Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №2 по курсу

«Операционные системы»

Группа: М8О-209Б-22

Студент: Концебалов О.С.

Преподаватель: Пономарев Н.В.

Оценка: ____________________

Дата: 09.12.2023

Москва, 2023.

Содержание

Постановка задачи

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 от входных данных и количества потоков. Получившиеся результаты необходимо объяснить.

Общие сведения о программе

Программа состоит из 4 папок: include, src, analytics, data_files, generators. В папках analytics и generators находятся Python скрипты для построения графика исходя из полученных данных о времени работы потоков и генератор тестовых данных для задачи соответственно. В папке data_files находятся два файла: с метриками и непосредственно тестовыми данными. В папке include находится файл ParallelQuickSort.h – заголовочный файл моей быстрой сортировки. В папке src находится файл ParallelQuickSort.cpp с реализацией функций из заголовочного файла.

Также есть Makefile для удобной работы с программой.

Общий метод и алгоритм решения

На вход программы поступает количество потоков, которое должно использоваться. Исходя из этого, массив, отданный нам для сортировки, разделяется на некоторое количество частей, так чтобы каждому потоку досталось по одной части массива. После этого каждый поток выполняет быструю сортировку своей части, а затем полученные части объединяются в одну с помощью слияния. Так же для анализа скорости сортировки фиксируется реальное время, затраченное на сортировку и записывается в файл с метриками.

Код программы

./analytics/draw_graphic.py

```python
import matplotlib.pyplot as plt
import csv
import getpass

def main():
    username = getpass.getuser()

    X = []
    Y = []

    with open("/home/" + username + "/MAI_OS/2_Lab/data_files/metrics.txt", "r") as metrics_file :
        plotting = csv.reader(metrics_file, delimiter=' ')

        for ROWS in plotting:
            X.append(float(ROWS[0]))
            Y.append(float(ROWS[1]))

    plt.plot(X, Y)
    plt.title('Time of threads stat')
    plt.xlabel('Number of threads')
    plt.ylabel('Time [ms]')
    plt.grid()
    plt.show()


if __name__ == "__main__" :
    main()
```

./generators/generator.py

```
import random
import getpass

def main():
    number_of_data = 10000
    username = getpass.getuser()

    with open("/home/" + username +
"/MAI_OS/2_Lab/data_files/test_data.txt", "w+") as
test_data_file :
        for _ in range(number_of_data):
            test_data_file.write(
                                 str(random.randint(-(2**30),
2**30)) + " " +
                                 str(random.randint(-(2**30),
2**30)) + " " +
                                 str(random.randint(-(2**30),
2**30)) + "\n"
                                )


if __name__ == "__main__" :
    main()
```

./include/ParallelQuickSort.h

```
#include <chrono>
#include <fstream>
#include <iostream>
#include <pthread.h>
#include <vector>
#include <string>

namespace parallel_sort {

struct ThreadArgs {
    std::vector<int64_t>& _vec;
    int64_t _left;
    int64_t _right;
};

class ParallelQuickSort final {
public:
    static void parallel_quick_sort(std::vector<int64_t>&,
const uint8_t);
    static void fill_vector_from_file(std::vector<int64_t>&,
const std::string&);

private:
    [[nodiscard]] static int64_t
partition(std::vector<int64_t>&, const int64_t, const int64_t)
noexcept;
    static void quick_sort(std::vector<int64_t>&, const
int64_t, const int64_t) noexcept;
    static void* thread_quick_sort(void*) noexcept;

    template <class T>
    static void swap(T&, T&) noexcept;

    static void save_metrics(const std::string&, const auto&,
const uint8_t) noexcept;
};

}; // namespace parallel_sort
```

./src/ParallelQuickSort.cpp

```
#include "../include/ParallelQuickSort.h"

using namespace parallel_sort;

void ParallelQuickSort::parallel_quick_sort(
                                            std::vector<int64_t
>& vector,
                                            const uint8_t
threads_num
```

```
                                           )
{
    if (threads_num < 1) {
        throw std::invalid_argument("Threads num must be mpre
than zero");
    }

    std::vector<pthread_t> threads(threads_num);
    std::vector<ThreadArgs> thread_args;

    auto start_time = std::chrono::steady_clock::now();

    for (int64_t i = 0; i != threads_num; ++i) {
        int64_t left = i * (vector.size() / threads_num);
        int64_t right = (i == threads_num - 1) ? vector.size()
- 1 : ((i + 1) * (vector.size() / threads_num)) - 1;

        ThreadArgs args { vector, left, right };
        thread_args.push_back(args);
    }

    for (int64_t i = 0; i != threads_num; ++i) {
        pthread_create(&threads[i], NULL, thread_quick_sort,
&thread_args[i]);
    }

    for (int64_t i = 0; i != threads_num; ++i) {
        pthread_join(threads[i], NULL);
    }

    quick_sort(vector, 0, vector.size() - 1);

    auto end_time = std::chrono::steady_clock::now();

    std::string file_name = "data_files/metrics.txt";
    auto spent_time =
std::chrono::duration_cast<std::chrono::milliseconds>(end_time
- start_time).count();

    save_metrics(file_name, spent_time, threads_num);
```

```
}

void ParallelQuickSort::fill_vector_from_file(
                                              std::vector<int64_t>& vector,
                                              const std::string& file_name
                                             )
{
    std::ifstream data_file(file_name);

    if (!data_file.is_open()) {
        throw std::runtime_error("Failed with opening file");
    }

    int64_t number;
    while (!data_file.eof()) {
        data_file >> number;
        vector.push_back(number);
    }
}

int64_t ParallelQuickSort::partition(
                                     std::vector<int64_t>& vector,
                                     const int64_t left,
                                     const int64_t right
                                    ) noexcept
{
    int64_t pivot = vector[right];
    int64_t i = (left - 1);

    for (int64_t j = left; j <= right - 1; ++j) {
        if (vector[j] <= pivot) {
            ++i;
            swap(vector[i], vector[j]);
        }
    }

    swap(vector[i + 1], vector[right]);
    return (i + 1);
}

void ParallelQuickSort::quick_sort(
                                   std::vector<int64_t>&
vector,
                                   const int64_t left,
                                   const int64_t right
                                  ) noexcept
{
    if (left < right) {
        int64_t partition_i = partition(vector, left, right);

        quick_sort(vector, left, partition_i - 1);
        quick_sort(vector, partition_i + 1, right);
    }
}

void* ParallelQuickSort::thread_quick_sort(void* args) noexcept
{
    ThreadArgs* thread_args = static_cast<ThreadArgs*>(args);

    std::vector<int64_t>& vector = thread_args->_vec;
    int64_t left = thread_args->_left;
    int64_t right = thread_args->_right;

    quick_sort(vector, left, right);
    return NULL;
}

template <class T>
void ParallelQuickSort::swap(T& first, T& second) noexcept {
    T tmp = first;
    first = second;
    second = tmp;
}

void ParallelQuickSort::save_metrics(
                                     const std::string&
file_name,
                                     const auto& spent_time,
                                     const uint8_t threads_num
                                    ) noexcept
{
    std::ofstream metrics_file(file_name, std::ios::app);
    metrics_file << static_cast<int>(threads_num) << ' ' <<
spent_time << std::endl;
}
```

./run.cpp

```
#include "include/ParallelQuickSort.h"

int main(int argc, char** argv){
    if (argc < 2) {
        std::cerr << "Usage" << argv[0] << "<threads_num>" <<
std::endl;
    }

    int64_t threads_num = std::stoi(argv[1]);
    std::string test_data_file_name =
"data_files/test_data.txt";
    std::vector<int64_t> vector;

    parallel_sort::ParallelQuickSort::fill_vector_from_file(vec
tor, test_data_file_name);
    parallel_sort::ParallelQuickSort::parallel_quick_sort(vecto
r, threads_num);

    std::cout << "Sorted arr: ";
    for (size_t i = 0; i != 10; ++i){
        std::cout << vector[i] << " ";
    }
    std::cout << std::endl << "Size: " << vector.size() <<
std::endl;

    return 0;
}
```

Использование утилиты strace

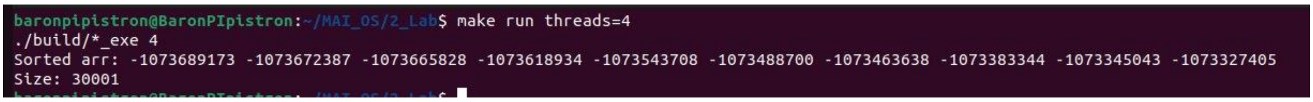
```
strace -f ./build/*_exe 5

execve("./build/Lab2_exe", ["./build/Lab2_exe", "5"], 0x7ffced5c45b0 /* 62 vars */) = 0

brk(NULL)                               = 0x5641eff83000

arch_prctl(0x3001 /* ARCH_??? */, 0x7ffe57554c20) = -1 EINVAL (Invalid argument)

mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ff94ab67000

access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)

openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=66003, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 66003, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7ff94ab56000

close(3)                                = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libstdc++.so.6", O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=2260296, ...}, AT_EMPTY_PATH) = 0

mmap(NULL, 2275520, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7ff94a800000

mprotect(0x7ff94a89a000, 1576960, PROT_NONE) = 0

mmap(0x7ff94a89a000, 1118208, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x9a000) = 0x7ff94a89a000

mmap(0x7ff94a9ab000, 454656, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1ab000) = 0x7ff94a9ab000
mmap(0x7ff94aa1b000, 57344, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x21a000) =
0x7ff94aa1b000

mmap(0x7ff94aa29000, 10432, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7ff94aa29000

close(3)                                = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libgcc_s.so.1",
O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) =
832

newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=125488, ...},
AT_EMPTY_PATH) = 0

mmap(NULL, 127720, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3,
0) = 0x7ff94ab36000

mmap(0x7ff94ab39000, 94208, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) =
0x7ff94ab39000

mmap(0x7ff94ab50000, 16384, PROT_READ,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1a000) =
0x7ff94ab50000

mmap(0x7ff94ab54000, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1d000) =
0x7ff94ab54000

close(3)                                = 0

openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6",
O_RDONLY|O_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\237\2\0\0\0\0\0"..., 832)
= 832

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"...,
784, 64) = 784

pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0\0"..., 48,
848) = 48

pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0
=\340\2563\265?\356\25x\261\27\313A#\350"..., 68, 896) = 68
```

```
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2216304, ...}, AT_EMPTY_PATH) = 0
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
mmap(NULL, 2260560, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7ff94a400000
mmap(0x7ff94a428000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7ff94a428000
mmap(0x7ff94a5bd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) = 0x7ff94a5bd000
mmap(0x7ff94a615000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x214000) = 0x7ff94a615000
mmap(0x7ff94a61b000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7ff94a61b000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libm.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=940560, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 942344, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7ff94aa4f000
mmap(0x7ff94aa5d000, 507904, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe000) = 0x7ff94aa5d000
mmap(0x7ff94aad9000, 372736, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x8a000) = 0x7ff94aad9000
mmap(0x7ff94ab34000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe4000) = 0x7ff94ab34000
```

```
close(3)                                = 0

mmap(NULL, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ff94aa4d000

arch_prctl(ARCH_SET_FS, 0x7ff94aa4e3c0) = 0

set_tid_address(0x7ff94aa4e690)         = 27577

set_robust_list(0x7ff94aa4e6a0, 24)     = 0

rseq(0x7ff94aa4ed60, 0x20, 0, 0x53053053) = 0

mprotect(0x7ff94a615000, 16384, PROT_READ) = 0

mprotect(0x7ff94ab34000, 4096, PROT_READ) = 0

mprotect(0x7ff94ab54000, 4096, PROT_READ) = 0

mmap(NULL, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ff94aa4b000

mprotect(0x7ff94aa1b000, 45056, PROT_READ) = 0

mprotect(0x5641ef9f0000, 4096, PROT_READ) = 0

mprotect(0x7ff94aba1000, 8192, PROT_READ) = 0

prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024,
rlim_max=RLIM64_INFINITY}) = 0

munmap(0x7ff94ab56000, 66003)           = 0

getrandom("\xc9\xcf\xa1\x76\xce\x99\x16\x57", 8, GRND_NONBLOCK) = 8

brk(NULL)                               = 0x5641eff83000

brk(0x5641effa4000)                     = 0x5641effa4000

futex(0x7ff94aa2977c, FUTEX_WAKE_PRIVATE, 2147483647) = 0

openat(AT_FDCWD, "data_files/test_data.txt", O_RDONLY) = 3

read(3, "-692341236 763343610 255380542\n8"..., 8191) = 8191

read(3, "62011\n91530661 -718790895 627331"..., 8191) = 8191

read(3, "260496414\n79383051 -159346961 59"..., 8191) = 8191

brk(0x5641effc8000)                     = 0x5641effc8000

read(3, " -762735654 -389754819\n596634860"..., 8191) = 8191

read(3, "46891\n1066298355 865520765 -1021"..., 8191) = 8191

read(3, "522859 379625262 1022474406\n9605"..., 8191) = 8191
read(3, "172 858645906 -673200857\n-541985"..., 8191) = 8191

read(3, "8443 686420068 127346613\n-430528"..., 8191) = 8191

read(3, "01923 752725321\n-625511622 49737"..., 8191) = 8191

read(3, " -321252255 459373756\n42884021 5"..., 8191) = 8191

read(3, "955639\n-158271838 253285459 -510"..., 8191) = 8191

mmap(NULL, 135168, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ff94a7df000

brk(0x5641effb8000)                     = 0x5641effb8000

read(3, "0597513 460929062 166155230\n5511"..., 8191) = 8191

read(3, "07542 -1022882217 423960680\n-866"..., 8191) = 8191

read(3, "3785361\n325664568 -241510882 749"..., 8191) = 8191

read(3, "0 89881889\n-156578463 -589213808"..., 8191) = 8191

read(3, "10060254 836014959 876540385\n-87"..., 8191) = 8191

read(3, "0430761\n-824811093 564273547 170"..., 8191) = 8191

read(3, "-904820235 558633827\n4397060 -54"..., 8191) = 8191

read(3, " 33783\n-308019348 -49972070 1809"..., 8191) = 8191

read(3, " -215057244\n-902699246 590016606"..., 8191) = 8191

read(3, "626038 -964302344\n-994731162 726"..., 8191) = 8191

mmap(NULL, 266240, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7ff94a79e000

munmap(0x7ff94a7df000, 135168)          = 0

read(3, "1 -1030194914 -989085512\n-437937"..., 8191) = 8191

read(3, "172\n969474459 -717332997 1045406"..., 8191) = 8191

read(3, "208331\n420504994 551782247 74452"..., 8191) = 8191

read(3, "2 -981784337 307162932\n692051980"..., 8191) = 8191

read(3, "580602259\n-316948654 -596509921 "..., 8191) = 8191

read(3, "756203690\n690041193 508371465 -1"..., 8191) = 8191

read(3, "995 -1022750896 818898137\n314055"..., 8191) = 8191

read(3, "6385\n-729213255 812472263 173332"..., 8191) = 8191
read(3, "00432694 -69824814\n434801799 770"..., 8191) = 8191

read(3, "27\n-133375170 -417464868 7770346"..., 8191) = 8191

read(3, "2498641\n-388378316 -711731991 -7"..., 8191) = 8191

read(3, "2827334 591539109\n-1067347929 -2"..., 8191) = 8191

read(3, "0230928\n-551122242 -224067438 -5"..., 8191) = 8191

read(3, "1 897723005 -6618273\n478611806 4"..., 8191) = 8191

read(3, "69663 -216784586 -402593351\n-663"..., 8191) = 8191

read(3, "10338 -641518895 900922097\n23996"..., 8191) = 8191

read(3, "68019\n-812872780 948666207 59442"..., 8191) = 8191

read(3, "96910322 933943114 25820787\n6540"..., 8191) = 2696

read(3, "", 8191)                       = 0

close(3)                                = 0

rt_sigaction(SIGRT_1, {sa_handler=0x7ff94a491870, sa_mask=[],
sa_flags=SA_RESTORER|SA_ONSTACK|SA_RESTART|SA_SIGINFO,
sa_restorer=0x7ff94a442520}, NULL, 8) = 0

rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0

mmap(NULL, 8392704, PROT_NONE,
MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) = 0x7ff949bff000

mprotect(0x7ff949c00000, 8388608, PROT_READ|PROT_WRITE) = 0

rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CL
ONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SET
TID|CLONE_CHILD_CLEARTID, child_tid=0x7ff94a3ff910,
parent_tid=0x7ff94a3ff910, exit_signal=0, stack=0x7ff949bff000,
stack_size=0x7fff00, tls=0x7ff94a3ff640}strace: Process 27578 attached

 => {parent_tid=[27578]}, 88) = 27578

[pid 27578] rseq(0x7ff94a3fffe0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 27577] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

[pid 27578] <... rseq resumed>)         = 0

[pid 27577] <... rt_sigprocmask resumed>NULL, 8) = 0

[pid 27578] set_robust_list(0x7ff94a3ff920, 24 <unfinished ...>
[pid 27577] mmap(NULL, 8392704, PROT_NONE,
MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0 <unfinished ...>

[pid 27578] <... set_robust_list resumed>) = 0

[pid 27577] <... mmap resumed>)        = 0x7ff9493fe000

[pid 27578] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

[pid 27577] mprotect(0x7ff9493ff000, 8388608, PROT_READ|PROT_WRITE
<unfinished ...>

[pid 27578] <... rt_sigprocmask resumed>NULL, 8) = 0

[pid 27577] <... mprotect resumed>)     = 0

[pid 27577] rt_sigprocmask(SIG_BLOCK, ~[], [], 8) = 0

[pid 27577]
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CL
ONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SET
TID|CLONE_CHILD_CLEARTID, child_tid=0x7ff949bfe910,
parent_tid=0x7ff949bfe910, exit_signal=0, stack=0x7ff9493fe000,
stack_size=0x7fff00, tls=0x7ff949bfe640}strace: Process 27579 attached

 => {parent_tid=[27579]}, 88) = 27579

[pid 27579] rseq(0x7ff949bfefe0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 27577] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

[pid 27579] <... rseq resumed>)        = 0

[pid 27577] <... rt_sigprocmask resumed>NULL, 8) = 0

[pid 27579] set_robust_list(0x7ff949bfe920, 24 <unfinished ...>

[pid 27577] mmap(NULL, 8392704, PROT_NONE,
MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0 <unfinished ...>

[pid 27579] <... set_robust_list resumed>) = 0

[pid 27578] rt_sigprocmask(SIG_BLOCK, ~[RT_1],  <unfinished ...>

[pid 27577] <... mmap resumed>)        = 0x7ff948bfd000

[pid 27579] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

[pid 27578] <... rt_sigprocmask resumed>NULL, 8) = 0

[pid 27577] mprotect(0x7ff948bfe000, 8388608, PROT_READ|PROT_WRITE
<unfinished ...>
[pid 27579] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid 27578] madvise(0x7ff949bff000, 8368128, MADV_DONTNEED <unfinished ...>
[pid 27577] <... mprotect resumed>)     = 0
[pid 27577] rt_sigprocmask(SIG_BLOCK, ~[],  <unfinished ...>
[pid 27578] <... madvise resumed>)      = 0
[pid 27577] <... rt_sigprocmask resumed>[], 8) = 0
[pid 27578] exit(0 <unfinished ...>
[pid 27577] clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7ff9493fd910, parent_tid=0x7ff9493fd910, exit_signal=0, stack=0x7ff948bfd000, stack_size=0x7fff00, tls=0x7ff9493fd640} <unfinished ...>
[pid 27578] <... exit resumed>)         = ?
strace: Process 27580 attached
[pid 27578] +++ exited with 0 +++
[pid 27577] <... clone3 resumed> => {parent_tid=[27580]}, 88) = 27580
[pid 27580] rseq(0x7ff9493fdfe0, 0x20, 0, 0x53053053 <unfinished ...>
[pid 27577] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid 27580] <... rseq resumed>)         = 0
[pid 27579] rt_sigprocmask(SIG_BLOCK, ~[RT_1],  <unfinished ...>
[pid 27577] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid 27580] set_robust_list(0x7ff9493fd920, 24 <unfinished ...>
[pid 27577] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0 <unfinished ...>
[pid 27580] <... set_robust_list resumed>) = 0
[pid 27579] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid 27577] <... mmap resumed>)         = 0x7ff9483fc000
[pid 27580] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid 27579] madvise(0x7ff9493fe000, 8368128, MADV_DONTNEED
<unfinished ...>

[pid 27577] mprotect(0x7ff9483fd000, 8388608, PROT_READ|PROT_WRITE) =
0

[pid 27580] <... rt_sigprocmask resumed>NULL, 8) = 0

[pid 27579] <... madvise resumed>)      = 0

[pid 27577] rt_sigprocmask(SIG_BLOCK, ~[],  <unfinished ...>

[pid 27579] exit(0 <unfinished ...>

[pid 27577] <... rt_sigprocmask resumed>[], 8) = 0

[pid 27579] <... exit resumed>)         = ?

[pid 27577]
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CL
ONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SET
TID|CLONE_CHILD_CLEARTID, child_tid=0x7ff948bfc910,
parent_tid=0x7ff948bfc910, exit_signal=0, stack=0x7ff9483fc000,
stack_size=0x7fff00, tls=0x7ff948bfc640} <unfinished ...>

[pid 27579] +++ exited with 0 +++

strace: Process 27581 attached

[pid 27577] <... clone3 resumed> => {parent_tid=[27581]}, 88) = 27581

[pid 27581] rseq(0x7ff948bfcfe0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 27577] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

[pid 27581] <... rseq resumed>)         = 0

[pid 27577] <... rt_sigprocmask resumed>NULL, 8) = 0

[pid 27581] set_robust_list(0x7ff948bfc920, 24 <unfinished ...>

[pid 27577] mmap(NULL, 8392704, PROT_NONE,
MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0 <unfinished ...>

[pid 27581] <... set_robust_list resumed>) = 0

[pid 27581] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

[pid 27577] <... mmap resumed>)         = 0x7ff947bfb000

[pid 27581] <... rt_sigprocmask resumed>NULL, 8) = 0

[pid 27580] rt_sigprocmask(SIG_BLOCK, ~[RT_1],  <unfinished ...>
[pid 27577] mprotect(0x7ff947bfc000, 8388608, PROT_READ|PROT_WRITE <unfinished ...>

[pid 27580] <... rt_sigprocmask resumed>NULL, 8) = 0

[pid 27577] <... mprotect resumed>)     = 0

[pid 27577] rt_sigprocmask(SIG_BLOCK, ~[],  <unfinished ...>

[pid 27580] madvise(0x7ff948bfd000, 8368128, MADV_DONTNEED <unfinished ...>

[pid 27577] <... rt_sigprocmask resumed>[], 8) = 0

[pid 27580] <... madvise resumed>)      = 0

[pid 27577] clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7ff9483fb910, parent_tid=0x7ff9483fb910, exit_signal=0, stack=0x7ff947bfb000, stack_size=0x7fff00, tls=0x7ff9483fb640} <unfinished ...>

[pid 27580] exit(0)                     = ?

strace: Process 27582 attached

[pid 27577] <... clone3 resumed> => {parent_tid=[27582]}, 88) = 27582

[pid 27580] +++ exited with 0 +++

[pid 27582] rseq(0x7ff9483fbfe0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 27577] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

[pid 27582] <... rseq resumed>)         = 0

[pid 27577] <... rt_sigprocmask resumed>NULL, 8) = 0

[pid 27577] futex(0x7ff948bfc910, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 27581, NULL, FUTEX_BITSET_MATCH_ANY <unfinished ...>

[pid 27582] set_robust_list(0x7ff9483fb920, 24 <unfinished ...>

[pid 27581] rt_sigprocmask(SIG_BLOCK, ~[RT_1],  <unfinished ...>

[pid 27582] <... set_robust_list resumed>) = 0

[pid 27581] <... rt_sigprocmask resumed>NULL, 8) = 0

[pid 27582] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>
[pid 27581] madvise(0x7ff9483fc000, 8368128, MADV_DONTNEED
<unfinished ...>

[pid 27582] <... rt_sigprocmask resumed>NULL, 8) = 0

[pid 27581] <... madvise resumed>)      = 0

[pid 27581] exit(0)                     = ?

[pid 27577] <... futex resumed>)        = 0

[pid 27581] +++ exited with 0 +++

[pid 27577] futex(0x7ff9483fb910,
FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 27582, NULL,
FUTEX_BITSET_MATCH_ANY <unfinished ...>

[pid 27582] rt_sigprocmask(SIG_BLOCK, ~[RT_1], NULL, 8) = 0

[pid 27582] madvise(0x7ff947bfb000, 8368128, MADV_DONTNEED) = 0

[pid 27582] exit(0)                     = ?

[pid 27577] <... futex resumed>)        = 0

[pid 27582] +++ exited with 0 +++

munmap(0x7ff949bff000, 8392704)         = 0

futex(0x7ff94aa29788, FUTEX_WAKE_PRIVATE, 2147483647) = 0

openat(AT_FDCWD, "data_files/metrics.txt",
O_WRONLY|O_CREAT|O_APPEND, 0666) = 3

lseek(3, 0, SEEK_END)                   = 105

write(3, "5 695\n", 6)                  = 6

close(3)                                = 0

newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...},
AT_EMPTY_PATH) = 0

write(1, "Sorted arr: -1073689173 -1073672"..., 133Sorted arr: -1073689173 -
1073672387 -1073665828 -1073618934 -1073543708 -1073488700 -1073463638
-1073383344 -1073345043 -1073327405

) = 133

write(1, "Size: 30001\n", 12Size: 30001

)         = 12

munmap(0x7ff94a79e000, 266240)          = 0
```

```
exit_group(0)                           = ?
+++ exited with 0 +++
```

Демонстрация работы программы



```
baronpipistron@BaronPIpistron:~/MAI_OS/2_Lab$ make run threads=4
./build/*_exe 4
Sorted arr: -1073689173 -1073672387 -1073665828 -1073618934 -1073543708 -1073488700 -1073463638 -1073383344 -1073345043 -1073327405
Size: 30001
baronpipistron@BaronPIpistron:~/MAI_OS/2_Lab$
```

Вывод

Во время лабораторной работы возникали трудности в процессе синхронизации потоков. Сначала пытался придумать какую-то сложную формулу для дробления массива и отдачи частей потоку, но все оказалось гораздо проще. В целом выполнять работу было куда легче и приятнее, чем первую, потому что был знакомый алгоритм быстрой сортировки, а работы с потоками на мой взгляд легче и приятнее, чем работа с процессами. Лаба довольно интересная, очень хорошее погружение в многопоточку, которая всегда довольно актуальна и ей получится найти применение.