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

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.