Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

Лабораторная работа №1 по курсу

«Операционные системы»

Группа: М8О-211Б-23

Студент: Воробьев Г. Я.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 18.11.24

Москва, 2024

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 максимального количества потоков, работающих в один момент времени, 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.

17. Найти в большом целочисленном массиве минимальный и максимальный элементы

Общий метод и алгоритм решения

Использованные системные вызовы:

- int pthread_create(pthread_t* restrict newthread, const pthread_attr_t* restrict_attr, void* (*start_routine)(void*), void* restrict arg) — создаёт поток с рутиной (стартовой функцией) и заданными аргументами.
- int pthread_join(pthread_t th, void** thread_return) — дожидается завершения потока.
- void exit(int status) - завершения выполнения процесса и возвращение статуса

Для mutex реализации были использованы:

- pthread_mutex_t – тип данных;
- int pthread_mutex_init(pthread_mutex_t *mutex, const pthread_mutexattr_t *mutexattr) – инициализация мьютекса;
- int pthread_mutex_lock(pthread_mutex_t *mutex) – блокировка мьютекса;
- int pthread_mutex_unlock(pthread_mutex_t *mutex) – разблокировка мьютекса;
- int pthread_mutex_destroy(pthread_mutex_t *mutex) – уничтожение мьютекса;

Программа выполняет многопоточный поиск минимального и максимального значений в массиве целых чисел. На вход подаются три аргумента: размер массива, количество потоков и сид для генерации случайных чисел. После проверки входных данных программа создаёт массив заданного размера, заполняя его случайными числами. Для распределения работы между потоками массив разбивается на равные части, а для каждого участка создаётся поток, которому передаётся диапазон индексов. Потоки выполняют функцию поиска локального минимума и максимума для своей части массива, измеряя время выполнения. Основной поток ожидает завершения всех потоков, после чего объединяет результаты, находя глобальные минимальное и максимальное значения. В ходе работы программа выводит время создания и выполнения потоков, а также общее время выполнения.

Результаты, включая глобальный минимум и максимум, выводятся в стандартный вывод. После завершения работы память, выделенная для массива, освобождается.

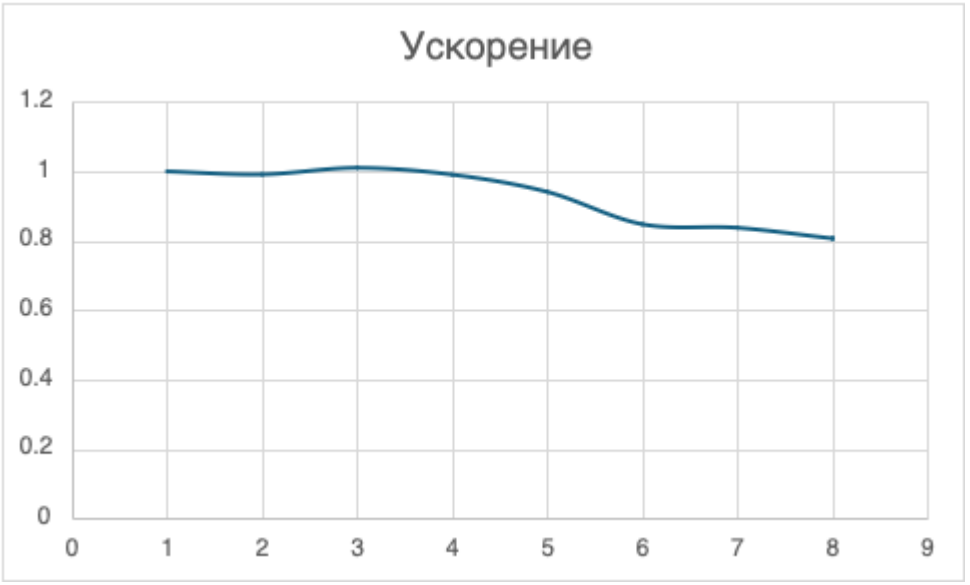
Ниже приведены данные, показывающие изменение ускорения и эффективности, с разным количеством потоков, для этой реализации.

Число потоков	Время выполнения, с	Ускорение	Эффективность
1	15.023467	1	1
2	15.152145	0.991507605	0.495754
3	14.869825	1.010332469	0.336777
4	15.163982	0.990733634	0.247683
5	15.966227	0.940952863	0.188191
6	17.719528	0.847848035	0.141308
7	17.938080	0.837518118	0.119645
8	18.621320	0.806788509	0.100849

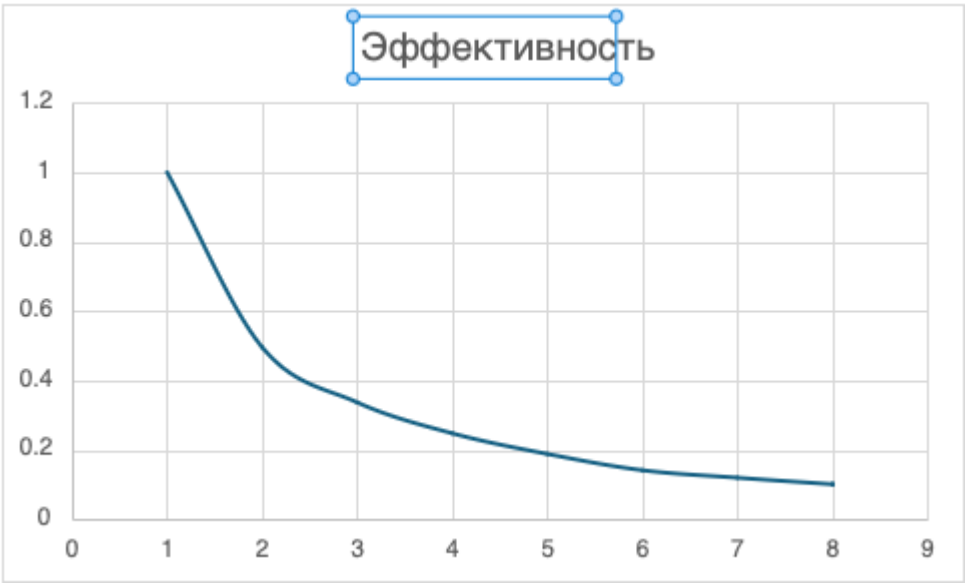
Тестирование на 3 потоках:

Размер массива	Время выполнения, с
10^8	0.943116
10^9	10.550963
10^10	14.869825
10^11	12.683320

Ускорение:



Эффективность:



Объяснение результатов:
Как видно из полученных результатов, задача поиска максимума и минимума в массиве не подходит для многопоточности, так как в этом случае программа замедляется. Это можно объяснить тем, что сама по себе задача поиска максимума и минимума элементарная и время на создание потоков только увеличивает время работы.

Код программы

main.c

```c
#include "../inc/core.h"

#include <time.h>

#include <pthread.h>



void print_array(int* arr);

void* find_min_max(void* arg);



// Мьютекс для синхронизации доступа к global_min и global_max

pthread_mutex_t mutex;



int global_min, global_max;
```

```
int main(int argc, char** argv)

{

   if (argc != 4) {

       const char msg[] = "Usage: %s <array_size> <max_threads> <seed>\n";

       write(STDERR_FILENO, msg, sizeof(msg));

       exit(EXIT_SUCCESS);

   }



   long arr_size =  atoi(argv[1]);

   int max_threads = atoi(argv[2]);

   //unsigned int seed = atoi(argv[3]);



   if (arr_size <= 0 || max_threads <= 0) {

       const char msg[] = "Error: Array size and max threads must be positive
integers.\n";

       write(STDERR_FILENO, msg, sizeof(msg));

       exit(EXIT_SUCCESS);

   }



   int* array = (int*) malloc(arr_size * sizeof(int));

   for (long i = 0; i < arr_size; i++)

       array[i] = rand() % 1000;



   pthread_t threads[max_threads];

   thread_data thread_data_arr[max_threads];



   int chunk_size = arr_size / max_threads + (arr_size % max_threads != 0);



   // Инициализация мьютекса

   if (pthread_mutex_init(&mutex, NULL) != 0) {
```

```
        perror("Failed to initialize mutex");
        exit(EXIT_FAILURE);
    }

    global_min = array[0];
    global_max = array[0];

    clock_t start_full, end_full, start_creation, end_creation;

    start_full = clock();
    for (int i = 0; i < max_threads; i++) {
        thread_data_arr[i].array = array;
        thread_data_arr[i].start = i * chunk_size;
        thread_data_arr[i].end = (i == max_threads - 1) ? arr_size : (i + 1) *
chunk_size;

        start_creation = clock();
        if (pthread_create(&threads[i], NULL, find_min_max, &thread_data_arr[i]) != 0) {
            perror("Failed to create thread");
            exit(EXIT_FAILURE);
        }
        end_creation = clock();

        printf("Creation time of %d: %f seconds\n", i, (double)(end_creation -
start_creation) / CLOCKS_PER_SEC);
    }

    clock_t start_join = clock();
    for (int i = 0; i < max_threads; i++) {
        if (pthread_join(threads[i], NULL) != 0) {
            perror("Failed to join thread");
```

```
            exit(EXIT_FAILURE);
        }
    }
    clock_t end_join = clock();

    end_full = clock();

    printf("Full time: %f seconds\n", (double)(end_full - start_full) / CLOCKS_PER_SEC);
    printf("Join time: %f seconds\n", (double)(end_join - start_join) / CLOCKS_PER_SEC);

    {
        char buff[INT_SIZE];
        int len = snprintf(buff, sizeof(buff), "Global Min: %d\n", global_min);
        write(STDOUT_FILENO, buff, len);
    }

    {
        char buff[INT_SIZE];
        int len = snprintf(buff, sizeof(buff), "Global Max: %d\n", global_max);
        write(STDOUT_FILENO, buff, len);
    }

    // Уничтожение мьютекса
    pthread_mutex_destroy(&mutex);

    free(array);
    return 0;
}

void* find_min_max(void* arg)
{
```

```
    clock_t start = clock();

    thread_data* data = (thread_data*) arg;

    int local_min = data->array[data->start];
    int local_max = data->array[data->start];

    for (int i = data->start; i < data->end; i++) {
        if (data->array[i] > local_max)
            local_max = data->array[i];

        if (data->array[i] < local_min)
            local_min = data->array[i];
    }

    // Защита глобальных переменных с помощью мьютекса
    pthread_mutex_lock(&mutex);
    if (local_min < global_min)
        global_min = local_min;
    if (local_max > global_max)
        global_max = local_max;
    pthread_mutex_unlock(&mutex);

    clock_t end = clock();

    printf("Function time: %f seconds\n", (double)(end - start) / CLOCKS_PER_SEC);

    return NULL;
}

void print_array(int* arr)
```

```
{
    const char msg[] = "Array:\n";
    write(STDOUT_FILENO, msg, sizeof(msg) - 1);

    while(*arr) {
        char buff[INT_SIZE + 1];
        int len = snprintf(buff, sizeof(buff), "%d\n", *arr);
        write(STDOUT_FILENO, buff, len);
        arr++;
    }
}
```

Протокол работы программы

Тестирование:

```
glebvorobev@Glebs-MacBook-Air LAB_2 % ./main 1000000000 3 18
Creation time of 0: 0.000863 seconds
Creation time of 1: 0.000057 seconds
Creation time of 2: 0.000029 seconds
Function time: 10.475455 seconds
Function time: 10.492419 seconds
Function time: 10.502128 seconds
Full time: 10.504352 seconds
Join time: 10.502172 seconds
Global Min: 0
Global Max: 999
```

Dtruss:

```
SYSCALL(args)            = return
main(1365,0x1f4947240) malloc: nano zone abandoned due to inability to reserve vm space.
munmap(0x102C0C000, 0x84000)             = 0 0
munmap(0x102C90000, 0x8000)              = 0 0
munmap(0x102C98000, 0x4000)              = 0 0
munmap(0x102C9C000, 0x4000)              = 0 0
munmap(0x102CA0000, 0x48000)             = 0 0
munmap(0x102CE8000, 0x4C000)             = 0 0
crossarch_trap(0x0, 0x0, 0x0)            = -1 Err#45
```

```
      open(".\0", 0x100000, 0x0)              = 3 0
      fcntl(0x3, 0x32, 0x16D267168)           = 0 0
      close(0x3)              = 0 0
      fsgetpath(0x16D267178, 0x400, 0x16D267158)              = 58 0
      fsgetpath(0x16D267188, 0x400, 0x16D267168)              = 14 0
      csrctl(0x0, 0x16D26758C, 0x4)           = -1 Err#1
      __mac_syscall(0x18FEFFC12, 0x2, 0x16D2674D0)            = 0 0
      csrctl(0x0, 0x16D26757C, 0x4)           = -1 Err#1
      __mac_syscall(0x18FEFCA45, 0x5A, 0x16D267510)           = 0 0
= 0 0 sysctl([unknown, 3, 0, 0, 0, 0] (2), 0x16D266A78, 0x16D266A70, 0x18FEFE738, 0xD)
= 0 0 sysctl([CTL_KERN, 150, 0, 0, 0, 0] (2), 0x16D266B28, 0x16D266B20, 0x0, 0x0)
      open("/\0", 0x20100000, 0x0)            = 3 0
      openat(0x3, "System/Cryptexes/OS\0", 0x100000, 0x0)             = 4 0
      dup(0x4, 0x0, 0x0)              = 5 0
      fstatat64(0x4, 0x16D266601, 0x16D266570)                = 0 0
      openat(0x4, "System/Library/dyld/\0", 0x100000, 0x0)            = 6 0
      fcntl(0x6, 0x32, 0x16D266600)           = 0 0
      dup(0x6, 0x0, 0x0)              = 7 0
      dup(0x5, 0x0, 0x0)              = 8 0
      close(0x3)              = 0 0
      close(0x5)              = 0 0
      close(0x4)              = 0 0
      close(0x6)              = 0 0
      __mac_syscall(0x18FEFFC12, 0x2, 0x16D266FF0)            = 0 0
      shared_region_check_np(0x16D266C10, 0x0, 0x0)           = 0 0
      fsgetpath(0x16D267190, 0x400, 0x16D2670B8)              = 82 0
      fcntl(0x8, 0x32, 0x16D267190)           = 0 0
      close(0x8)              = 0 0
      close(0x7)              = 0 0
      getfsstat64(0x0, 0x0, 0x2)              = 11 0
      getfsstat64(0x102B9C050, 0x5D28, 0x2)           = 11 0
      getattrlist("/\0", 0x16D2670D0, 0x16D267040)            = 0 0
      stat64("/System/Volumes/Preboot/Cryptexes/OS/System/Library/dyld/dyld_shared_cache_arm
      64e\0", 0x16D267430, 0x0)          = 0 0
      dtrace: error on enabled probe ID 1696 (ID 845: syscall::stat64:return): invalid
      address (0x0) in action #11 at DIF offset 12
      stat64("/Users/glebvorobev/Documents/Projects/OS/build/LAB_2/main\0", 0x16D2668E0
      , 0x0)          = 0 0
      open("/Users/glebvorobev/Documents/Projects/OS/build/LAB_2/main\0", 0x0, 0x0)
                = 3 0
      mmap(0x0, 0x11BA8, 0x1, 0x40002, 0x3, 0x0)              = 0x102B9C000 0
      fcntl(0x3, 0x32, 0x16D2669F8)           = 0 0
      close(0x3)              = 0 0
      munmap(0x102B9C000, 0x11BA8)            = 0 0
= 0 0 stat64("/Users/glebvorobev/Documents/Projects/OS/build/LAB_2/main\0", 0x16D266E50, 0x0)
      stat64("/usr/lib/libSystem.B.dylib\0", 0x16D265DA0, 0x0)                = -1 Err#2
      stat64("/System/Volumes/Preboot/Cryptexes/OS/usr/lib/libSystem.B.dylib\0",
      0x16D265D50, 0x0)             = -1 Err#2
```

```
open("@rpath/libclang_rt.asan_osx_dynamic.dylib\0", 0x0, 0x0)           = -1 Err#2
open("@rpath\0", 0x100000, 0x0)          = -1 Err#2
stat64("/Users/glebvorobev/Documents/Projects/OS/build/LAB_2/libclang_rt.asan_osx_dyna
mic.dylib\0", 0x16D265C70, 0x0)            = -1 Err#2
stat64("/Library/Developer/CommandLineTools/usr/lib/clang/16/lib/darwin/libclang_rt.as
an_osx_dynamic.dylib\0", 0x16D265C70, 0x0)                 = 0 0
stat64("/Library/Developer/CommandLineTools/usr/lib/clang/16/lib/darwin/libclang_rt.as
an_osx_dynamic.dylib\0", 0x16D2656A0, 0x0)                 = 0 0
open("/Library/Developer/CommandLineTools/usr/lib/clang/16/lib/darwin/libclang_rt.asan
_osx_dynamic.dylib\0", 0x0, 0x0)           = 3 0
mmap(0x0, 0x40B750, 0x1, 0x40002, 0x3, 0x0)              = 0x102BA0000 0
fcntl(0x3, 0x32, 0x16D2657B8)            = 0 0
close(0x3)               = 0 0
open("/Library/Developer/CommandLineTools/usr/lib/clang/16/lib/darwin/libclang_rt.asan
_osx_dynamic.dylib\0", 0x0, 0x0)           = 3 0
fstat64(0x3, 0x16D264E40, 0x0)           = 0 0
fcntl(0x3, 0x61, 0x16D265438)            = 0 0
fcntl(0x3, 0x62, 0x16D265438)            = 0 0
mmap(0x102FAC000, 0xA8000, 0x5, 0x40012, 0x3, 0x32C000)          = 0x102FAC000 0
mmap(0x103054000, 0x4000, 0x3, 0x40012, 0x3, 0x3D4000)           = 0x103054000 0
mmap(0x103058000, 0x4000, 0x3, 0x40012, 0x3, 0x3D8000)           = 0x103058000 0
mmap(0x103A40000, 0x30000, 0x1, 0x40012, 0x3, 0x3DC000)          = 0x103A40000 0
close(0x3)               = 0 0
munmap(0x102BA0000, 0x40B750)            = 0 0
open("/Users/glebvorobev/Documents/Projects/OS/build/LAB_2/main\0", 0x0, 0x0)
          = 3 0
__mac_syscall(0x18FEFFC12, 0x2, 0x16D264490)             = 0 0
map_with_linking_np(0x16D264260, 0x1, 0x16D264290)               = 0 0
close(0x3)               = 0 0
mprotect(0x102B90000, 0x4000, 0x1)               = 0 0
open("/Library/Developer/CommandLineTools/usr/lib/clang/16/lib/darwin/libclang_rt.asan
_osx_dynamic.dylib\0", 0x0, 0x0)           = 3 0
__mac_syscall(0x18FEFFC12, 0x2, 0x16D264490)             = 0 0
map_with_linking_np(0x16D263670, 0x1, 0x16D2636A0)               = 0 0
close(0x3)               = 0 0
mprotect(0x103054000, 0x4000, 0x1)               = 0 0
open("/dev/dtracehelper\0", 0x2, 0x0)            = 3 0
ioctl(0x3, 0x80086804, 0x16D2639D8)              = 0 0
close(0x3)               = 0 0
shared_region_check_np(0xFFFFFFFFFFFFFFFF, 0x0, 0x0)             = 0 0
access("/AppleInternal/XBS/.isChrooted\0", 0x0, 0x0)             = -1 Err#2
bsdthread_register(0x1902020F4, 0x1902020E8, 0x4000)             = 1073746399 0
getpid(0x0, 0x0, 0x0)            = 1365 0
shm_open(0x190099F41, 0x0, 0xFFFFFFFF9DC5E000)           = 3 0
fstat64(0x3, 0x16D2640A0, 0x0)           = 0 0
mmap(0x0, 0x8000, 0x1, 0x1, 0x3, 0x0)            = 0x102BA4000 0
close(0x3)               = 0 0
csops(0x555, 0x0, 0x16D2641DC)           = 0 0
sysctl([CTL_HW, 7, 0, 0, 0, 0] (2), 0x1F494ECF0, 0x16D264028, 0x0, 0x0)          = 0 0
```

```
mmap(0x0, 0x4000, 0x3, 0x1002, 0x63000000, 0x0)          = 0x102BAC000 0
stat64("/\0", 0x16D262360, 0x0)          = 0 0
getattrlist("/Users\0", 0x190116D20, 0x16D263C70)                = 0 0
getattrlist("/Users/glebvorobev\0", 0x190116D20, 0x16D263C70)            = 0 0
getattrlist("/Users/glebvorobev/Documents\0", 0x190116D20, 0x16D263C70)          = 0 0
getattrlist("/Users/glebvorobev/Documents/Projects\0", 0x190116D20, 0x16D263C70)
                 = 0 0
getattrlist("/Users/glebvorobev/Documents/Projects/OS\0", 0x190116D20, 0x16D263C70)
             = 0 0
getattrlist("/Users/glebvorobev/Documents/Projects/OS/build\0", 0x190116D20,
0x16D263C70)                = 0 0
getattrlist("/Users/glebvorobev/Documents/Projects/OS/build/LAB_2\0", 0x190116D20,
0x16D263C70)          = 0 0
getattrlist("/Users/glebvorobev/Documents/Projects/OS/build/LAB_2/main\0",
0x190116D20, 0x16D263C70)             = 0 0
munmap(0x102BAC000, 0x4000)              = 0 0
mmap(0x0, 0x4000, 0x3, 0x1002, 0x63000000, 0x0)          = 0x102BAC000 0
getattrlist("/Users\0", 0x190116D20, 0x16D263C70)                = 0 0
getattrlist("/Users/glebvorobev\0", 0x190116D20, 0x16D263C70)            = 0 0
getattrlist("/Users/glebvorobev/Documents\0", 0x190116D20, 0x16D263C70)          = 0 0
getattrlist("/Users/glebvorobev/Documents/Projects\0", 0x190116D20, 0x16D263C70)
              = 0 0
getattrlist("/Users/glebvorobev/Documents/Projects/OS\0", 0x190116D20, 0x16D263C70)
            = 0 0
getattrlist("/Users/glebvorobev/Documents/Projects/OS/build\0", 0x190116D20,
 0x16D263C70)                = 0 0
getattrlist("/Users/glebvorobev/Documents/Projects/OS/build/LAB_2\0", 0x190116D20,
0x16D263C70)          = 0 0
getattrlist("/Users/glebvorobev/Documents/Projects/OS/build/LAB_2/main\0",
 0x190116D20, 0x16D263C70)             = 0 0
munmap(0x102BAC000, 0x4000)              = 0 0
mmap(0x0, 0x4000, 0x3, 0x1002, 0x63000000, 0x0)          = 0x102BAC000 0
mmap(0x0, 0x4000, 0x3, 0x1002, 0x63000000, 0x0)          = 0x102BB0000 0
sysctl([unknown, 3, 0, 0, 0, 0] (2), 0x16D264068, 0x16D264060, 0x10304C34A, 0xE)
                 = 0 0
sysctl([CTL_KERN, 2, 0, 0, 0, 0] (2), 0x16D264100, 0x16D2640F8, 0x0, 0x0)
               = 0 0
mmap(0x0, 0x390000, 0x3, 0x1002, 0x63000000, 0x0)                = 0x102BB4000 0
mprotect(0x102F44000, 0x4000, 0x1)               = 0 0
getrlimit(0x1004, 0x16D264150, 0x0)              = 0 0
setrlimit(0x1004, 0x16D264150, 0x0)              = 0 0
sysctl([unknown, 3, 0, 0, 0, 0] (2), 0x16D263E48, 0x16D263E40, 0x10304C752, 0x15)
                = 0 0
sysctl([CTL_KERN, 145, 0, 0, 0, 0] (2), 0x16D263EE0, 0x16D263F28, 0x0, 0x0)
             = 0 0
mmap(0x700001C000, 0xE00008000, 0x3, 0x1052, 0x63000000, 0x0)            =
0x700001C000 0
mmap(0x27E00024000, 0xDF1FFFFC000, 0x3, 0x1052, 0x63000000, 0x0)                =
0x27E00024000 0
```

```
mmap(0x7E00024000, 0x20000000000, 0x0, 0x1052, 0x63000000, 0x0)          =
0x7E00024000 0
sigaltstack(0x0, 0x16D264120, 0x0)               = 0 0
mmap(0x0, 0x80000, 0x3, 0x1002, 0x63000000, 0x0)                 = 0x103A70000 0
sigaltstack(0x16D264138, 0x0, 0x0)               = 0 0
sigaction(0xB, 0x16D2640F8, 0x0)                 = 0 0
sigaction(0xA, 0x16D2640F8, 0x0)                 = 0 0
sigaction(0x8, 0x16D2640F8, 0x0)                 = 0 0
mmap(0x600000000000, 0x40000004000, 0x0, 0x1052, 0x63000000, 0x0)                =
 0x600000000000 0
mmap(0x640000000000, 0x4000, 0x3, 0x1012, 0x63000000, 0x0)               =
0x640000000000 0
mmap(0x0, 0x800000, 0x0, 0x1042, 0x63000000, 0x0)                = 0x103AF0000 0
mmap(0x0, 0x10000, 0x3, 0x1002, 0x63000000, 0x0)                 = 0x102F48000 0
mmap(0x0, 0x4000, 0x3, 0x1002, 0x63000000, 0x0)          = 0x102F58000 0
mmap(0x0, 0x4000, 0x3, 0x1002, 0x63000000, 0x0)          = 0x102F5C000 0
getrlimit(0x1003, 0x16D263FE0, 0x0)              = 0 0
mmap(0x702D970000, 0xFC000, 0x3, 0x1052, 0x63000000, 0x0)                =
0x702D970000 0
sigaltstack(0x0, 0x16D2640F0, 0x0)               = 0 0
mmap(0x0, 0x200000, 0x3, 0x1002, 0x63000000, 0x0)                = 0x1042F0000 0
munmap(0x1042F0000, 0x10000)             = 0 0
munmap(0x104400000, 0xF0000)             = 0 0
mmap(0x0, 0x4000, 0x3, 0x1002, 0x63000000, 0x0)          = 0x102F60000 0
__mac_syscall(0x19C167505, 0x2, 0x16D263ED0)             = 0 0
mmap(0x0, 0x4000, 0x3, 0x1002, 0x63000000, 0x0)          = 0x102F64000 0
__mac_syscall(0x19C167505, 0x2, 0x16D263ED0)             = 0 0
__mac_syscall(0x19C167505, 0x2, 0x16D263ED0)             = 0 0
__mac_syscall(0x19C167505, 0x2, 0x16D263ED0)             = 0 0
__mac_syscall(0x19C167505, 0x2, 0x16D263ED0)             = 0 0
__mac_syscall(0x19C167505, 0x2, 0x16D263ED0)             = 0 0
stat64("/usr/bin/atos\0", 0x16D263FD0, 0x0)              = 0 0
mmap(0x0, 0x200000, 0x3, 0x1002, 0x63000000, 0x0)                = 0x104400000 0
munmap(0x104500000, 0x100000)            = 0 0
mmap(0x0, 0x200000, 0x3, 0x1002, 0x63000000, 0x0)                = 0x104500000 0
munmap(0x104600000, 0x100000)            = 0 0
munmap(0x102F64000, 0x4000)              = 0 0
mmap(0x0, 0x4000, 0x3, 0x1002, 0x63000000, 0x0)          = 0x102F64000 0
mmap(0x0, 0x4000, 0x3, 0x1002, 0x63000000, 0x0)          = 0x102F68000 0
ioctl(0x2, 0x4004667A, 0x16D26414C)              = 0 0
mprotect(0x102F74000, 0x4000, 0x0)               = 0 0
mprotect(0x102F80000, 0x4000, 0x0)               = 0 0
mprotect(0x102F84000, 0x4000, 0x0)               = 0 0
mprotect(0x102F90000, 0x4000, 0x0)               = 0 0
mprotect(0x102F94000, 0x4000, 0x0)               = 0 0
mprotect(0x102FA0000, 0x4000, 0x0)               = 0 0
mprotect(0x102F6C000, 0xC8, 0x1)                 = 0 0
mprotect(0x102F6C000, 0xC8, 0x3)                 = 0 0
mprotect(0x102F6C000, 0xC8, 0x1)                 = 0 0
```

```
mprotect(0x102F44000, 0x4000, 0x3)               = 0 0
mprotect(0x102F44000, 0x4000, 0x1)               = 0 0
mprotect(0x102FA4000, 0xC8, 0x1)                 = 0 0
write(0x2, "main(1365,0x1f4947240) malloc: nano zone abandoned due to inability to
 reserve vm space.\n\0", 0x59)                 = 89 0
proc_info(0x2, 0x555, 0x11)              = 56 0
socket(0x1, 0x2, 0x0)            = 3 0
fcntl(0x3, 0x2, 0x1)             = 0 0
connect(0x3, 0x16D263106, 0x6A)          = 0 0
sendto(0x3, 0x16D2631F0, 0xDD)           = 221 0
issetugid(0x0, 0x0, 0x0)                 = 0 0
mmap(0x61B000000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x61B000000000 0
mmap(0x61BE00000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
 0x61BE00000000 0
mmap(0x0, 0x100000, 0x3, 0x1002, 0x63000000, 0x0)               = 0x104600000 0
mmap(0x0, 0x800000, 0x3, 0x1042, 0x63000000, 0x0)               = 0x104700000 0
mmap(0x606000000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x606000000000 0
mmap(0x606E00000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x606E00000000 0
getentropy(0x16D262F48, 0x20, 0x0)               = 0 0
mmap(0x603000000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x603000000000 0
mmap(0x603E00000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x603E00000000 0
mmap(0x602000000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x602000000000 0
mmap(0x602E00000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x602E00000000 0
mmap(0x624000000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x624000000000 0
mmap(0x624E00000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x624E00000000 0
mmap(0x604000000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x604000000000 0
mmap(0x604E00000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
 0x604E00000000 0
mmap(0x616000000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x616000000000 0
mmap(0x616E00000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
 0x616E00000000 0

mmap(0x608000000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
 0x608000000000 0
mmap(0x608E00000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
 0x608E00000000 0
mmap(0x613000000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
 0x613000000000 0
```

```
mmap(0x613E00000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
 0x613E00000000 0
mmap(0x612000000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
 0x612000000000 0
mmap(0x612E00000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x612E00000000 0
getattrlist("/Users/glebvorobev/Documents/Projects/OS/build/LAB_2/main\0",
 0x16D264040, 0x16D26405C)             = 0 0
access("/Users/glebvorobev/Documents/Projects/OS/build/LAB_2\0", 0x4, 0x0)
                = 0 0
open("/Users/glebvorobev/Documents/Projects/OS/build/LAB_2\0", 0x0, 0x0)
                = 4 0
fstat64(0x4, 0x612000000090, 0x0)                = 0 0
csrctl(0x0, 0x16D26422C, 0x4)            = 0 0
fcntl(0x4, 0x32, 0x16D263F28)            = 0 0
close(0x4)               = 0 0
mmap(0x60C000000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x60C000000000 0
mmap(0x60CE00000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x60CE00000000 0
open("/Users/glebvorobev/Documents/Projects/OS/build/LAB_2/Info.plist\0", 0x0, 0x0)
               = -1 Err#2
mmap(0x60A000000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x60A000000000 0
mmap(0x60AE00000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x60AE00000000 0
mmap(0x607000000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x607000000000 0
mmap(0x607E00000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x607E00000000 0
proc_info(0x2, 0x555, 0xD)               = 64 0
csops_audittoken(0x555, 0x10, 0x16D2642B0)               = 0 0
sysctl([unknown, 3, 0, 0, 0, 0] (2), 0x16D263D98, 0x16D263D90, 0x19392CD3A, 0x15)
= 0 0
sysctl([CTL_KERN, 148, 0, 0, 0, 0] (2), 0x16D264698, 0x16D264690, 0x0, 0x0)
= 0 0
mmap(0x0, 0x4000, 0x3, 0x1002, 0x63000000, 0x0)          = 0x102FA4000 0
mmap(0x0, 0x4000, 0x3, 0x1002, 0x63000000, 0x0)          = 0x102FA8000 0
mmap(0x0, 0xEE6B8000, 0x3, 0x1002, 0x63000000, 0x0)              = 0x300000000 0
mmap(0x103AF0000, 0x20000, 0x3, 0x1012, 0x63000000, 0x0)                 = 0x103AF0000
0
mmap(0x7060024000, 0x1DCD0000, 0x3, 0x1052, 0x63000000, 0x0)             =
0x7060024000 0
Creation time of 0: 0.000077 seconds
Creation time of 1: 0.000021 seconds
Creation time of 2: 0.000101 seconds
getrusage(0x0, 0x16D276F60, 0x0)                 = 0 0
getrusage(0x0, 0x16D276F60, 0x0)                 = 0 0
mmap(0x0, 0x10000, 0x3, 0x1002, 0x63000000, 0x0)                 = 0x1042F0000 0
bsdthread_create(0x102FFD818, 0x1042F0000, 0x16D2FF000)          = 1831858176 0
mmap(0x0, 0x4000, 0x3, 0x1002, 0x63000000, 0x0)          = 0x104F00000 0
```

```
thread_selfid(0x0, 0x0, 0x0)             = 13467 0
getrusage(0x0, 0x16D276F60, 0x0)                 = 0 0
fstat64(0x1, 0x16D276D70, 0x0)           = 0 0
mmap(0x702DA70000, 0xC000, 0x3, 0x1052, 0x63000000, 0x0)                 =
0x702DA70000 0
ioctl(0x1, 0x4004667A, 0x16D276DBC)              = 0 0
sigaltstack(0x0, 0x16D2FEF30, 0x0)               = 0 0
mmap(0x0, 0x80000, 0x3, 0x1002, 0x63000000, 0x0)                 = 0x104F04000 0
sigaltstack(0x16D2FEF48, 0x0, 0x0)               = 0 0
getrusage(0x0, 0x16D2FEC10, 0x0)                 = 0 0
mmap(0x621000000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x621000000000 0
mmap(0x621E00000000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x621E00000000 0
write_nocancel(0x1, "Creation time of 0: 0.000077 seconds\n\0", 0x25)            = 37 0
getrusage(0x0, 0x16D276F60, 0x0)                 = 0 0
mmap(0x0, 0x10000, 0x3, 0x1002, 0x63000000, 0x0)                 = 0x104F84000 0
bsdthread_create(0x102FFD818, 0x104F84000, 0x16D38B000)          = 1832431616 0
getrusage(0x0, 0x16D276F60, 0x0)                 = 0 0
thread_selfid(0x0, 0x0, 0x0)             = 13468 0
write_nocancel(0x1, "Creation time of 1: 0.000021 seconds\n\0", 0x25)            = 37 0
getrusage(0x0, 0x16D276F60, 0x0)                 = 0 0
mmap(0x702DA84000, 0xC000, 0x3, 0x1052, 0x63000000, 0x0)                 =
0x702DA84000 0
mmap(0x0, 0x10000, 0x3, 0x1002, 0x63000000, 0x0)                 = 0x104F94000 0
sigaltstack(0x0, 0x16D38AF30, 0x0)               = 0 0
mmap(0x0, 0x80000, 0x3, 0x1002, 0x63000000, 0x0)                 = 0x104FA4000 0
sigaltstack(0x16D38AF48, 0x0, 0x0)               = 0 0
getrusage(0x0, 0x16D38AC10, 0x0)                 = 0 0
bsdthread_create(0x102FFD818, 0x104F94000, 0x16D417000)          = 1833005056 0
getrusage(0x0, 0x16D276F60, 0x0)                 = 0 0
write_nocancel(0x1, "Creation time of 2: 0.000101 seconds\n\0", 0x25)            = 37 0
getrusage(0x0, 0x16D276F60, 0x0)                 = 0 0
thread_selfid(0x0, 0x0, 0x0)             = 13469 0
mmap(0x702DA94000, 0xC000, 0x3, 0x1052, 0x63000000, 0x0)                 = 0x702DA94000
0
sigaltstack(0x0, 0x16D416F30, 0x0)               = 0 0
mmap(0x0, 0x80000, 0x3, 0x1002, 0x63000000, 0x0)                 = 0x105024000 0
sigaltstack(0x16D416F48, 0x0, 0x0)               = 0 0
getrusage(0x0, 0x16D416C10, 0x0)                 = 0 0
Function time: 10.509831 seconds
Function time: 10.533672 seconds
Function time: 10.537916 seconds
Full time: 10.538492 seconds
Join time: 10.538008 seconds
Global Min: 0
Global Max: 999
getrusage(0x0, 0x16D2FEC10, 0x0)                 = 0 0
write_nocancel(0x1, "Function time: 10.509831 seconds\n\0", 0x21)               = 33 0
```

```
__disable_threadsignal(0x1, 0x0, 0x0)            = 0 0
mmap(0x624000010000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
0x624000010000 0
mmap(0x0, 0x4000, 0x3, 0x1002, 0x63000000, 0x0)          = 0x1050A4000 0
madvise(0x624000014000, 0xC000, 0x5)             = 0 0
munmap(0x1050A4000, 0x4000)              = 0 0
sigaltstack(0x16D2FEED8, 0x16D2FEEC0, 0x0)               = 0 0
munmap(0x104F04000, 0x80000)             = 0 0
mmap(0x702DA70000, 0xC000, 0x3, 0x1052, 0x63000000, 0x0)                  =
0x702DA70000 0
munmap(0x1042F0000, 0x10000)             = 0 0
ulock_wake(0x1000002, 0x16D2FF034, 0x0)          = 0 0
ulock_wait(0x1020002, 0x16D2FF034, 0x1D07)               = 0 0
getrusage(0x0, 0x16D416C10, 0x0)                 = 0 0
write_nocancel(0x1, "Function time: 10.533672 seconds\n\0", 0x21)                = 33 0
__disable_threadsignal(0x1, 0x0, 0x0)            = 0 0
mmap(0x624000020000, 0x10000, 0x3, 0x1012, 0x63000000, 0x0)              =
 0x624000020000 0
mmap(0x0, 0x4000, 0x3, 0x1002, 0x63000000, 0x0)          = 0x1042F0000 0
madvise(0x607000004000, 0x8000, 0x5)             = 0 0
munmap(0x1042F0000, 0x4000)              = 0 0
sigaltstack(0x16D416ED8, 0x16D416EC0, 0x0)               = 0 0
munmap(0x105024000, 0x80000)             = 0 0
mmap(0x702DA94000, 0xC000, 0x3, 0x1052, 0x63000000, 0x0)                  =
0x702DA94000 0
munmap(0x104F94000, 0x10000)             = 0 0
getrusage(0x0, 0x16D38AC10, 0x0)                 = 0 0
write_nocancel(0x1, "Function time: 10.537916 seconds\n\0", 0x21)                = 33 0
__disable_threadsignal(0x1, 0x0, 0x0)            = 0 0
mmap(0x0, 0x4000, 0x3, 0x1002, 0x63000000, 0x0)          = 0x1042F0000 0
madvise(0x603000004000, 0x8000, 0x5)             = 0 0
madvise(0x604000004000, 0xC000, 0x5)             = 0 0
munmap(0x1042F0000, 0x4000)              = 0 0
sigaltstack(0x16D38AED8, 0x16D38AEC0, 0x0)               = 0 0
munmap(0x104FA4000, 0x80000)             = 0 0
mmap(0x702DA84000, 0xC000, 0x3, 0x1052, 0x63000000, 0x0)                  =
0x702DA84000 0
munmap(0x104F84000, 0x10000)             = 0 0
ulock_wake(0x1000002, 0x16D38B034, 0x0)          = 0 0
ulock_wait(0x1020002, 0x16D38B034, 0xA0B)                = 0 0
getrusage(0x0, 0x16D276F60, 0x0)                 = 0 0
getrusage(0x0, 0x16D276F60, 0x0)                 = 0 0
write_nocancel(0x1, "Full time: 10.538492 seconds\n\0", 0x1D)            = 29 0
write_nocancel(0x1, "Join time: 10.538008 seconds\n\0", 0x1D)            = 29 0
write(0x1, "Global Min: 0\n\0", 0xE)             = 14 0
write(0x1, "Global Max: 999\n\0", 0x10)          = 16 0
mmap(0x7060020000, 0x1DCD4000, 0x3, 0x1052, 0x63000000, 0x0)             =
0x7060020000 0
madvise(0x7060020000, 0x1DCD4000, 0x5)           = 0 0
```

```
munmap(0x300000000, 0xEE6B8000)          = 0 0
```

Вывод

В ходе выполнения данной лабораторной работы я научился создавать программы на языке C, которые обрабатывают данные в многопоточном режиме, а также применять стандартные средства операционной системы для управления потоками и их синхронизации.

В результате экспериментов и тестирования программы, реализующей поиск минимума и максимума, я проанализировал, как входной размер данных и количество потоков влияют на ускорение и эффективность выполнения алгоритма.

Из полученных результатов я пришел к выводу, что задача поиска максимума и минимума в массиве не подходит для многопоточности, так как в этом случае программа замедляется. Это можно объяснить тем, что сама по себе задача поиска максимума и минимума элементарная и время на создание потоков только увеличивает время работы.