Отдельный цикл, потому что в программе уменьшаем customerCnt.
    for (unsigned long i : pthread)
        pthread_join(i,  thread_return: nullptr); //Посетитель ждем своей очереди.

    return 0;
}
```

Рисунок 5 – Метод main

В самом начале генерируется переменная – будет ли образовываться очередь или нет, инициализируются два мутекса. Создаем поток парикмахера и массив потоков посетителей, запускаем их в цикле. В следующем цикле оформляем ожидание через join.

Тестирование программы

```
edwni@DESKTOP-BULN7MI:.../22508/CLionProjects/multithreading_t4$ ./a.out
Парикмахер Нечепорчук: Zzzz... Ожидаю посетителя..
Посетитель №1: Здравствуйте, можно постричься?
Парикмахер Нечепорчук: Пришел посетителя. Стригу его.
Посетитель №1: Очень нравится! Спасибо.
Парикмахер Нечепорчук: Zzzz... Ожидаю посетителя..
Посетитель №2: Здравствуйте, можно постричься?
Парикмахер Нечепорчук: Пришел посетителя. Стригу его.
Посетитель №2: Очень нравится! Спасибо.
Парикмахер Нечепорчук: Zzzz... Ожидаю посетителя..
Посетитель №3: Здравствуйте, можно постричься?
Парикмахер Нечепорчук: Пришел посетителя. Стригу его.
Посетитель №3: Очень нравится! Спасибо.
Парикмахер Нечепорчук: Zzzz... Ожидаю посетителя..
Посетитель №4: Здравствуйте, можно постричься?
Парикмахер Нечепорчук: Пришел посетителя. Стригу его.
Посетитель №4: Очень нравится! Спасибо.
Парикмахер Нечепорчук: Zzzz... Ожидаю посетителя..
Посетитель №5: Здравствуйте, можно постричься?
Парикмахер Нечепорчук: Пришел посетителя. Стригу его.
Посетитель №5: Очень нравится! Спасибо.
Парикмахер Нечепорчук: Zzzz... Ожидаю посетителя..
Посетитель №6: Здравствуйте, можно постричься?
Парикмахер Нечепорчук: Пришел посетителя. Стригу его.
Посетитель №6: Очень нравится! Спасибо.
Парикмахер Нечепорчук: Zzzz... Ожидаю посетителя..
Посетитель №7: Здравствуйте, можно постричься?
Парикмахер Нечепорчук: Пришел посетителя. Стригу его.
Посетитель №7: Очень нравится! Спасибо.
Парикмахер Нечепорчук: Zzzz... Ожидаю посетителя..
Посетитель №8: Здравствуйте, можно постричься?
Парикмахер Нечепорчук: Пришел посетителя. Стригу его.
Посетитель №8: Очень нравится! Спасибо.
Парикмахер Нечепорчук: Zzzz... Ожидаю посетителя..
Посетитель №9: Здравствуйте, можно постричься?
Парикмахер Нечепорчук: Пришел посетителя. Стригу его.
Посетитель №9: Очень нравится! Спасибо.
Парикмахер Нечепорчук: Zzzz... Ожидаю посетителя..
Посетитель №10: Здравствуйте, можно постричься?
Парикмахер Нечепорчук: Пришел посетителя. Стригу его.
Посетитель №10: Очень нравится! Спасибо.
```

Рисунок 6- результат выполнения 1

Как видно из (рис. 6), программа работает корректно. Парикмахер спит, ожидая посетителя. Посетитель приходит и будит парикмахера. Следующий заходит только после того, как проводили до выхода предыдущего.

Москва 2020

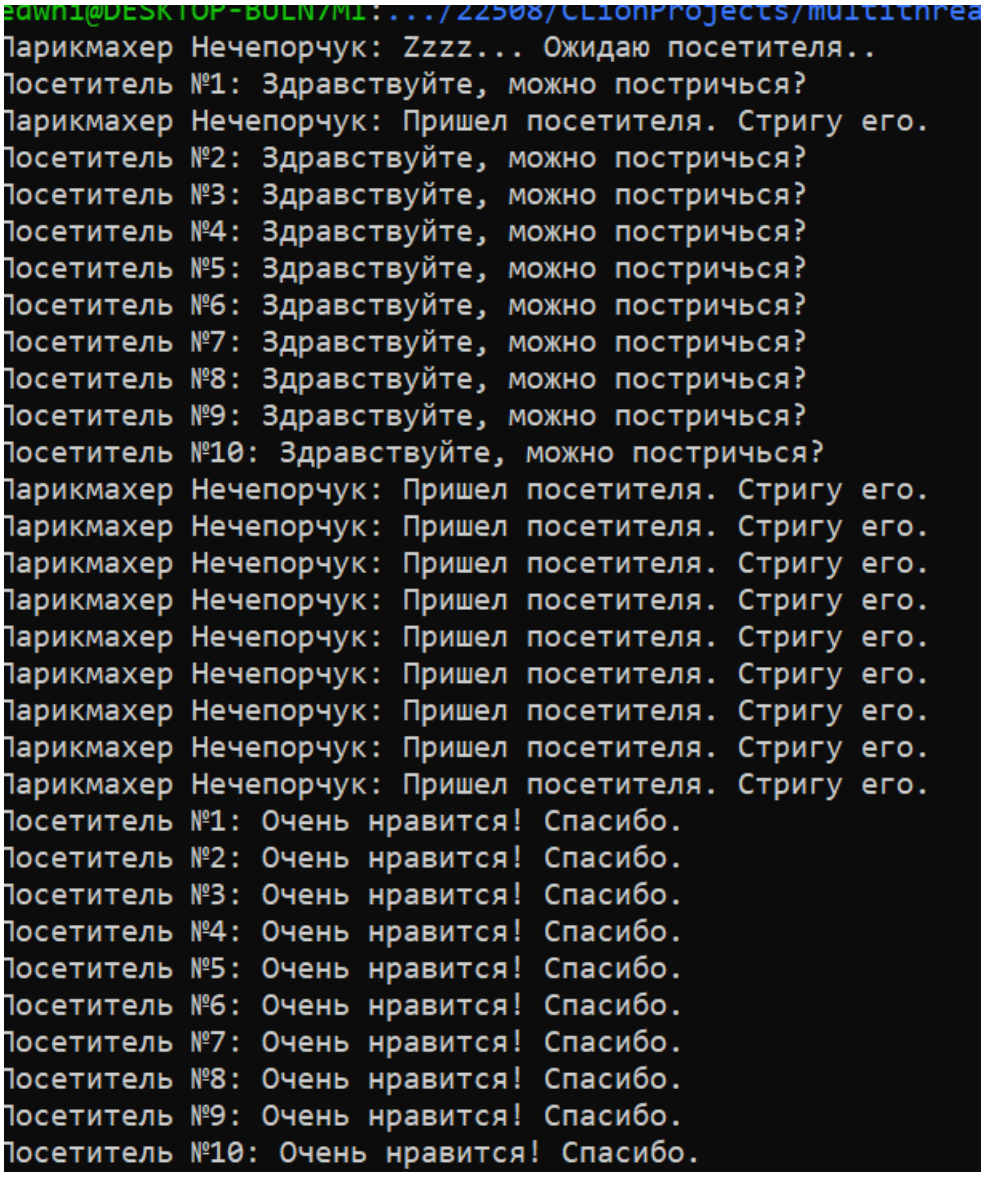

Рисунок 7 - результат выполнения 2

По (рис. 7) в данном случае получилось, что у нас образовывается очередь. Парикмахер стрижет каждого по очереди.

Используемые источники

1. SoftCraft, Архитектура вычислительных систем. Многопоточность [Электронный ресурс] //URL: http://www.softcraft.ru/edu/comparch/lect/07-parthread/multitreading.pdf, свободный. (дата обращения 11.12.2020, режим доступа: свободный).
2. SoftCraft, Архитектура вычислительных систем. Многопоточное программирование.Синхронизация [Электронный ресурс] //URL: http://www.softcraft.ru/edu/comparch/practice/thread/02-sync/, свободный. (дата обращения 11.12.2020, режим доступа: свободный).
3. ИНТУИТ. Операционные системы – аспекты параллелизма. [Электронный ресурс] //URL: https://intuit.ru/studies/courses/4447/983/lecture/14923?page=4, свободный. (дата обращения 11.12.2020, режим доступа: свободный).
4. iRunnerWiki, Библиотека pthreads. [Электронный ресурс] //URL: https://acm.bsu.by/wiki/Unix2019b/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_pthreads, свободный. (дата обращения: 11.12.2020, режим доступа: свободный).
5. learnc.info, Введение в мьютексы. Мьютексы в POSIX threads [Электронный ресурс] //URL: https://learnc.info/c/pthreads_mutex_introduction.html, свободный. (дата обращения: 11.12.2020, режим доступа: свободный).

Москва 2020