**Московский авиационный институт**
**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»
Дисциплина: «Операционные Системы»

# Лабораторная работа № 3

## Тема: Управление потоками в ОС и обеспечение синхронизации между потоками

Студент: Семин Александр Витальевич

Группа: М8О-206Б-20

Преподаватель: Соколов Андрей Алексеевич

Дата:

Оценка:

Москва, 2021

1. **Постановка задачи**

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработке использовать стандартные средства создания потоков операционной системы (Windows/Unix). Ограничение потоков должно быть задано ключом запуска вашей программы.

Также необходимо уметь демонстрировать 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.

Вариант 19.

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

2. **Описание программы**

Изначально создается массив, размер которого запрашивается у пользователя. Размер массива – граничное число, до которого выполняется алгоритм «решето Эратосфена». Затем массив заполняется числами от 1 до конечного. Также создается массив потоков. Далее происходит выполнение алгоритма с помощью многопоточности: каждый проход по массиву выполняется отдельным потоком. Для корректной работы многопоточности в программе используется мьютекс, который дает доступ потокам для обработки массива для избегания изменения одних данных несколькими потоками одновременно в одном месте. При завершении выполнения алгоритма у пользователя запрашиваются числа, простоту которых необходимо проверить.

3. **Набор тестов**

*Первый тестовый набор:*

```
progger@asus:~/Desktop/OS_labs/l3$ ./a.out 100
Введите размер массива:
94
<…> *вырезан список номеров потоков для удобства чтения
Преобразование массива с помощью "решета" Эратосфена выполнено
1 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89
Введите число от 1 до 89, которое хотите проверить на простоту
Для завершения программы введите q
```

3

Число 3 простое
Введите число от 1 до 93, которое хотите проверить на простоту
Для завершения программы введите q
39

Число 39 непростое
Введите число от 1 до 93, которое хотите проверить на простоту
Для завершения программы введите q
48

Число 48 непростое
Введите число от 1 до 93, которое хотите проверить на простоту
Для завершения программы введите q
49

Число 49 непростое
Введите число от 1 до 93, которое хотите проверить на простоту
Для завершения программы введите q
53

Число 53 простое
Введите число от 1 до 93, которое хотите проверить на простоту
Для завершения программы введите q
q

*Второй тестовый набор:*
Введите размер массива:
8
Поток 140475185637120 создан
Поток 140475177244416 создан
Поток 140475168851712 создан
Поток 140475160459008 создан
Поток 140475152066304 создан
Поток 140475143673600 создан
Поток 140475135280896 создан

Поток 140475126888192 создан
Ожидание потока 140475185637120
Ожидание потока 140475177244416
Ожидание потока 140475168851712
Ожидание потока 140475160459008
Ожидание потока 140475152066304
Ожидание потока 140475143673600
Ожидание потока 140475135280896
Ожидание потока 140475126888192
Преобразование массива с помощью "решета Эратосфена" выполнено
1 2 3 5 7
Введите число от 1 до 7, которое хотите проверить на простоту
Для завершения программы введите q
3

Число 3 простое
Введите число от 1 до 7, которое хотите проверить на простоту
Для завершения программы введите q
4

Число 4 непростое
Введите число от 1 до 7, которое хотите проверить на простоту
Для завершения программы введите q

## 4. Листинг программы

### main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>
#include <stdbool.h>

pthread_mutex_t mutex;


typedef struct {
    int p;
    int* arr;
    int size;
}data;

void resh_era(data* st) {
    int p = st->p;
    if (p == 0 || p == 1) {
        st->p++;
        return;
    }

    for (int i = 0; i < st->size; i++) {
        if (st->arr[i] % p == 0 && st->arr[i] != p) {
            st->arr[i] = 0;
        }
    }
    st->p++;
}

void* pthr_resh(void* arg) {

    pthread_mutex_lock(&mutex);
    data* st_ = (data*)arg;

    resh_era(st_);
    pthread_mutex_unlock(&mutex);
}

bool check_prostoe(data* st, int a) {
    for (int i = 0; i < st->size;i++) {
        if (st->arr[i] == a)
            return true;
    }
    return false;
}

void scan_int(int* user, char c_user) {
    int res = 0;
```

```
    res = c_user - '0';
    int i = 10;
    char c = getchar();
    while(c != ' ' && c != '\n' && c != '\0' && c != EOF) {
        if (c >= '0' && c <= '9') {
            res = res*i + (c -'0');
            i *= 10;
        } else {
            printf("user oshibsya\n");
            break;
        }
        c = getchar();
    }
    putchar(c);
    *user = res;
}

int main(int argc, char* argv[]) {
    if (argc > 2) {
        printf("Указан неверный ключ\n");
    } else if(argc < 2) {
        printf("Не указан ключ\n");
        return -1;
    }
    int thrs = atoi(argv[1]);

    int N;
    printf("Введите размер массива:\n");
    scanf("%d", &N);
    data st;
    st.p = 0;
    st.size = N;
    int thr_size = thrs;
    if (thrs < N) {
        printf("Недостаточное количество потоков\n");
        return -1;
    }
    int* new_ = malloc(sizeof(int)*N);

    if (new_ == NULL) {
        return -1;
    }

    st.arr = new_;
    bool arr_bool[N];

    for (int i = 0; i < N; i++) {
        arr_bool[i] = false;
        st.arr[i] = i;
    }

//////////////////////////////////////////////////
```

```
    pthread_t thread[thr_size];
    pthread_mutex_init(&mutex, NULL);


////////////// выполнение алгоритма решето эратосфена


    int status;
    for (int i = 0; i < N; i++) {
        if (arr_bool[i] == true) {
            continue;
        }

        status = pthread_create(&thread[i], NULL, pthr_resh, (void*)&st);
        printf("Поток %ld создан\n", thread[i]);
        if (status != 0) {
            return -1;
        }
        arr_bool[i] = true;
    }
    int ret;


////////// ожидание выполнения всех потоков


    for (int i = 0; i <N; i++) {
        ret = pthread_join(thread[i], NULL);
        printf("Ожидание потока %ld\n", thread[i]);
        if (ret != 0) {
            return -1;
        }
    }

    printf("Преобразование массива с помощью ''решета'' Эратосфена
выполнено\n");

    for (int i = 0; i < N; i++) {
        if (st.arr[i] == 0) {
            continue;
        } else {
            printf("%d ", st.arr[i]);
        }
    }
    printf("\n");
///////////////// пользовательское взаимодействие
    bool first = true;
    while(true) {
        if (!first) {
            printf("Введите число от 1 до %d, которое хотите проверить на
простоту\n", N-1);
            printf("Для завершения программы введите q\n");
        }
```

```
        char c = getchar();
        int user = 0;
        if (c == 'q') {
            break;
        } else if (!first && (c >= '0' && c <= '9')) {
            scan_int(&user, c);
        } else if (!first && (user >= N || user == 0 || user < 0)) {
            printf("Wrong value\n");
            continue;
        }
        bool res = false;
        if (!first) {
            res = check_prostoe(&st,user);
            if (res)
                printf("Число %d простое\n", user);
            else
                printf("Число %d непростое\n", user);
        }

        first = false;
    }

    free(st.arr);
    pthread_mutex_destroy(&mutex);
}
```

5. **Выводы**

   Данная лабораторная работа знакомит с простейшим многопоточным взаимодействием для выполнения некоторого алгоритма. Уверен, знания, приобретенные в процессе выполнения данной работы, например, взаимодействие потоков, работа с мьютексом, ожидание выполнения всех потоков, помогут мне в будущем.

## ЛИТЕРАТУРА


1. Многопоточность в Си [Электронный ресурс]. URL: http://espressocode.top/multithreading-c-2/