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: Щеглаков Илья Викторович

Группа: М8О-206Б-20

Вариант: 14

Преподаватель:

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2022

**Содержание**

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Исходный код
5. Выводы

**Репозиторий**

**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 14: есть колода из 52 карт, рассчитать экспериментально (метод Монте-Карло) вероятность того, что сверху лежат две одинаковых карты. Количество раундов подается с ключом.

**Общие сведения о программе**

Программа написана на языке Си в UNIX-подобной операционной системе (Ubuntu). 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.

**Исходный код**

#include <stdio.h>

#include <pthread.h>

#include <stdlib.h>

#include <time.h>

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 and cheak

Arg \*arg = (Arg \*) args;

int n = arg->points;

int i = arg->i;

for (int j = 0; j < n; j++) {

double x = get\_rand\_range(0, 52);

double y = get\_rand\_range(0, 51);

for (int k = 0; k < 52; k += 4) {

if (k <= x && x < k + 4 && k <= y && y < k + 3) N[i]++;

}

}

return NULL;

}

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

if (argc != 3) {

printf("Syntax: ./\*executable\_file\_name\* Rounds Number\_of\_threads\n");

exit(1);

}

int Rounds = atoi(argv[1]), threads\_num = atoi(argv[2]);

N = (int \*) calloc(threads\_num, sizeof(int)); // array of number points

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 = Rounds / threads\_num;

Arg a;

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

a.points = points\_for\_thread + (i < Rounds % 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

n += (double) N[i] / Rounds;

}

printf("Monte-Carlo chance is %.5f\n", (double) n);

printf("Real chance is %.5f\n", (double) 1 / 17);

free(threads);

return 0;

}

**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.