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: Кривошапкин Егор Борисович

Группа: М8О-209Б-23

Вариант: 8

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2024

**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/EgorX2000/os\_labs/tree/main/2

**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 8: Есть K массивов одинаковой длины. Необходимо сложить эти массивы. Необходимо предусмотреть стратегию, адаптирующуюся под количество массивов и их длину (по количеству операций)

**Общие сведения о программе**

Программа написана на языке Си в операционной системе Windows. Для запуска программы в качестве аргумента командной строки необходимо указать максимальное количество используемых потоков.

**Общий метод и алгоритм решения**

Пусть на вход от пользователя поступило n потоков. Тогда каждый поток будет обрабатывать k/n массивов.

**Исходный код**

main.cpp

#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() { // return random double from 0 to 1

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

}

double get\_rand\_range(double min, double max) { // return random double from min to max

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

}

void \*thread\_function(void \*args) { // create n random points of square size 2\*R and check if point at circle

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("Syntax: ./\*executable\_file\_name\* 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)); // array of number points at circle

srand(time(NULL));

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++) { // calculate points at circle

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

}

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

(double) 4 \* R \* R \* n); // (Circle Square)/(Square of square with size 2\*R) = N/M, where M = points\_num

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

free(threads);

return 0;

}

**Демонстрация работы программы**

[Temi4@localhost OS]$ cd 3\_lab/src/

[Temi4@localhost src]$ gcc main.c -pthread

[Temi4@localhost src]$ ./a.out

Syntax: ./\*executable\_file\_name\* Radius Number\_of\_points Number\_of\_threads

[Temi4@localhost src]$ ./a.out 1 100 5

Monte-Carlo Circle square is 3.12000

Real Circle square is 3.14159

[Temi4@localhost src]$ ./a.out 1 100000 8

Monte-Carlo Circle square is 3.14156

Real Circle square is 3.14159

[Temi4@localhost src]$ ./a.out 4 100000 8

Monte-Carlo Circle square is 50.08896

Real Circle square is 50.26548

**Выводы**

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

Создание потоков происходит быстрее, чем создание процессов, а все потоки используют одну и ту же область данных. Поэтому 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.