Архитектура вычислительных систем.
ИДЗ 4. Отчёт.
Вариант 22.
Работа на 8 баллов.

Глебов Павел. БПИ228.

19 декабря 2023 г.

1 Условие задачи

Задача о картинной галерее. Вахтер следит за тем, чтобы в картинной галерее одновременно было не более **50** посетителей. Для обозрения представлены **5** картин. Каждый посетитель случайно переходит от картины к картине, но если на желаемую картину любуются более десяти посетителей, он стоит в стороне и ждет, пока число желающих увидеть эту картину не станет меньше. Посетитель покидает галерею по завершении осмотра всех картин. Каждый посетитель уникальный (имеет свой номер). В галерею также пытаются постоянно зайти новые посетители, которые ожидают своей очереди и разрешения от вахтера, если та заполнена. Создать многопоточное приложение, моделирующее однодневную работу картинной галереи (например, можно ограничить числом посетителей от 100 до 300). **Вахтер и посетители — отдельные потоки.**

2 Основная программа (main.cpp):

```
#include "watchman.h"

int main() {
    for (int k = 1; k < 6; ++k) {
        std::string answerOutputFile;
        std::ofstream out;
        std::ifstream in("input" + std::to_string(k) + ".
            txt");
        std::cout << "Enter the number of visitors to the
            gallery:" << std::endl;
        in >> visitors;
```

```
std::vector<pthread_t> vecVisitor(visitors);
pthread_t watchmanPtr;

pthread_create(&watchmanPtr, nullptr, watchmanPth
    , static_cast<void *>(new int(0)));
outputProgram.emplace_back("The gallery is open!
    The watchman begins to let visitors in.");

for (int i = 0; i < visitors; ++i) {
    pthread_create(&vecVisitor[i], nullptr,
        visitorPth, static_cast<void *>(new int(i
        + 1)));
    if (i > MAX_VISITORS) {
        outputProgram.emplace_back("Visitor " +
            std::to_string(i + 1) + " comes to the
            gallery and waits.");
    } else {
        outputProgram.emplace_back("Visitor " +
            std::to_string(i + 1) + " came to the
            gallery.");
    }
}

for (int i = 0; i < visitors; ++i) {
    pthread_join(vecVisitor[i], nullptr);
}

pthread_cancel(watchmanPtr);
pthread_join(watchmanPtr, nullptr);
outputProgram.emplace_back("The gallery is closed
    ! We look forward to seeing you all next time.
    ");



std::cout << "Do you want to save the answer to a
    file? Enter Y/N:" << std::endl;
in >> answerOutputFile;

if (answerOutputFile == "Y") {
    std::string nameFile;
    std::cout << "Enter name file:" << std::endl;
    in >> nameFile;
    out.open(nameFile + ".txt");
```

```
            for (auto &el: outputProgram) {
                out << el << std::endl;
            }

            out.close();
        }


        std::cout << "Do you want to print the response
            to the console? Enter Y/N: " << std::endl;
        in >> answerOutputFile;

        if (answerOutputFile == "Y") {
            for (auto &el: outputProgram) {
                std::cout << el << std::endl;
            }
        }

        outputProgram.clear();
    }

    return 0;
}
```

Не стал загромождать отчет, поэтому код остальных программ не стал вставлять.

3 Тесты, которые проходит программа

Моя программа могла работать без входного набора данных, поскольку по условию этого не предполагалось, но я сделал так, чтобы можно было задать число посетителей в галереи. При запуске в консоль будет запрошено ввести это число, а после отработки будет запрошен вывод. Всего есть 2 варианта вывода: в файл, в консоль. Изначально будет предложен вывод в файл, и ввод имени файла(без расширения), а после можно также вывести в консоль. Такие тестовые файлы я решил создать для проверки своей программы:

1. 100 Y output1 N
2. 150 Y output2 N
3. 200 Y output3 N
4. 250 Y output4 N
5. 300 Y output5 N

На каждом тесте программа работает около 1-2 минут, поэтому проверка всех может занять значительное количество времени. Все результаты работы записались в файлы $output1.txt$, ..., $output5.txt$.

4 Алгоритм решения

Моя программа делает потоки - вахтера и посетителей. Первый поток нам нужен для того, чтобы вахтер проверял сколько человек смотрит на картины(вообще изначально так было задумано, но у меня что-то не получилось, поэтому я его вначале создал и проверку оформил в остальных потоках), а остальные потоки поздразумевались под поведение посетителей. Поток вахтёр изначально создается и остается быть созданным, а потом создается под каждого посетителя поток, и они пытаются подойти к картине, если есть свободное место, иначе ждут пока оно освободится.

5 Дополнительная информация по коду и условию

1. Я использовал разбиение большинства функций на файлы, чтобы код было проще понимать. В main.h я вынес все константы и глобальные переменные; в watchman.h вынес функцию по проверку картин вахтером; в visitor.h вынес функцию для поведения посетителя и ожидания, если все картины заняты; в random.h вынес свою функцию для генерации случайных чисел на промежутке.

2. Все переменные названы максимально по кодстайлу, чтобы не требовалось прилагать дополнительных комментариев по ним. Думаю, интуитивно будет понятно для чего каждая была создана.

3. В программе реализованы ввод с файлов и многократное использование, хотя условие моей задачи изначально этого не предполагало. Также вывод в файл или консоль после прекращения работы галереи (условие на 8 баллов).

4. Чтобы не испытывать проблему с доступом в консоль и настройки семафора, я выполнил запись в вектор строк, чтобы туда заносить информацию о работе галереи.

5. Функция main.cpp оформлена так, что сначала считывается с файла количество посетителей в галереи, создается поток вахтера и создаются потоки посетителей. Дальше идет поведение посетителей, то есть посетители начинают пытаться посмотреть на картины. Как только в галереи не остается людей, поток вахтера закрывается и галерея прекращает работу.

6. Я создал MUTEX для ограничения нескольких потоков изменения состояния картин. Также COND для контролирования посетителей при максимальном количестве смотрях на картину.