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: Кажекин Д.А.

Группа: М8О-207Б-21

Вариант: 19

Преподаватель: Черемисинов Максим

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/DKazhekin/OC

**Постановка задачи**

**Цель работы**

Целью является приобретение практических навыков в:

Управление потоками в ОС

Обеспечение синхронизации между потоками

**Заданиею**

Необходимо реализовать проверку числа на простоту при помощи алгоритма «решето Эратосфена»

**Общие сведения о программе**

Для начала работы необходимо скомпилировать файл lab3.c. При запуске следует указать ключ типа int.

**Общий метод и алгоритм решения**

Отдельно реализована функция prime(), которая непосредственно осуществляет проверку числа на простоту. \*routine() вызывается каждым новым созданным потоком для выполнения поставленной задачи (внутри содержит вызов prime() ).

В начале работы программы пользователь должен ввести N чисел для обработки. Далее с помощью pthread\_create() и цикла создается N потоков, которые решают нашу задачу.

**Исходный код**

#include "unistd.h"

#include "time.h"

#include "stdio.h"

#include "stdlib.h"

#include "pthread.h"

#include "string.h"

// Mutex используется, чтобы защитить код от выполнения другими потоками одновременно

int prime(int digit) {

int i, j;

int number = 1000;

int primes[number + 1];

//populating array with naturals numbers

for (i = 2; i <= number; i++)

primes[i] = i;

i = 2;

while ((i \* i) <= number) {

if (primes[i] != 0) {

for (j = 2; j < number; j++) {

if (primes[i] \* j > number)

break;

else

// Instead of deleteing , making elemnets 0

primes[primes[i] \* j] = 0;

}

}

i++;

}

for (int k = 0; k < 1001; k++) {

if (primes[k] == digit) {

return 1;

}

}

return 2;

}

void \*routine(void \*arg) {

int number = \*(int \*) arg;

int t = prime(number);

if (t == 1){

printf("The number %d is prime\n", number);

}

else{

printf("The number %d is not prime\n", number);

}

free(arg);

return NULL;

}

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

int n = atoi(argv[1]);

pthread\_t pid[n];

int data[n];

printf("Enter %d numbers!\n", n);

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

int a;

scanf("%d", &a);

data[i] = a;

}

double start = clock();;

if (argc == 2) {

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

int \* a = malloc(sizeof(int));

\*a = data[i];

if (pthread\_create(&pid[i], NULL, &routine, a) != 0) {

perror("Couldn't create a thread\n");

return 1;

}

printf("Thread %d has started\n", i);

}

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

if (pthread\_join(pid[i], NULL) != 0){ // Если pthread\_join() оставить в предыдущем for, то не будет

return 2; // Одновременности выполнения, т.к. поток создатся и программа

} // Будет ждать завершения работы потока для создания нового

printf("Thread has finished execution!\n");

}

}

else{

printf("Please enter an appropriate program key !\n");

}

printf("Count of threads: %d\n", n);

printf("Программа работала %.4lf секунд\n", (clock() - start) / (CLOCKS\_PER\_SEC));

return 0;

}

**Демонстрация работы программы**

deniskazhekin@Deniss-MacBook-Air project % gcc parent1.c -o parent

deniskazhekin@Deniss-MacBook-Air project % ./parent 5

Enter 5 numbers!

5

2

4

1

2

Thread 0 has started

The number 5 is prime

Thread 1 has started

The number 2 is prime

Thread 2 has started

The number 4 is not prime

Thread 3 has started

Thread 4 has started

The number 1 is not prime

Thread has finished execution!

Thread has finished execution!

Thread has finished execution!

Thread has finished execution!

The number 2 is prime

Thread has finished execution!

Count of threads: 5

Программа работала 0.0007 секунд

**Сводка о времени работы программы в зависимости от количества задействованных потоков**

Count of threads: 10 🡪 Программа работала 0.0011 секунд

Count of threads: 20 🡪 Программа работала 0.0027 секунд

Count of threads: 50 🡪 Программа работала 0.0047 секунд

Count of threads: 100 🡪 Программа работала 0.0084 секунд

**Выводы**

Составлена и отлажена многопоточная программа на языке Си, выполняющая проверку числа на простоту алгоритмом Эратосфена. Тем самым, приобретены навыки в распараллеливании вычислений, управлении потоками и обеспечении синхронизации между ними.