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: Люгге Т.В.

Группа: М80-201Б-21

Вариант: 14

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2023

**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/MonkeDLyugge/LabOS

**Постановка задачи**

**Цель работы**

Целью является 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.

В отчете привести исследование зависимости ускорения и эффективности алгоритма от входящих данных и количества потоков. Получившиеся результаты необходимо объяснить.

**14 Вариант**

**Есть колода из 52 карт, рассчитать экспериментально (метод Монте-Карло) вероятность**

**того, что сверху лежат две одинаковых карты. Количество раундов подается с ключом**

**Общий метод и алгоритм решения**

На вход подается подается количество экспериментов. Так каждому потоку необходимо рандомно «вытянуть две карты» (так мы обеспечим качественную «тасовку» карт) и сравнить их на значение.

**Исходный код**

**main.cpp**

#include <iostream>

#include <cstdio>

#include <ctime>

#include "t\_handler.hpp"

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

if (argc != 2) {

perror("Wrong arguments number");

return -1;

}

int numberOfExperiments = std::atoi(argv[1]);

if (numberOfExperiments <= 0) {

perror("Argument must be more than ZERO");

return -1;

}

int sum = Chances(numberOfExperiments);

std::cout << (double) sum / (double) numberOfExperiments << "\n";

return 0;

}

|  |
| --- |
| **t\_handler.cpp** |
| #include "t\_handler.hpp"  #include <pthread.h>  struct TData {  std::vector<TCard>& deck;  int success;  int thread;  TData(std::vector<TCard>& d, int s, int t) : deck(d), success(s), thread(t) {}  };  std::vector<TCard> NewDeck() {  const std::string names[CARD\_NAMES] = {"2", "3", "4", "5", "6", "7", "8", "9", "10",  "J", "Q", "K", "T"};  int k = 0;  TCard newCard;  std::vector<TCard> deck(CARDS);  for (int i = 0; i < CARD\_NAMES; i++) {  newCard.name = names[i];  newCard.suit = 'D';  deck[k] = newCard;  k++;  newCard.suit = 'C';  deck[k] = newCard;  k++;  newCard.suit = 'H';  deck[k] = newCard;  k++;  newCard.suit = 'S';  deck[k] = newCard;  k++;  }  return deck;  }  void\* ThreadExperiment(void\* argv) {  std::vector<TCard>deck = ((TData \*) argv)->deck;  int &thread = ((TData \*) argv)->thread;  int &success = ((TData \*) argv)->success;    srand(time(nullptr) \* thread);  int firstCard = rand() % CARDS;  srand(time(nullptr) \* thread \* 2);  int secondCard = rand() % CARDS;  while (firstCard == secondCard) {  secondCard = rand() % CARDS;  }  success += (deck[firstCard].name == deck[secondCard].name);  return nullptr;  }  int Chances(int numberOfThreads) {  std::vector<pthread\_t>threads(numberOfThreads);    std::vector<TCard> deck = NewDeck();  TData data(deck, 0, numberOfThreads);  for (int i = 0; i < numberOfThreads; i++) {  if (pthread\_create(&threads[i], nullptr, ThreadExperiment, &data)) {  perror("Thread create error");  exit(EXIT\_FAILURE);  }  }  for(int i = 0; i < numberOfThreads; i++) {  if (pthread\_join(threads[i], nullptr) != 0) {  perror("Can't wait for thread\n");  exit(EXIT\_FAILURE);  }  }    return data.success;  } |

**t\_handler.hpp**

#ifndef THANDLER\_HPP

#define THANDLER\_HPP

#include <string>

#include <vector>

#include <iostream>

struct TCard {

std::string name; // = {2, 3, 4, ..., J, Q, K, T};

char suit; // = {D, H, C, S}

};

const int CARDS = 52;

const int CARD\_NAMES = 13;

std::vector<TCard> NewDeck();

void\* ThreadExperiment(void\* argv);

int Chances(int numberOfThreads);

#endif

**Демонстрация работы программы**

[microhacker@microhacker-HLYL-WXX9](mailto:microhacker@microhacker-HLYL-WXX9):~/Desktop/LabOS$ ./main.cpp 4

0.1

**Выводы**

Составлена и отлажена многопоточная программа на языке Си. Тем самым, приобретены навыки в распараллеливании вычислений, управлении потоками и обеспечении синхронизации между ними.