Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №3 по курсу**

**«Операционные системы»**

Студент: Денисов Максим

Группа: М8О-207Б-21

Вариант: 16

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2022

**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/ClownOff/OS

**Постановка задачи**

**Цель работы**

Приобретение практических навыков в:

· Управление потоками в ОС

· Обеспечение синхронизации между потоками

**Задание**

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработке использовать стандартные средства создания потоков операционной системы (Windows/Unix). Ограничение потоков должно быть задано ключом запуска программы.  
Необходимо уметь продемонстрировать количество потоков, используемых программой, с помощью стандартных средств операционной системы.

Привести исследование зависимости ускорения и эффективности алгоритма от входящих данных и количества потоков. Объяснить получившиеся результаты.

Вариант 16: Задаётся радиус окружности. Необходимо с помощью метода Монте-Карло рассчитать её площадь

**Общие сведения о программе**

Для компиляции программы требуется указать ключ -pthread. Для запуска программы в качестве 1 аргумента командной строки необходимо указать радиус окружности, в качестве 2 аргумента ‑­ количество проверяемых точек, в качестве 3 аргумента ‑­ количество потоков.

* pthread\_create() — создание потока с передачей ему аргументов. В случае успеха возвращает 0.
* pthread\_join() — ожидает завершения потока обозначенного THREAD\_ID. Если этот поток к тому времени был уже завершен, то функция немедленно возвращает значение.

**Общий метод и алгоритм решения**

Пусть на вход от пользователя поступило n точек и m потоков. Тогда каждый поток будет обрабатывать n/m точек. Чтобы избежать работы с критической областью памяти будем хранить количество точек, попавших в круг и обрабатываемых i-м потоком, в динамическом массиве N.

Каждый поток обрабатывает n/m точек. Для этого генерируется случайная точка с координатами x, y, лежащая в пределах квадрата . При этом если точка лежит в пределах круга (удовлетворяет неравенству ) , то инкрементируется значение N[i]. В качестве аргумента поток принимает количество обрабатываемых точек и номер i. После завершения работы всех потоков выводится площадь, вычисленная методом Монте-Карло и стандартным методом

**Исходный код**

#include <stdio.h>

#include <math.h>

#include <pthread.h>

#include <stdlib.h>

#include <time.h>

int R;

int \*N;

typedef struct arguments {

    int points;

    int i;

} Arg;

double get\_rand() { // возврат рандомного числа от 0 до 1

    return ((double) rand()) / RAND\_MAX;

}

double get\_rand\_range(double min, double max) { // возвращает рандомное число от min до max

    return get\_rand() \* (max - min) + min;

}

void \*thread\_function(void \*args) { // cсоздаёт n рандомных точек в квадрате размером 2\*R и проверяет находятся ли точки в круге

    Arg \*arg = (Arg \*) args;

    int n = arg->points;

    int i = arg->i;

    for (int j = 0; j < n; j++) {

        double x = get\_rand\_range(-R, R);

        double y = get\_rand\_range(-R, R);

        if (x \* x + y \* y <= R \* R) {

            N[i]++;

        }

    }

    return NULL;

}

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

    if (argc != 4) {

        printf("./a.out Radius Number\_of\_points Number\_of\_threads\n");

        exit(1);

    }

    R = atoi(argv[1]);

    int points\_num = atoi(argv[2]), threads\_num = atoi(argv[3]);

    N = (int \*) calloc(threads\_num, sizeof(int)); // массив точек в круге

    double time\_spent = 0.0;

    clock\_t begin = clock();

    pthread\_t \*threads = (pthread\_t \*) calloc(threads\_num, sizeof(pthread\_t));

    if (threads == NULL) {

        printf("Can't allocate memory for threads\n");

        exit(1);

    }

    int points\_for\_thread = points\_num / threads\_num;

    Arg a;

    for (int i = 0; i < threads\_num; i++) {

        a.points = points\_for\_thread + (i < (points\_num % threads\_num));

        a.i = i;

        if (pthread\_create(&threads[i], NULL, thread\_function, &a) != 0) {

            printf("Can not create thread\n");

            exit(1);

        }

    }

    for (int i = 0; i < threads\_num; i++) {

        if (pthread\_join(threads[i], NULL) != 0) {

            printf("Join error\n");

            exit(1);

        }

    }

    double n = 0;

    for (int i = 0; i < threads\_num; i++) { // подсчёт точек в круге

        n += N[i] / 1.0 / points\_num;

    }

    printf("Monte-Carlo Circle square is %.5f\n",

           (double) 4 \* R \* R \* n);

    printf("Real Circle square is %.5f\n", (double) M\_PI \* R \* R);

    clock\_t end = clock();

    time\_spent += (double)(end - begin);

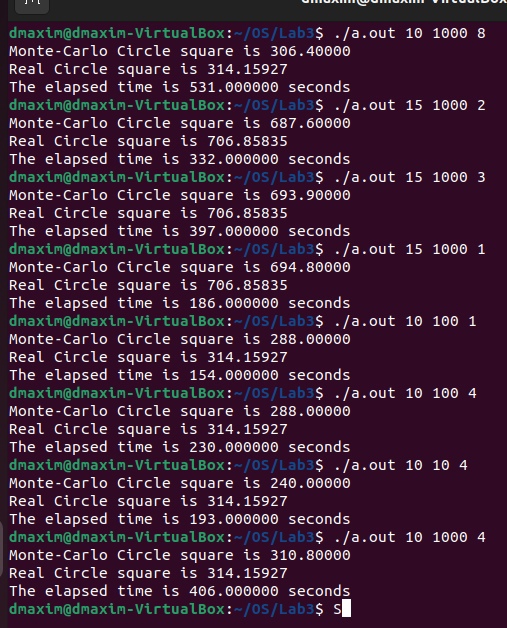
    printf("The elapsed time is %f seconds\n", time\_spent),

    free(threads);

    return 0;

}

**Демонстрация работы программы**



**Выводы**

Язык Си позволяет пользователю взаимодействовать с потоками операционной системы. Для этого на Unix-подобных системах требуется подключить библиотеку pthread.h.

Многопоточность – один из способов ускорить обработку каких-либо данных: выполнение однотипных, не зависящих друг от друга задач, можно поручить отдельным потокам, которые будут работать параллельно.

Средствами языка Си можно совершать системные запросы на создание потока, ожидания завершения потока, а также использовать различные примитивы синхронизации.