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: Гордовой Д.С.

Группа: М8о–201Б–19

Вариант: 19

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2020.

**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.

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

**Общие сведения о программе**

Для компиляции программы требуется указать ключ -pthread. При запуске в качестве первого параметра указываем количество потоков а вторым наше число, которое будет проходить проверку на простоту

В программе есть потоковаю функцию void\*rem\_multipl(void\* i), в которой помечаются все числа решета, кратные i. Так как все потоки программы работают в одном и том же пространстве памяти, аргументы для передачи потоковой функции хранятся по разным адресам (в массиве, размер которого равен количеству потоков).

В программе предусмотрена проверка на системные ошибки – ошибки выделения памяти, ошибки запуска.

**Общий метод и алгоритм решения**.

При запуске программы у пользователя запрашивается число num, которое необходимо проверить на простоту. Проверить на простоту можно только неотрицательное число.

Из аргументов командной строки берётся количество потоков, которое может использовать программа и само число для проверки. Производится выделение памяти для массива потоков, для массива аргументов потоковой функции и для самого решета. Решето представляет собой массив символов sieve (т.к. размер символьного типа char минимальный). sieve[i] равно нулю, если число простое и единице в противном случае.

По определению числа 0 и 1 не являются простыми, поэтому сразу помечаем их единицами в решете. Необходимо проверить все числа от 2 до num включительно. Если ячейка решета, соответствующая числу i, равна нулю, то это число простое и требуется «вычеркнуть» (пометить единицей) все числа, кратные i. Эта задача и делегируется другим потокам.

Потоковая функция rem\_multipl принимает на вход число i и помечает единицами все числа, кратные i.

Сделаем правило, по которому будет выбираться поток для выполнения функции. Объявим переменную cur\_thread и приравняем ее к нулю. Далее мы будем создавать поток с индексом now\_thred(mod therads\_num), где threads\_num – общее количество потоков. Потоки у нас якобы будут закольцованными, ведь когда cur\_thread становится больше количества потоков, потоки начинают использоваться повторно. Чтобы избежать ситуации, когда задача будет делегирована потоку, работа которого еще не окончена, будем дожидаться окончания работы потока. После делегирования задач, переменная now\_thread инкрементируется.

После обработки всего решета необходимо дождаться окончания работы всех активных потоков. После этого необходимо посмотреть число в sieve[num] и сделать вывод о его простоте.

**Основные файлы программы**

**lab3.c:**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include<time.h>

#include<sys/time.h>

char\* sieve;

long long num;

// функция потока - маркировка чисел, кратных i

void\* rem\_multipl(void\* i\_void) {

long long i = \*(long long\*)i\_void;

for (long long j = i \* i; j <= num; j += i) {

sieve[j] = 1;

}

pthread\_exit(NULL);

}

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

clock\_t begin = clock();

if (argc != 3) {

printf("Syntax: ./out Number\_of\_threads Number\_for\_test\n");

exit(1);

}

int threads\_num = atoi(argv[1]);

num = atoi(argv[2]);

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

if (threads == NULL) {

printf("Can't allocate space for threads\n");

exit(2);

}

// массив аргументов, которые будут переданы в функцию потока

long long\* args = (long long\*)malloc(threads\_num\* sizeof(long long));

if (args == NULL) {

printf("Can't create an array for arguments for threads\n");

exit(3);

}

// создаем массив, заполненный 0 для решета

// 0 - простое число, 1 - непростое число

sieve = (char\*)calloc((num + 1), sizeof(char));

if (sieve == NULL) {

printf("Can't create an array for sieve\n");

exit(3);

}

// маркировка чисел, не являющихся простыми по определению

sieve[0] = 1;

sieve[1] = 1;

int cur\_thread = 0; // id текущего потока

for (long long i = 2; i \* i <= num; ++i) {

if (sieve[i] == 1) { // пропуск не простых чисел

continue;

}

if (cur\_thread >= threads\_num) {

// мы должны подождать, пока работает необходимый поток

pthread\_join(threads[cur\_thread % threads\_num], NULL);

}

args[cur\_thread % threads\_num] = i; // копирование аргумента для функции потока в специальный массив

pthread\_create(&threads[cur\_thread % threads\_num], NULL, rem\_multipl, &args[cur\_thread % threads\_num]);

++cur\_thread;

}

// ждем все потоки

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

pthread\_join(threads[i], NULL);

}

if (sieve[num] == 1) {

printf("%lld is not a prime number\n", num);

}

else {

printf("%lld is a prime number\n", num);

}

clock\_t end = clock();

double time\_spent = (double)(end - begin) / CLOCKS\_PER\_SEC;

printf("%f\n", time\_spent);

free(sieve);

free(threads);

free(args);

}

**Пример работы**

kitanosa@kitanosa:~/labs/os/lab3$ ./a.out 1 -1

-1 is a prime number

0.000253

kitanosa@kitanosa:~/labs/os/lab3$ ./a.out 1 11

11 is a prime number

0.001128

kitanosa@kitanosa:~/labs/os/lab3$ ./a.out 2 11

11 is a prime number

0.001184

kitanosa@kitanosa:~/labs/os/lab3$ ./a.out 3 11

11 is a prime number

0.001192

kitanosa@kitanosa:~/labs/os/lab3$ ./a.out 2 15

15 is not a prime number

0.001291

**Вывод**

Чтобы создать поток понадобится меньше времени чем создать процесс. К тому же, потоки используют одну и ту же область памяти. Таким образом многопоточность позволяет ускорть обработку данных, сокращая время работы над однотипными задачаси не зависящими друг от друга.

В этой лабораторной я реализовал алгоритм проверки числа на простоту с помощью решета Эратосфена.