Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №3 по курсу
«Операционные системы»

Потоки операционных систем

Студент: Никулин Кристиан Ильич
Группа: М8О–208Б–21
Вариант: 14
Преподаватель: Соколов Андрей Алексеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022.

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 обработки использовать стандартные средства создания потоков операционной системы (Windows/Unix). Ограничение потоков должно быть задано ключом запуска вашей программы.

Так же необходимо уметь продемонстрировать количество потоков, используемое вашей программой с помощью стандартных средств операционной системы.

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 представляет из себя один файл main.c

При компиляции необходимо указать ключ -pthread для работы с потоками

Общий метод и алгоритм решения.

Для запуска программы необходимо указать два ключа: кол-во попыток и кол-во потоков. Попытки равномерно распределяются между потоками. В каждом высчитывается вероятность, что две первые карты одинаковые, после чего все результаты складываются и сравниваются с реальной вероятностью. Чем больше у нас будет попыток, тем ближе результат к достоверному.

Основные файлы программы

main.c:

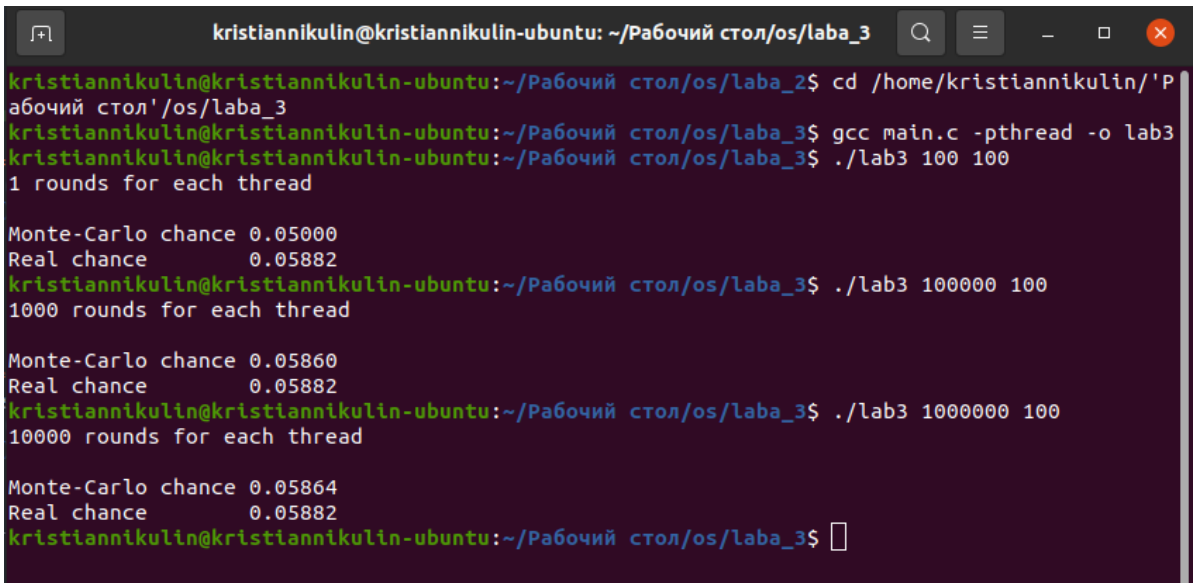
```
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>
#include <string.h>
```

```
#include <fcntl.h>
#include <wait.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/uio.h>
#include <pthread.h>
#include <errno.h>
#include <ctype.h>
#include <time.h>
int * N;
typedef struct arguments {
    int r;
    int ii;
}
Arg;
void * thread_function(void * args) {
    Arg * arg = (Arg * ) args;
    int r = arg -> r;
    int ii = arg -> ii;
    int deck0[] = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,
        26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51};
    int deck1[52], deck3[52]; //51
    int ik = 51, ij, ijk;
    for (int j = 0; j < r; j++) {
        for (int i = 0; i < 52; i++) {
            deck1[i] = deck0[i];
        }
        ijk = 51;
        for (int i = 0; i <= ik; i++) {
            if (i != ik) {
                ij = rand() % ijk;
            } else {
                ij = 0;
            }
```

```
            deck3[i] = deck1[ij] % 13;
            deck1[ij] = deck1[ijk];
            ijk--;
        }
        if ((deck3[ik]) == (deck3[ik - 1])){
            N[ii]++;
        }
    }
}
int main(int argc, char * argv[]) {
    if (argc != 3) {
        printf("syntax error ./lab3 count_of_rounds number_of_threads\n");
        exit(1);
    }
    clock_t begin = clock();
    int rounds = atoi(argv[1]), threads_num = atoi(argv[2]);
    N = (int * ) calloc(threads_num, sizeof(int));
    pthread_t * threads = (pthread_t *) calloc(threads_num, sizeof(pthread_t));
    if (threads == NULL) {
        printf("threads memory error\n");
        exit(2);
    }
    int rounds_for_thread = rounds / threads_num;
    printf("%d rounds for each thread\n", rounds_for_thread);
    Arg a;
    for (int i = 0; i < threads_num; i++) {
        a.r = rounds_for_thread;
        a.ii = i;
        if (pthread_create(&threads[i], NULL, thread_function, &a) != 0) {
            printf("create thread error\n");
            exit(3);
        }
    }
    for (int i = 0; i < threads_num; i++) {
```

```
        if (pthread_join(threads[i], NULL) != 0) {
            printf("join error\n");
            exit(4);
        }
    }
    double ans = 0;
    for (int i = 0; i < threads_num; i++) {
        ans += (double) N[i] / rounds;
    }
    clock_t end = clock();
    double time_spent = 0.0;
    time_spent += (double)(end - begin) / CLOCKS_PER_SEC;
    printf("Working time - %f\n", time_spent);
    printf("Monte-Carlo chance %.5f\n", (double) ans);
    printf("Real chance        %.5f\n", (double) 3 / 51);
    free(threads);
    return 0;
}
```

Примеры работы



Вывод

Проделав лабораторную работу, я приобрёл практические навыки в управлении потоками в ОС Unix и обеспечении синхронизации между ними.