Правительство Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет «Высшая школа экономики»

Факультет компьютерных наук

Департамент программной инженерии

Отчет к домашнему заданию По дисциплине «Архитектура вычислительных систем»

Работу выполнил:

Студент группы БПИ-196 Мосолков Е.Н.

Москва 2020

Содержание

1. ЗАДАЧА

16. *Задача об инвентаризации по рядам.* После нового года в библиотеке университета обнаружилась пропажа каталога. После поиска и наказания виноватых, ректор дал указание восстановить каталог силами студентов. Фонд библиотека представляет собой прямоугольное помещение, в котором находится M рядов по N шкафов по K книг в каждом шкафу. Требуется создать многопоточное приложение, составляющее каталог. При решении задачи использовать метод «портфель задач», причем в качестве отдельной задачи задается составление каталога одним студентом для одного ряда.

2. РЕШЕНИЕ

Способ входных данных – аргументы командной строки.

Формат аргументов командной строки:

1. Путь к файлу
2. Путь к входному тесту (в формате .txt)
3. Путь к выходному файлу (в формате .txt)

Входной файл содержит 3 строки с числами – m, n, k

Программа считывает входные данные и создает catalog (тип vector<string>) с размером m в котором хранятся все записи каталога в формате строк. При этом catalog – это глобальная переменная (что сделано для доступа из разных задач)

Способ распределения задач – «Портфель задач». Каждый поток получает задачу по номеру (глобальная переменная taskCount), выполняет задачу и добавляет результат в catalog, затем проверяет остались ли еще задачи, и если да, то выполняет следующую задачу.

Используем mutex для разделения доступа к номеру задачи (глобальная переменная taskCount)

Для ввода и вывода данных в файл используем библиотеку fstream

После формирования catalog, происходит вывод в консоль и в файл.

КОД ПРОГРАММЫ

```
#include <iostream>
#include <vector>
#include <string>
#include <fstream>
#include <thread>
#include <mutex>

using namespace std;

int m, n, k, taskCount = 0;

vector<string> catalog;

string getRow(int row);

vector<string> getArguments(ifstream &in);

/**
 * A task method that fills catalog
 */
void task() {
    mutex mut;

    // calculate row number
    mut.lock();
    int row = taskCount++;
    mut.unlock();

    while (taskCount - 1 < m) {
        // calculate row
        string val = getRow(row);
        // we add a row to the catalog
        catalog.insert(catalog.begin() + row, val);

        // increase row count
        mut.lock();
        row = taskCount++;
        mut.unlock();
    }
}

/**
 * This method gets a row in string format
 */
string getRow(int row) {
    string str = "row: " + to_string(row + 1) + "\n";
    for (int i = 1; i <= n; i++) {
        str += "\tshelf: " + to_string(i) + "\n";
        for (int j = 1; j <= k; j++) {
            str += "\t\tbook: " + to_string(j) + "\n";
        }
    }
    return str;
}

/**
 * This method reads a text file and returns a vector of m, n, k string values
 */
vector<string> getArguments(ifstream &in) {
    vector<string> args;
    string arg;
    while (!in.eof()){
```

```
        getline(in, arg);
        args.push_back(arg);
    }
    return args;
}

int main(int argc, char** argv) {
    // Check if there are invalid arguments in command prompt
    if (argc != 3) {
        cout << "Command prompt arguments are class path, input file path "
                "and output file path\nThree and only three arguments allowed";
        return -1;
    }
    ifstream in;
    ofstream out;
    in.open(argv[1]);
    vector<string> args = getArguments(in);
    in.close();
    m = stoi(args[0]);
    n = stoi(args[1]);
    k = stoi(args[2]);
    catalog.resize(m);
    thread firstThread(task);
    thread secondThread(task);
    firstThread.join();
    secondThread.join();
    out.open(argv[2]);
    for (int i = 0; i < m; ++i) {
        string outVal = catalog.at(i);
        cout << outVal + "\n";
        out << outVal + "\n";
    }
    out.close();
    return 0;
}
```

3. ТЕСТИРОВАНИЕ

3.1. Параметры командной строки

При вводе некорректного числа аргументов, программа завершается с кодом -1, и выводит сообщение в консоль о том, что количество аргументов командной строки недопустимо.

3.2. Корректные данные

На рисунке 1 показан набор входных данных 3х тестов, на рисунке 2 показан вывод для этих тестов, а рисунок 3 демонстрирует вывод в консоль для теста 1.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. YouTube канал «#SimpleCode» плейлист «Многопоточное программирование»: https://www.youtube.com/watch?v=NawpxG81RRk&list=PLQOaTSbfxUtAc_RpyDiWCHq0YTzLtVSD0
2. SoftCraft «Практические приемы построения многопоточных приложений»: http://www.softcraft.ru/edu/comparch/tasks/t03/
3. Парадигмы параллельного программирования: https://pro-prof.com/forums/topic/parallel-programming-paradigms