Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

**Лабораторная работа №2 по курсу**

**«Операционные системы»**

Группа: М80-206Б-22

Студент: Сарайкин Н.С.

Преподаватель: Миронов Е.С.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: 15.02.2024

Москва, 2024

**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.

**Вариант 17.**

Найти в большом целочисленном массиве минимальный и максимальный элементы

**Общий метод и алгоритм решения**

Использованные системные вызовы:

* **pthread\_create**(&th\_id, NULL, func, &args); – создает новый поток c ID th\_id, который будет выполнять функцию func c с аргументами args.
* **pthread\_join**(th\_id, NULL) - ждет завершения потока th\_id.
* **pthread\_exit**(NULL); - завершает вызывающий поток.

Программа заполняет массив array случайными значениями, затем пользователь вводит количество потоков через аргумент командной строки. Далее программа создает указанное количество потоков и каждый поток находит минимальное и максимальное значение в своем диапазоне элементов массива array. Результаты минимальных и максимальных значений каждого потока сохраняются в массивах minresult и maxresult соответственно. С помощью pthread\_join создаем “барьер”, который будет ожидать завершения работы каждого потока. Далее находится общее минимальное и максимальное значение из результатов потоков. В конце программа выводит время выполнения, минимальный и максимальный элементы массива.

**Код программы**

**main.c**

#include <stdio.h>

#include <pthread.h> // pthread\_create, pthread\_join, pthread\_exit

#include <stdlib.h> // atoi()

#include <time.h>

#define ARRAY\_SIZE 100000000

int thread\_count = 0;

int array[ARRAY\_SIZE];

int min\_result[ARRAY\_SIZE];

int max\_result[ARRAY\_SIZE];

void\* find\_min\_max(void\* thread\_id) {

int id = \*(int\*)thread\_id;

int start = id \* (ARRAY\_SIZE / thread\_count);

int end = (id + 1) \* (ARRAY\_SIZE / thread\_count);

int min = array[start];

int max = array[start];

for (int i = start; i < end; i++) {

if (array[i] < min) min = array[i];

if (array[i] > max) max = array[i];

}

min\_result[id] = min;

max\_result[id] = max;

pthread\_exit(NULL);

}

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

// Заполнение массива случайными значениями

// srand(time(NULL));

for (int i = 0; i < ARRAY\_SIZE; i++) {

array[i] = rand();

// printf("%d\n", array[i]);

}

if (argc < 2) {

printf("Error: At least one argument is required.\n");

return 1;

}

thread\_count = atoi(argv[1]);

pthread\_t threads[thread\_count];

int thread\_ids[thread\_count];

clock\_t start = clock();

for (int i = 0; i < thread\_count; i++) {

thread\_ids[i] = i;

pthread\_create(&threads[i], NULL, find\_min\_max, (void\*)&thread\_ids[i]);

}

for (int i = 0; i < thread\_count; i++) {

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

}

int min = min\_result[0];

int max = max\_result[0];

for (int i = 1; i < thread\_count; i++) {

if (min\_result[i] < min) min = min\_result[i];

if (max\_result[i] > max) max = max\_result[i];

}

clock\_t end = clock();

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

printf("Время выполнения: %.5f сек\n", time);

printf("Минимальный элемент: %d\n", min);

printf("Максимальный элемент: %d\n", max);

return 0;

}

**Протокол работы программы**

**Тестирование:**

mattrrixwsl@DESKTOP-HRTTO4C:/mnt/c/Users/Никита/Desktop/Projects/MAI8fac\_OS/lab2/pr

ograms$ ./main 5

Время выполнения: 0.09513 сек

Минимальный элемент: 7

Максимальный элемент: 2147483611

===============================================================================

mattrrixwsl@DESKTOP-HRTTO4C:/mnt/c/Users/Никита/Desktop/Projects/MAI8fac\_OS/lab2/prog

rams$ ./main 2

Время выполнения: 0.07783 сек

Минимальный элемент: 7

Максимальный элемент: 2147483611

===============================================================================

mattrrixwsl@DESKTOP-HRTTO4C:/mnt/c/Users/Никита/Desktop/Projects/MAI8fac\_OS/lab2/prog

rams$ ./main 3

Время выполнения: 0.07422 сек

Минимальный элемент: 7

Максимальный элемент: 2147483611

===============================================================================

mattrrixwsl@DESKTOP-HRTTO4C:/mnt/c/Users/Никита/Desktop/Projects/MAI8fac\_OS/lab2/prog

rams$ ./main 4

Время выполнения: 0.08138 сек

Минимальный элемент: 7

Максимальный элемент: 2147483611

===============================================================================

mattrrixwsl@DESKTOP-HRTTO4C:/mnt/c/Users/Никита/Desktop/Projects/MAI8fac\_OS/lab2/prog

rams$ ./main 5

Время выполнения: 0.09045 сек

Минимальный элемент: 7

Максимальный элемент: 2147483611

===============================================================================

mattrrixwsl@DESKTOP-HRTTO4C:/mnt/c/Users/Никита/Desktop/Projects/MAI8fac\_OS/lab2/prog

rams$ ./main 6

Время выполнения: 0.09726 сек

Минимальный элемент: 7

Максимальный элемент: 2147483611

**Таблица зависимости времени выполнения от исходных данных и количества потоков:**

| **Число потоков** | **Время исполнения (мс)** | **Ускорение** | **Эффективность** |
| --- | --- | --- | --- |
| **1** | **79,74** | **1** | **1** |
| **2** | **77,31** | **1,03** | **0,515** |
| **3** | **74,22** | **1,07** | **0,366** |
| **4** | **81,38** | **0,97** | **0,242** |
| **5** | **90,45** | **0,88** | **0,176** |
| **6** | **97,26** | **0,82** | **0,136** |

**Ускорение** показывает во сколько раз применение параллельного алгоритма уменьшает время решения задачи по сравнению с последовательным алгоритмом. Ускорение определяется величиной SN=T1/TN, где T1 - время выполнения на одном потоке, TN  - время выполнения на N потоках.

**Эффективность** - величина EN = SN/N, где SN - ускорение, N - количество используемых потоков.

**Strace:**

execve("./main", ["./main", "6"], 0x7ffc16adf5b0 /\* 35 vars \*/) = 0

brk(NULL) = 0x55fa69f30000

arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7ffeda3f2810) = -1 EINVAL (Invalid argument)

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1,

0) = 0x7fb6bb6e0000

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=19739, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 19739, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7fb6bb6db000

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"..., 48, 848) = 48

pread64(3,

"\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\302\211\332Pq\2439\235\350\223\322\257\201\326\243\f"..., 68,

896) = 68

newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=2220400, ...}, 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, 2264656, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) =

0x7fb6bb4b2000

mprotect(0x7fb6bb4da000, 2023424, PROT\_NONE) = 0

mmap(0x7fb6bb4da000, 1658880, PROT\_READ|PROT\_EXEC,

MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7fb6bb4da000

mmap(0x7fb6bb66f000, 360448, PROT\_READ,

MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1bd000) = 0x7fb6bb66f000

mmap(0x7fb6bb6c8000, 24576, PROT\_READ|PROT\_WRITE,

MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x215000) = 0x7fb6bb6c8000

mmap(0x7fb6bb6ce000, 52816, PROT\_READ|PROT\_WRITE,

MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fb6bb6ce000

close(3) = 0

mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1,

0) = 0x7fb6bb4af000

arch\_prctl(ARCH\_SET\_FS, 0x7fb6bb4af740) = 0

set\_tid\_address(0x7fb6bb4afa10) = 8451

set\_robust\_list(0x7fb6bb4afa20, 24) = 0

rseq(0x7fb6bb4b00e0, 0x20, 0, 0x53053053) = 0

mprotect(0x7fb6bb6c8000, 16384, PROT\_READ) = 0

mprotect(0x55fa21183000, 4096, PROT\_READ) = 0

mprotect(0x7fb6bb71a000, 8192, PROT\_READ) = 0

prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) =

0

munmap(0x7fb6bb6db000, 19739) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=1, tv\_nsec=343580500}) = 0

rt\_sigaction(SIGRT\_1, {sa\_handler=0x7fb6bb543870, sa\_mask=[],

sa\_flags=SA\_RESTORER|SA\_ONSTACK|SA\_RESTART|SA\_SIGINFO,

sa\_restorer=0x7fb6bb4f4520}, 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) = 0x7fb6bacae000

mprotect(0x7fb6bacaf000, 8388608, PROT\_READ|PROT\_WRITE) = 0

getrandom("\xc6\x3e\x15\x6a\xde\xea\x91\x14", 8, GRND\_NONBLOCK) = 8

brk(NULL) = 0x55fa69f30000

brk(0x55fa69f51000) = 0x55fa69f51000

rt\_sigprocmask(SIG\_BLOCK, ~[], [], 8) = 0

**clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THR**

**EAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_**

**CLEARTID, child\_tid=0x7fb6bb4ae910, parent\_tid=0x7fb6bb4ae910, exit\_signal=0,**

**stack=0x7fb6bacae000, stack\_size=0x7fff00, tls=0x7fb6bb4ae640} => {parent\_tid=[8469]},**

**88) = 8469**

rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0

mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK,

-1, 0) = 0x7fb6ba4ad000

mprotect(0x7fb6ba4ae000, 8388608, PROT\_READ|PROT\_WRITEstrace: Process 8469 attached

) = 0

[pid 8469] rseq(0x7fb6bb4aefe0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 8451] rt\_sigprocmask(SIG\_BLOCK, ~[], [], 8) = 0

[pid 8451]

**clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THR**

**EAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD**

**CLEARTID, child\_tid=0x7fb6bacad910, parent\_tid=0x7fb6bacad910, exit\_signal=0,**

**stack=0x7fb6ba4ad000, stack\_size=0x7fff00, tls=0x7fb6bacad640} <unfinished ...>**

[pid 8469] <... rseq resumed>) = 0

strace: Process 8470 attached

[pid 8469] set\_robust\_list(0x7fb6bb4ae920, 24 <unfinished ...>

[pid 8451] **<... clone3 resumed> => {parent\_tid=[8470]}, 88) = 8470**

[pid 8451] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 8470] rseq(0x7fb6bacadfe0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 8469] <... set\_robust\_list resumed>) = 0

[pid 8470] <... rseq resumed>) = 0

[pid 8451] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 8469] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 8451] mmap(NULL, 8392704, PROT\_NONE,

MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0 <unfinished ...>

[pid 8470] set\_robust\_list(0x7fb6bacad920, 24 <unfinished ...>

[pid 8451] <... mmap resumed>) = 0x7fb6b9cac000

[pid 8469] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 8451] mprotect(0x7fb6b9cad000, 8388608, PROT\_READ|PROT\_WRITE <unfinished ...>

[pid 8470] <... set\_robust\_list resumed>) = 0

[pid 8451] <... mprotect resumed>) = 0

[pid 8470] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 8451] rt\_sigprocmask(SIG\_BLOCK, ~[], <unfinished ...>

[pid 8470] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 8451] <... rt\_sigprocmask resumed>[], 8) = 0

[pid 8451]

**clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THR**

**EAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_**

**CLEARTID, child\_tid=0x7fb6ba4ac910, parent\_tid=0x7fb6ba4ac910, exit\_signal=0,**

**stack=0x7fb6b9cac000, stack\_size=0x7fff00, tls=0x7fb6ba4ac640}strace: Process 8471**

**attached => {parent\_tid=[8471]}, 88) = 8471**

[pid 8471] rseq(0x7fb6ba4acfe0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 8451] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 8471] <... rseq resumed>) = 0

[pid 8451] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 8451] mmap(NULL, 8392704, PROT\_NONE,

MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0 <unfinished ...>

[pid 8471] set\_robust\_list(0x7fb6ba4ac920, 24) = 0

[pid 8451] <... mmap resumed>) = 0x7fb6b94ab000

[pid 8471] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 8451] mprotect(0x7fb6b94ac000, 8388608, PROT\_READ|PROT\_WRITE) = 0

[pid 8471] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 8451] rt\_sigprocmask(SIG\_BLOCK, ~[], [], 8) = 0

[pid 8451]

**clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THR**

**EAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_**

**CLEARTID, child\_tid=0x7fb6b9cab910, parent\_tid=0x7fb6b9cab910, exit\_signal=0,**

**stack=0x7fb6b94ab000, stack\_size=0x7fff00, tls=0x7fb6b9cab640}strace: Process 8472**

**attached => {parent\_tid=[8472]}, 88) = 8472**

[pid 8472] rseq(0x7fb6b9cabfe0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 8451] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 8472] <... rseq resumed>) = 0

[pid 8451] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 8472] set\_robust\_list(0x7fb6b9cab920, 24 <unfinished ...>

[pid 8451] mmap(NULL, 8392704, PROT\_NONE,

MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0 <unfinished ...>

[pid 8472] <... set\_robust\_list resumed>) = 0

[pid 8451] <... mmap resumed>) = 0x7fb6b8caa000

[pid 8472] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 8451] mprotect(0x7fb6b8cab000, 8388608, PROT\_READ|PROT\_WRITE <unfinished ...>

[pid 8472] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 8451] <... mprotect resumed>) = 0

[pid 8451] rt\_sigprocmask(SIG\_BLOCK, ~[], [], 8) = 0

[pid 8451]

**clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THR**

**EAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_**

**CLEAR**

**TID, child\_tid=0x7fb6b94aa910, parent\_tid=0x7fb6b94aa910, exit\_signal=0,**

**stack=0x7fb6b8caa000, stack\_size=0x7fff00, tls=0x7fb6b94aa640}strace: Process 8473**

**attached=> {parent\_tid=[8473]}, 88) = 8473**

[pid 8473] rseq(0x7fb6b94aafe0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 8451] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 8473] <... rseq resumed>) = 0

[pid 8451] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 8451] mmap(NULL, 8392704, PROT\_NONE,

MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0 <unfinished ...>

[pid 8473] set\_robust\_list(0x7fb6b94aa920, 24 <unfinished ...>

[pid 8451] <... mmap resumed>) = 0x7fb6b84a9000

[pid 8473] <... set\_robust\_list resumed>) = 0

[pid 8451] mprotect(0x7fb6b84aa000, 8388608, PROT\_READ|PROT\_WRITE <unfinished ...>

[pid 8473] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 8451] <... mprotect resumed>) = 0

[pid 8473] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 8451] rt\_sigprocmask(SIG\_BLOCK, ~[], [], 8) = 0

[pid 8451]

**clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THR**

**EAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_**

**CLEARTID, child\_tid=0x7fb6b8ca9910, parent\_tid=0x7fb6b8ca9910, exit\_signal=0,**

**stack=0x7fb6b84a9000, stack\_size=0x7fff00, tls=0x7fb6b8ca9640}strace: Process 8474**

**attached => {parent\_tid=[8474]}, 88) = 8474**

[pid 8474] rseq(0x7fb6b8ca9fe0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 8451] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 8474] <... rseq resumed>) = 0

[pid 8451] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 8474] set\_robust\_list(0x7fb6b8ca9920, 24 <unfinished ...>

[pid 8451] futex(0x7fb6bb4ae910, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME,

8469, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

[pid 8474] <... set\_robust\_list resumed>) = 0

[pid 8474] rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0

[pid 8472] openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

[pid 8472] newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=19739, ...}, AT\_EMPTY\_PATH) =

0

[pid 8472] mmap(NULL, 19739, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7fb6bb6db000

[pid 8469] futex(0x7fb6bb71ca48, FUTEX\_WAIT\_PRIVATE, 2, NULL <unfinished ...>

[pid 8472] close(3) = 0

[pid 8472] mmap(NULL, 134217728, PROT\_NONE,

MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_NORESERVE, -1, 0) = 0x7fb6b04a9000

[pid 8472] munmap(0x7fb6b04a9000, 62222336) = 0

[pid 8472] munmap(0x7fb6b8000000, 4886528) = 0

[pid 8472] mprotect(0x7fb6b4000000, 135168, PROT\_READ|PROT\_WRITE) = 0

[pid 8472] openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libgcc\_s.so.1",

O\_RDONLY|O\_CLOEXEC) = 3

[pid 8472] 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

[pid 8472] newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=125488, ...}, AT\_EMPTY\_PATH)

= 0

[pid 8472] mmap(NULL, 127720, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0

<unfinished ...>

[pid 8470] futex(0x7fb6bb71ca48, FUTEX\_WAIT\_PRIVATE, 2, NULL <unfinished ...>

[pid 8472] <... mmap resumed>) = 0x7fb6b8489000

[pid 8472] mmap(0x7fb6b848c000, 94208, PROT\_READ|PROT\_EXEC,

MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x3000) = 0x7fb6b848c000

[pid 8472] mmap(0x7fb6b84a3000, 16384, PROT\_READ,

MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1a000) = 0x7fb6b84a3000

[pid 8472] mmap(0x7fb6b84a7000, 8192, PROT\_READ|PROT\_WRITE,

MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1d000) = 0x7fb6b84a7000

[pid 8472] close(3) = 0

[pid 8472] mprotect(0x7fb6b84a7000, 4096, PROT\_READ) = 0

[pid 8472] munmap(0x7fb6bb6db000, 19739) = 0

[pid 8472] futex(0x7fb6bb71ca48, FUTEX\_WAKE\_PRIVATE, 1) = 1

[pid 8469] <... futex resumed>) = 0

[pid 8472] futex(0x7fb6b84a8210, FUTEX\_WAKE\_PRIVATE, 2147483647 <unfinished ...>

[pid 8469] futex(0x7fb6bb71ca48, FUTEX\_WAKE\_PRIVATE, 1 <unfinished ...>

[pid 8472] <... futex resumed>) = 0

[pid 8470] <... futex resumed>) = 0

[pid 8469] <... futex resumed>) = 1

[pid 8472] rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], <unfinished ...>

[pid 8470] futex(0x7fb6bb71ca48, FUTEX\_WAKE\_PRIVATE, 1 <unfinished ...>

[pid 8472] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 8469] rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], <unfinished ...>

[pid 8472] madvise(0x7fb6b94ab000, 8368128, MADV\_DONTNEED <unfinished ...>

[pid 8470] <... futex resumed>) = 0

[pid 8472] <... madvise resumed>) = 0

[pid 8469] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 8472] exit(0 <unfinished ...>

[pid 8470] rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], <unfinished ...>

[pid 8472] <... exit resumed>) = ?

[pid 8469] madvise(0x7fb6bacae000, 8368128, MADV\_DONTNEED <unfinished ...>

**[pid 8472] +++ exited with 0 +++**

[pid 8470] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 8469] <... madvise resumed>) = 0

[pid 8469] exit(0 <unfinished ...>

[pid 8470] madvise(0x7fb6ba4ad000, 8368128, MADV\_DONTNEED <unfinished ...>

[pid 8469] <... exit resumed>) = ?

[pid 8473] rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], <unfinished ...>

[pid 8470] <... madvise resumed>) = 0

[pid 8451] <... futex resumed>) = 0

[pid 8473] <... rt\_sigprocmask resumed>NULL, 8) = 0

**[pid 8469] +++ exited with 0 +++**

[pid 8451] futex(0x7fb6bacad910, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME,

8470, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

[pid 8473] madvise(0x7fb6b8caa000, 8368128, MADV\_DONTNEED <unfinished ...>

[pid 8470] exit(0 <unfinished ...>

[pid 8473] <... madvise resumed>) = 0

[pid 8470] <... exit resumed>) = ?

[pid 8451] <... futex resumed>) = 0

[pid 8473] exit(0 <unfinished ...>

**[pid 8470] +++ exited with 0 +++**

[pid 8451] futex(0x7fb6ba4ac910, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME,

8471, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

[pid 8473] <... exit resumed>) = ?

**[pid 8473] +++ exited with 0 +++**

[pid 8474] rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], NULL, 8) = 0

[pid 8474] madvise(0x7fb6b84a9000, 8368128, MADV\_DONTNEED) = 0

[pid 8474] exit(0) = ?

**[pid 8474] +++ exited with 0 +++**

[pid 8471] rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], NULL, 8) = 0

[pid 8471] madvise(0x7fb6b9cac000, 8368128, MADV\_DONTNEED) = 0

[pid 8471] exit(0) = ?

**[pid 8471] +++ exited with 0 +++**

<... futex resumed>) = 0

munmap(0x7fb6bacae000, 8392704) = 0

munmap(0x7fb6ba4ad000, 8392704) = 0

clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, {tv\_sec=1, tv\_nsec=491704600}) = 0

newfstatat(1, "", {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0x4), ...},

AT\_EMPTY\_PATH) = 0

write(1, "\320\222\321\200\320\265\320\274\321\217

\320\262\321\213\320\277\320\276\320\273\320\275\320\265\320\275\320\270\321\217:"...,

48Время выполнения: 0.14812 сек

) = 48

write(1,

"\320\234\320\270\320\275\320\270\320\274\320\260\320\273\321\214\320\275\321\213\320\271

\321\215\320\273\320\265\320\274\320"..., 41Минимальный элемент: 7

) = 41

write(1,

"\320\234\320\260\320\272\321\201\320\270\320\274\320\260\320\273\321\214\320\275\321\213\

320\271 \321\215\320\273\320\265\320"..., 52Максимальный элемент: 2147483611

) = 52

exit\_group(0) = ?

+++ exited with 0 +++

**Вывод**

В результате выполненной лабораторной работы я научился работать с многопоточностью, а именно распараллеливать вычисления и оценивать эффективность работы с различным количеством потоков. Результаты показывают, что с данным заданием наиболее эффективно справляется многопоточная программа. Но с 4-мя и более потоками, временные издержки, уходящие на многопоточность, начинают нивелировать эффективность и уступать по временной производительности однопоточной программе, так как для работы с потоками требуется все больше и больше ресурсов.   
 Безусловно, было интересно продумывать работу каждого потока и организовывать логику их выполнения. Это полезный опыт, который пригодится в будущем.