Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №2 по курсу**

**«Операционные системы»**

Студент: Ибрагимов Далгат Магомедалиевич

Группа: М8О-208Б-22

Вариант: 16

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2023

**Содержание**

1. Репозиторий

2. Постановка задачи

3. Общие сведения о программе

4. Общий метод и алгоритм решения

5. Исходный код

6. Демонстрация работы программы

7. Выводы

**Репозиторий**

<https://github.com/L0ckR/OS_LABS>

**Задание**

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработки использовать стандартные средства создания потоков операционной системы (Windows/Unix). Ограничение максимального количества потоков, работающих в один момент времени, должно быть задано ключом запуска вашей программы.

**Описание работы программы**

Необходимо было написать программу для нахождения площади методом Монте-Карло. В данном методе можно распараллелить проверку точек на соответствие функции окружности. В ходе выполнения лабораторной работы я использовал следующие системные вызовы:

* pthread\_create() - создание потока
* pthread\_join() - ожидание завершения потока

**Исходный код**

**========================== monte-carlo.hpp ==========================**

**#pragma once**

**#include <iostream>**

**#include <vector>**

**#include <pthread.h>**

**#include <atomic>**

**#include <cmath>**

**#include <ctime>**

**struct Args{**

**double radius = 0;**

**size\_t dotsPerThread = 0;**

**std::atomic<size\_t> \* dotsInCircle = nullptr;**

**unsigned seed = time(nullptr);**

**};**

**struct Coordinates{**

**double x = 0;**

**double y = 0;**

**};**

**bool IsInCircle(const Coordinates &cords, double radius);**

**Coordinates CreateRandCoord(double radius, unsigned int &seed);**

**void \*task(void \*input);**

**double CircleArea(size\_t threadQuantity, double radius, size\_t dotsQuantity);**

**========================== monte-carlo.сpp ==========================**

**#include <monte-carlo.hpp>**

**bool IsInCircle(const Coordinates &cords, double radius)**

**{**

**return (cords.x \* cords.x + cords.y \* cords.y) <= radius \* radius;**

**}**

**Coordinates CreateRandCoord(double radius, unsigned int &seed)**

**{**

**Coordinates cords = {**

**(double)rand\_r(&seed) / (RAND\_MAX / radius),**

**(double)rand\_r(&seed) / (RAND\_MAX / radius)};**

**return cords;**

**}**

**void \*task(void \*input)**

**{**

**const auto &args = \*(reinterpret\_cast<Args \*>(input));**

**unsigned int seed = args.seed;**

**size\_t counter = 0;**

**for (size\_t i = 0; i < args.dotsPerThread; i++)**

**{**

**if (IsInCircle(CreateRandCoord(args.radius, seed), args.radius))**

**{**

**++counter;**

**}**

**}**

**(\*args.dotsInCircle).fetch\_add(counter);**

**return nullptr;**

**}**

**double CircleArea(size\_t threadQuantity, double radius, size\_t dotsQuantity)**

**{**

**std::atomic<size\_t> dotsInCircle {0};**

**double result = 0;**

**if (threadQuantity >= 1)**

**{**

**const size\_t actualThreadQuantity = std::min(threadQuantity, dotsQuantity);**

**std::vector<pthread\_t> threads(actualThreadQuantity);**

**const size\_t dotsPerThread = dotsQuantity / actualThreadQuantity;**

**std::vector<Args> argsForThread(actualThreadQuantity);**

**for (size\_t i = 0; i < actualThreadQuantity; i++)**

**{**

**argsForThread[i].radius = radius;**

**argsForThread[i].dotsPerThread = dotsPerThread;**

**argsForThread[i].dotsInCircle = &dotsInCircle;**

**argsForThread[i].seed += i;**

**if ((i + 1) \* dotsPerThread >= dotsQuantity)**

**{**

**argsForThread[i].dotsPerThread = dotsQuantity - i \* dotsPerThread;**

**pthread\_create(&threads[i], nullptr, task, reinterpret\_cast<void \*>(&argsForThread[i]));**

**}**

**else**

**{**

**pthread\_create(&threads[i], nullptr, task, reinterpret\_cast<void \*>(&argsForThread[i]));**

**}**

**}**

**for (auto &thread : threads)**

**{**

**pthread\_join(thread, nullptr);**

**}**

**result = ((double)dotsInCircle / dotsQuantity);**

**}**

**else**

**{**

**size\_t dotsInCircleSingle = 0;**

**unsigned int seed = (unsigned)time(nullptr);**

**for (size\_t i = 0; i < dotsQuantity; i++)**

**{**

**if (IsInCircle(CreateRandCoord(radius, seed), radius))**

**{**

**++dotsInCircleSingle;**

**}**

**}**

**result = ((double)dotsInCircleSingle / dotsQuantity);**

**}**

**return ((2 \* radius) \* (2 \* radius)) \* result;**

**}**

**========================== main.cpp ==========================**

**#include <monte-carlo.hpp>**

**int main(int argc, char const \*argv[])**

**{**

**if (argc != 2){**

**std::cout << "wrong arguments" << std::endl;**

**return -1;**

**}**

**std::cout.precision(15);**

**std::cout.setf(std::ios\_base::fixed, std::ios\_base::floatfield);**

**unsigned long threadQuantity = std::atol(argv[1]);**

**double radius;**

**size\_t dotsQuantity;**

**std::cin >> radius;**

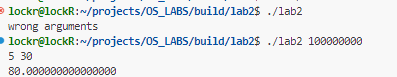
**std::cin >> dotsQuantity;**

**std::cout << CircleArea(threadQuantity, radius, dotsQuantity) << std::endl;**

**return 0;**

**}**

**Демонстрация работы программы**

****

**Выводы**

В результате выполнения данной лабораторной работы была написана программа на языке С++ для решения системы линейных уравнений методом

Гаусса, обрабатывающая данные в многопоточном режиме. Я приобрела практические навыки в управлении потоками в ОС и обеспечении синхронизации

между потоками.