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: Румынина Екатерина Александровна

Группа: М8О-201Б-21

Вариант: 14

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/KetRum0/mai_os_labs>

**Постановка задачи**

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

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**

main.c – основная программа, принимающая на вход количество потоков и раундов, которое затем передает в функцию chance.

lab3.h – объявление функции chance.

lab3.c – реализация функции chance с применением потоков.

**Общий метод и алгоритм решения**

Получив количество потоков и количество раундов, разделяем поровну раунды между потоками. Каждый поток определённое количество раз генерирует два числа, которые соответствуют картам, и проверяет их совпадение. Если совпало – повышает счётчик. Затем количество всех совпадений делится на количество всех раундов и получается необходимая в задании вероятность.

**Исходный код**

**main.c**

#include <lab3.h>

#include <stdlib.h>

#include <stdio.h>

#include <pthread.h>

#include <unistd.h>

#include <time.h>

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

int n;

int threadCount;

scanf("%d",&n);

scanf("%d",&threadCount);

if ((threadCount <= 0) | (n <= 0))

{

perror("Wrong arguments");

exit(EXIT\_FAILURE);

}

double result = chance(n, threadCount);

printf("%g\n",result);

}

**lab.c**

#define INT\_MAX 2147483647

#include <stdlib.h>

#include <stdio.h>

#include <pthread.h>

#include <unistd.h>

#include <time.h>

int\* count;

int deck[] = {0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5,6,6,6,6,7,7,7,7,8,8,8,8,9,9,9,9,10,10,10,10,11,11,11,11,12,12,12,12};

typedef struct arguments{

int rounds\_for\_thread;

int num\_of\_thread;

int seed\_thread;

}Arguments;

void \* thread\_func(void\* args){

int card1, card2;

Arguments \* Args = (Arguments \*) args;

int r = Args -> rounds\_for\_thread;

int n = Args -> num\_of\_thread;

int seed = Args -> seed\_thread;

for (int i = 0; i < r; i++){

card1 = 0;

card2 = 0;

while (card1==card2){

card1 = rand\_r(&seed) % 52;

card2 = rand\_r(&seed) % 52;

}

if (deck[card1] == deck[card2]){

count[n]++;

}

}

}

double chance(int n, int threadCount){

int rounds, threads;

rounds = n;

threads = threadCount;

int rounds\_thread = rounds/threads;

count = (int \*) calloc(threads, sizeof(int));

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

Arguments a[threads];

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

a[i].num\_of\_thread = i;

a[i].rounds\_for\_thread = rounds\_thread;

a[i].seed\_thread = rand();

}

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

if (pthread\_create(&th[i], NULL, &thread\_func, (void\*) &a[i]) != 0){

perror("pthread\_create error");

exit(EXIT\_FAILURE);

}

}

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

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

perror("thread\_join error");

exit(EXIT\_FAILURE);

}

}

int sum\_count = 0;

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

sum\_count += count[i];

}

// end = time(NULL);

// int time\_taken = end - start;

// printf("Time taken to execute in seconds : %d\n", time\_taken);

// printf("%g\n",(double) sum\_count/rounds);

// printf("Real chance: %g\n", (double) 3/51);

double result = (double) sum\_count/rounds;

return result;

}

**lab.h**

#ifndef OS\_LABS\_LAB3\_H

#define OS\_LABS\_LAB3\_H

double chance(int n, int threadCount);

#endif //OS\_LABS\_LAB3\_H

**Демонстрация работы программы**

ket@ket-laptop:~/Desktop/mai\_os\_labs/lab3$ ./lab3

10000 1

0.0571

ket@ket-laptop:~/Desktop/mai\_os\_labs/lab3$ ./lab3

10000000 1

0.0588707

ket@ket-laptop:~/Desktop/mai\_os\_labs/lab3$ ./lab3

10000000 2

0.0586865

ket@ket-laptop:~/Desktop/mai\_os\_labs/lab3$ ./lab3

1000000000 4

0.0588157

ket@ket-laptop:~/Desktop/mai\_os\_labs/lab3$ ./lab3

1000000000 8

0.0588158

ket@ket-laptop:~/Desktop/mai\_os\_labs/lab3$ ./lab3

1000000000 10

0.0588163

**Выводы**

В результате выполнения данной лабораторной работы я научилась управлять потоками в ОС и обеспечивать синхронизацию между ними.