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»

Выполнил: П. Д. Косарим
Группа: М8О-207БВ-24
Преподаватель: Е. С. Миронов

Москва, 2025

Условие

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 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. Найти в большом целочисленном массиве минимальный и максимальный элементы

Вариант: 17

Метод решения

1. Программа принимает входные параметры: имя файла с массивом и максимальное количество потоков.
2. Массив разбивается на примерно равные части по количеству потоков.
3. Каждый поток ищет локальный минимум и максимум в своей части массива.
4. Главный поток собирает результаты и находит глобальные минимум и максимум.

Описание программы

maxmin.h - заголовочный файл:
1. Объявляет функцию поиска минимума и максимума

maxmin.cpp - функция поиска минимума и максимума:
1. Получает указатель на массив и диапазон поиска
2. Последовательно перебирает элементы в заданном диапазоне
3. Находит локальные минимальный и максимальный элементы
4. Возвращает результаты через указатели

main.cpp - основной процесс:
1. Парсит аргументы командной строки (размер массива, количество потоков)
2. Создает массив случайных чисел
3. Распределяет работу между потоками
4. Запускает потоки для параллельного поиска
5. Собирает и объединяет результаты от всех потоков
6. Сравнивает производительность с однопоточной версией

Используемые системные вызовы и библиотечные функции:
1. std::thread - создание и управление потоками
2. join() - ожидание завершения потока
3. std::chrono - высокоточное измерение времени выполнения
4. std::rand() - генерация случайных чисел

Результаты

При запуске программы, ключами которой являются размер массива и кол-во потоков, программа находит максимум и минимум массива сначала с помощью многопоточного поиска, потом с помощью однопоточного поиска, затем находится ускорение и эффективность многопоточности. Все результаты программы (максимум, минимум, время многопоточного поиска, время однопоточного поиска, ускорение и эффективность) выводятся в консоль.

Пример работы:
./maxmin 100000000 8

8 потоков:
минимум: 1
максимум: 1000
время выполнения: 27674 мксек

однопоточный поиск:
минимум: 1
максимум: 1000
время выполнения: 106817 мксек

ускорение: 3.86x
эффективность: 48.25%
многопоточная версия быстрее в 3.86 раз

Исследование зависимости ускорения и эффективности алгоритма от входных данных и количества потоков (рис.1 и рис.2):
1. Для маленьких массивов (1к-10к элементов) ускорение будет незначительным или отрицательным. Для больших массивов (1кк+ элементов)ускорение станет заметным. Эффективность будет расти с увеличением размера массива. Потому что при маленьких массивах накладные расходы на создание потоков и синхронизацию превышают выгоду от многопоточности.

2. Ускорение будет расти до определенного предела. После превышения количества физических ядер эффективность начнет падать. Оптимальное количество потоков примерно равно количеству физических ядер процессора. Потому что многопоточность дает ограниченный прирост производительности. Слишком большое количество потоков увеличивает накладные расходы.

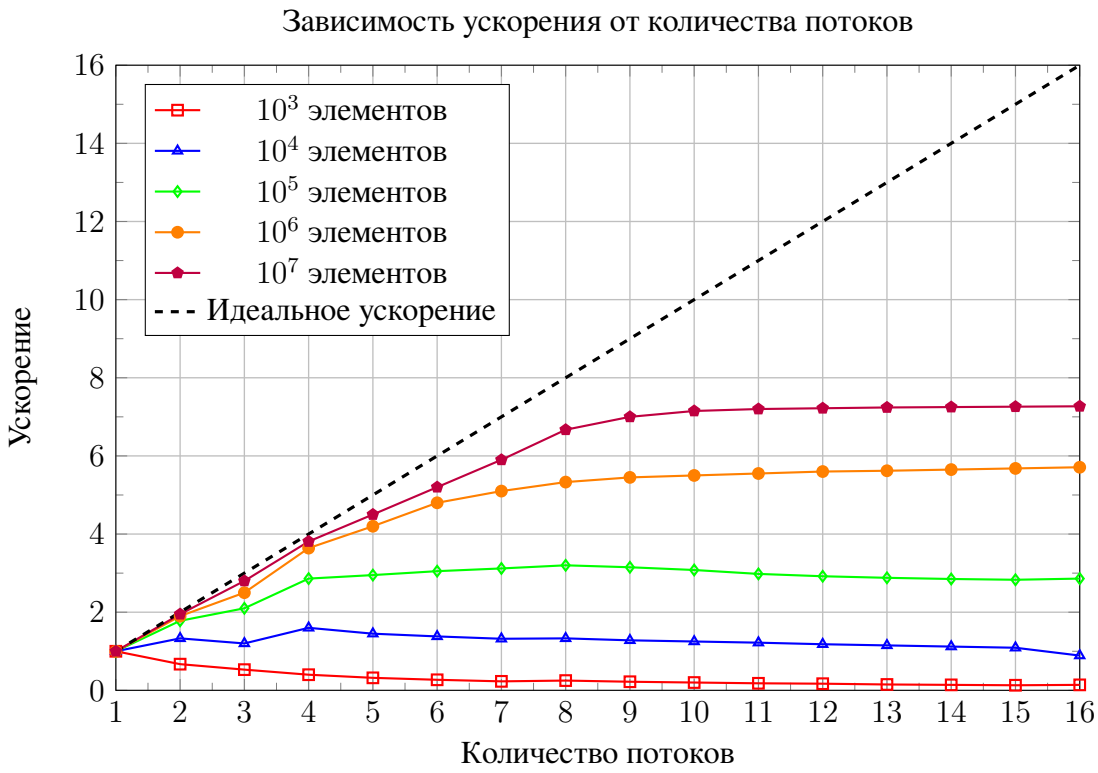
Зависимость ускорения от количества потоков
Ускорение
Количество потоков
10^3 элементов
10^4 элементов
10^5 элементов
10^6 элементов
10^7 элементов
Идеальное ускорение
0
2
4
6
8
10
12
14
16
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

Рис. 1

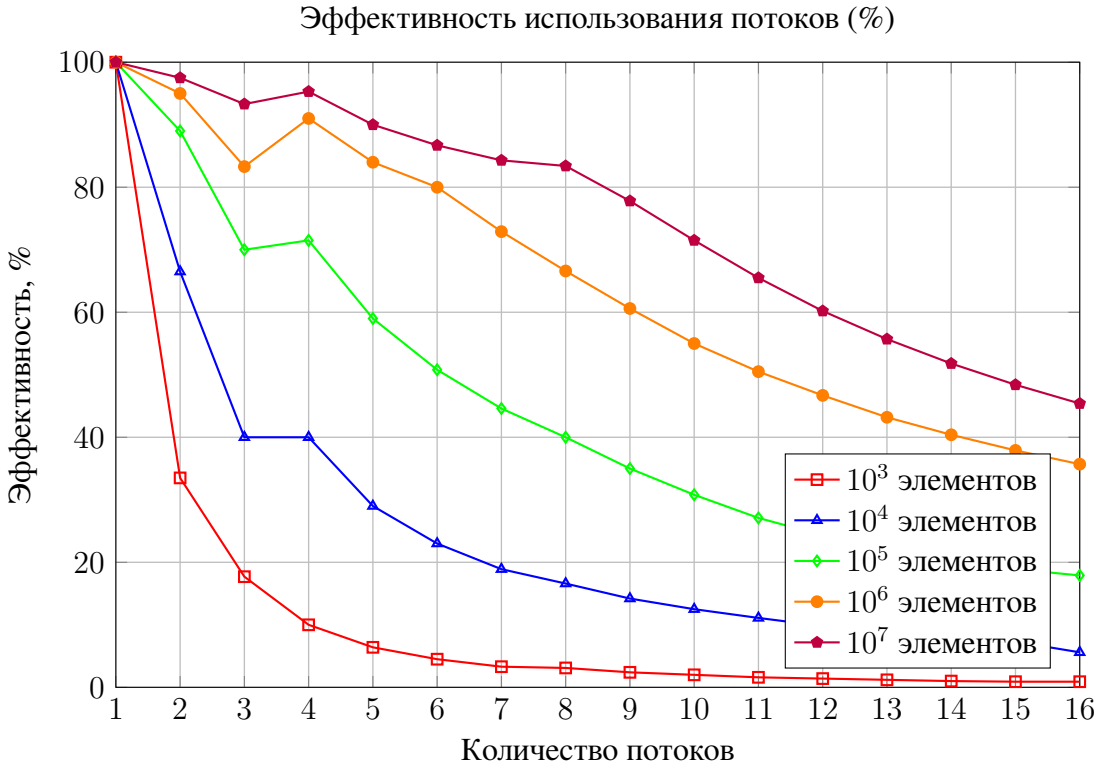
Эффективность использования потоков (%)
Эффективность, %
Количество потоков
10^3 элементов
10^4 элементов
10^5 элементов
10^6 элементов
10^7 элементов
0
20
40
60
80
100
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

Рис. 2

Выводы

Написали программу, которая делит массив на части и запускает несколько потоков, чтобы каждый искал минимум и максимум на своем участке.

Посчитали КПД каждого потока, получилось около 70–95%. Это значит, что потоки работали почти без простоев, не мешая друг другу и действительно ускоряя процесс, а не тратя время на внутренние разборки.

Выяснили, что много потоков не всегда хорошо. Когда создали 16 потоков для того же миллиона чисел, ускорение если и росло, то незначительно, потому что у процессора ограниченное количество настоящих ядер. Когда потоков становится слишком много, они начинают тратить больше времени на переключение между задачами, чем на саму работу.

Чем больше массив, тем выгоднее использовать несколько потоков. На маленьких массивах накладные расходы на создание потоков мешают эффективности.

Программа наглядно показывает все преимущества многопоточности для программы с простым перебором чисел. Каждый поток делает одно и то же, не завися от результатов другого.

Исходная программа

```h
#pragma once

void maxmin(int* arr, int start, int end, int* local_max_ptr, int* local_min_ptr);
```

Листинг 1: maxmin.h

```cpp
#include <iostream>

#include "maxmin.h"

void maxmin (int* arr, int start, int end, int* local_max_ptr, int* local_min_ptr) {
    int local_max = arr[start];
    int local_min = arr[start];

    for (int i = start + 1; i < end; ++i) {
        if (arr[i] > local_max) {
            local_max = arr[i];
        }
        if (arr[i] < local_min) {
            local_min = arr[i];
        }
    }

    *local_max_ptr = local_max;
    *local_min_ptr = local_min;
}
```

Листинг 2: maxmin.cpp

```cpp
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <thread>
#include <climits>
#include <iomanip>
#include <chrono>

#include "maxmin.h"

int* create_arr (int size) {
    if (size <= 0) {
        std::cerr << "size must be positive" << std::endl;
        return nullptr;
    }

    int* arr = new int[size];

    if (arr == nullptr) {
        std::cerr << "memory allocation error" << std::endl;
        exit(1);
    }

    std::srand(std::time(nullptr));
```

```
25
26      for (int i = 0; i< size; ++i) {
27          arr[i] = (std::rand() % 1000) + 1; // 1  1000
28      }
29
30      //
31      //arr[0] = 0;
32      //arr[size - 1] = 2000;
33
34      return arr;
35  }
36
37  int main(int argc, char* argv[]) { // 0 -  , 1 -  arr, 2 -
38      if (argc < 3) {
39          std::cout << "argument error" << std::endl;
40          return 1;
41      }
42
43      int arr_size = std::atoi(argv[1]);
44      int k_threads = std::atoi(argv[2]);
45
46      if (arr_size <= 0 || k_threads <= 0) {
47          std::cout << "argument error" << std::endl;
48          return 1;
49      }
50
51      if (arr_size < k_threads) {
52          k_threads = arr_size;
53      }
54
55      int* arr = create_arr(arr_size);
56      if (arr == nullptr) {
57      return 1;
58      }
59
60      std::thread* arr_thread = new std::thread[k_threads]; //
61
62      int* local_max = new int[k_threads];
63      int* local_min = new int[k_threads];
64      int gl_max = INT_MIN;
65      int gl_min = INT_MAX;
66
67      //
68      auto start_time = std::chrono::high_resolution_clock::now();
69
70      int elem_thread = arr_size / k_threads;
71      int remainder_elem = arr_size % k_threads; //
72      int start_i = 0;
73
74      for (int i = 0; i < k_threads; ++i) {
75          int end_i = start_i + elem_thread;
76
77          if (i < remainder_elem) { //
78              end_i++;
79          }
80
81          arr_thread[i] = std::thread(maxmin, arr, start_i, end_i, &local_max[i], &
                local_min[i]);
```

```
82          start_i = end_i;
83      }
84
85      for (int i = 0; i < k_threads; ++i) {
86          arr_thread[i].join();//
87      }
88
89      //
90      auto end_time = std::chrono::high_resolution_clock::now();
91
92      for (int i = 0; i< k_threads; ++i) {
93          if (gl_max < local_max[i]) {
94              gl_max = local_max[i];
95          }
96          if (gl_min > local_min[i]) {
97              gl_min = local_min[i];
98          }
99      }
100
101     //
102     auto multi_duration = std::chrono::duration_cast<std::chrono::microseconds>(
            end_time - start_time);
103
104     //
105     int one_max, one_min;
106     auto start_single = std::chrono::high_resolution_clock::now();
107
108     maxmin(arr, 0, arr_size, &one_max, &one_min);
109
110     auto end_single = std::chrono::high_resolution_clock::now();
111     auto single_duration = std::chrono::duration_cast<std::chrono::microseconds>(
            end_single - start_single);
112
113     double speedup = (double)single_duration.count() / multi_duration.count();
114     double efficiency = (speedup / k_threads) * 100;
115
116
117     std::cout << k_threads << " :" << std::endl;
118     std::cout << " : " << gl_min << std::endl;
119     std::cout << " : " << gl_max << std::endl;
120     std::cout << "  : " << multi_duration.count() << " " << std::endl;
121
122     std::cout << "\n :" << std::endl;
123     std::cout << " : " << one_min << std::endl;
124     std::cout << " : " << one_max << std::endl;
125     std::cout << "  : " << single_duration.count() << " " << std::endl;
126
127     std::cout << "\n: " << std::fixed << std::setprecision(2) << speedup << "x" << std
            ::endl;
128     std::cout << ": " << std::setprecision(2) << efficiency << "%" << std::endl;
129
130     if (speedup > 1.0) {
131         std::cout << "    " << speedup << " " << std::endl;
132     } else if (speedup < 1.0) {
133         std::cout << "  " << std::endl;
134     } else {
135         std::cout << "" << std::endl;
136     }
```

```
137
138     delete[] arr;
139     delete[] arr_thread;
140     delete[] local_min;
141     delete[] local_max;
142
143     return 0;
144
145 }
```

Листинг 3: main.cpp

```
      9000  execve("./maxmin",["./maxmin","100000000","8"],0x7ffeec805838
/* 49 vars */) = 0
      9000  brk(NULL)                         = 0x60b82b2aa000
      9000
mmap(NULL,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0) =
0x773898681000
      9000  access("/etc/ld.so.preload",R_OK) = -1 ENOENT (Нет такого
файла или каталога)
      9000  openat(AT_FDCWD,"/etc/ld.so.cache",O_RDONLY|O_CLOEXEC) = 3
      9000  fstat(3,{st_mode=S_IFREG|0644,st_size=56203,...}) = 0
      9000  mmap(NULL,56203,PROT_READ,MAP_PRIVATE,3,0) = 0x773898673000
      9000  close(3)                          = 0
      9000
openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libstdc++.so.6",O_RDONLY|O_CLOEXEC)
= 3
      9000
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
      9000  fstat(3,{st_mode=S_IFREG|0644,st_size=2592224,...}) = 0
      9000  mmap(NULL,2609472,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0) =
0x773898200000
      9000
mmap(0x77389829d000,1343488,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3
= 0x77389829d000
      9000
mmap(0x7738983e5000,552960,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,0x1e5000)
= 0x7738983e5000
      9000
mmap(0x77389846c000,57344,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,
= 0x77389846c000
      9000
mmap(0x77389847a000,12608,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,-1
= 0x77389847a000
      9000  close(3)                          = 0
      9000
openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libgcc_s.so.1",O_RDONLY|O_CLOEXEC)
= 3
```

```
     9000
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
     9000  fstat(3,{st_mode=S_IFREG|0644,st_size=183024,...}) = 0
     9000  mmap(NULL,185256,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0) =
0x773898645000
     9000
mmap(0x773898649000,147456,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,
= 0x773898649000
     9000
mmap(0x77389866d000,16384,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,0x28000)
= 0x77389866d000
     9000
mmap(0x773898671000,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,0
= 0x773898671000
     9000  close(3)                          = 0
     9000
openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libc.so.6",O_RDONLY|O_CLOEXEC) = 3
     9000
read(3,"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\243\2\0\0\0\0\0"...,832)
= 832
     9000
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
     9000  fstat(3,{st_mode=S_IFREG|0755,st_size=2125328,...}) = 0
     9000
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
     9000  mmap(NULL,2170256,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0) =
0x773897e00000
     9000
mmap(0x773897e28000,1605632,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3
= 0x773897e28000
     9000
mmap(0x773897fb0000,323584,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,0x1b0000)
= 0x773897fb0000
     9000
mmap(0x773897fff000,24576,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,
= 0x773897fff000
     9000
mmap(0x773898005000,52624,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,-1
= 0x773898005000
     9000  close(3)                          = 0
     9000
openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libm.so.6",O_RDONLY|O_CLOEXEC) = 3
     9000
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
```

```
      9000  fstat(3,{st_mode=S_IFREG|0644,st_size=952616,...}) = 0
      9000  mmap(NULL,950296,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0) =
0x77389855c000
      9000
mmap(0x77389856c000,520192,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,
= 0x77389856c000
      9000
mmap(0x7738985eb000,360448,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,0x8f000)
= 0x7738985eb000
      9000
mmap(0x773898643000,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,0
= 0x773898643000
      9000  close(3)                           = 0
      9000
mmap(NULL,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0) =
0x77389855a000
      9000
mmap(NULL,12288,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0) =
0x773898557000
      9000  arch_prctl(ARCH_SET_FS,0x773898557740) = 0
      9000  set_tid_address(0x773898557a10)   = 9000
      9000  set_robust_list(0x773898557a20,24) = 0
      9000  rseq(0x773898558060,0x20,0,0x53053053) = 0
      9000  mprotect(0x773897fff000,16384,PROT_READ) = 0
      9000  mprotect(0x773898643000,4096,PROT_READ) = 0
      9000  mprotect(0x773898671000,4096,PROT_READ) = 0
      9000  mprotect(0x77389846c000,45056,PROT_READ) = 0
      9000  mprotect(0x60b7f7eec000,4096,PROT_READ) = 0
      9000  mprotect(0x7738986bf000,8192,PROT_READ) = 0
      9000
prlimit64(0,RLIMIT_STACK,NULL,{rlim_cur=8192*1024,rlim_max=RLIM64_INFINITY})
= 0
      9000  munmap(0x773898673000,56203)     = 0
      9000  futex(0x77389847a7bc,FUTEX_WAKE_PRIVATE,2147483647) = 0
      9000  getrandom("\x64\x28\x85\xc4\x21\x3e\xa1\xf1",8,GRND_NONBLOCK)
= 8
      9000  brk(NULL)                           = 0x60b82b2aa000
      9000  brk(0x60b82b2cb000)                 = 0x60b82b2cb000
      9000
mmap(NULL,400003072,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0)
= 0x773880087000
      9000
rt_sigaction(SIGRT_1,{sa_handler=0x773897e99530,sa_mask=[],sa_flags=SA_RESTORER|SA_ON
= 0
      9000  rt_sigprocmask(SIG_UNBLOCK,[RTMIN RT_1],NULL,8) = 0
      9000
mmap(NULL,8392704,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,-1,0) =
0x77387f886000
```

```
      9000  mprotect(0x77387f887000,8388608,PROT_READ|PROT_WRITE) = 0
      9000  rt_sigprocmask(SIG_BLOCK,~[],[],8) = 0
      9000
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|
=>{parent_tid=[9014]},88) = 9014
      9000  rt_sigprocmask(SIG_SETMASK,[],NULL,8) = 0
      9000
mmap(NULL,8392704,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,-1,0) =
0x77387f085000
      9000  mprotect(0x77387f086000,8388608,PROT_READ|PROT_WRITE) = 0
      9014  rseq(0x773880086fe0,0x20,0,0x53053053 <unfinished ...>
      9000  rt_sigprocmask(SIG_BLOCK,~[], <unfinished ...>
      9014  <... rseq resumed>)                = 0
      9000  <... rt_sigprocmask resumed>[],8) = 0
      9014  set_robust_list(0x7738800869a0,24 <unfinished ...>
      9000
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|
<unfinished ...>
      9014  <... set_robust_list resumed>)     = 0
      9014  rt_sigprocmask(SIG_SETMASK,[], <unfinished ...>
      9000  <... clone3 resumed>=>{parent_tid=[9015]},88) = 9015
      9015  rseq(0x77387f885fe0,0x20,0,0x53053053 <unfinished ...>
      9014  <... rt_sigprocmask resumed>NULL,8) = 0
      9000  rt_sigprocmask(SIG_SETMASK,[], <unfinished ...>
      9015  <... rseq resumed>)                = 0
      9000  <... rt_sigprocmask resumed>NULL,8) = 0
      9015  set_robust_list(0x77387f8859a0,24 <unfinished ...>
      9000
mmap(NULL,8392704,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,-1,0
<unfinished ...>
      9015  <... set_robust_list resumed>)     = 0
      9000  <... mmap resumed>)                = 0x77387e884000
      9015  rt_sigprocmask(SIG_SETMASK,[], <unfinished ...>
      9000  mprotect(0x77387e885000,8388608,PROT_READ|PROT_WRITE
<unfinished ...>
      9015  <... rt_sigprocmask resumed>NULL,8) = 0
      9000  <... mprotect resumed>)            = 0
      9000  rt_sigprocmask(SIG_BLOCK,~[],[],8) = 0
      9000
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|
=>{parent_tid=[9016]},88) = 9016
      9016  rseq(0x77387f084fe0,0x20,0,0x53053053 <unfinished ...>
      9000  rt_sigprocmask(SIG_SETMASK,[], <unfinished ...>
      9016  <... rseq resumed>)                = 0
      9000  <... rt_sigprocmask resumed>NULL,8) = 0
      9000
mmap(NULL,8392704,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,-1,0
<unfinished ...>
```

```
     9016  set_robust_list(0x77387f0849a0,24 <unfinished ...>
     9000  <... mmap resumed>)               = 0x77387e083000
     9000  mprotect(0x77387e084000,8388608,PROT_READ|PROT_WRITE) = 0
     9016  <... set_robust_list resumed>)    = 0
     9000  rt_sigprocmask(SIG_BLOCK,~[], <unfinished ...>
     9016  rt_sigprocmask(SIG_SETMASK,[], <unfinished ...>
     9000  <... rt_sigprocmask resumed>[],8) = 0
     9000
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|
<unfinished ...>
     9016  <... rt_sigprocmask resumed>NULL,8) = 0
     9000  <... clone3 resumed>=>{parent_tid=[9017]},88) = 9017
     9000  rt_sigprocmask(SIG_SETMASK,[], <unfinished ...>
     9017  rseq(0x77387e883fe0,0x20,0,0x53053053 <unfinished ...>
     9000  <... rt_sigprocmask resumed>NULL,8) = 0
     9017  <... rseq resumed>)               = 0
     9000
mmap(NULL,8392704,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,-1,0
<unfinished ...>
     9017  set_robust_list(0x77387e8839a0,24 <unfinished ...>
     9000  <... mmap resumed>)               = 0x77387d882000
     9017  <... set_robust_list resumed>)    = 0
     9000  mprotect(0x77387d883000,8388608,PROT_READ|PROT_WRITE
<unfinished ...>
     9017  rt_sigprocmask(SIG_SETMASK,[], <unfinished ...>
     9000  <... mprotect resumed>)           = 0
     9017  <... rt_sigprocmask resumed>NULL,8) = 0
     9000  rt_sigprocmask(SIG_BLOCK,~[],[],8) = 0
     9000
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|
=>{parent_tid=[9018]},88) = 9018
     9018  rseq(0x77387e082fe0,0x20,0,0x53053053 <unfinished ...>
     9000  rt_sigprocmask(SIG_SETMASK,[], <unfinished ...>
     9018  <... rseq resumed>)               = 0
     9000  <... rt_sigprocmask resumed>NULL,8) = 0
     9018  set_robust_list(0x77387e0829a0,24 <unfinished ...>
     9000
mmap(NULL,8392704,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,-1,0
<unfinished ...>
     9018  <... set_robust_list resumed>)    = 0
     9000  <... mmap resumed>)               = 0x77387d081000
     9018  rt_sigprocmask(SIG_SETMASK,[], <unfinished ...>
     9000  mprotect(0x77387d082000,8388608,PROT_READ|PROT_WRITE
<unfinished ...>
     9018  <... rt_sigprocmask resumed>NULL,8) = 0
     9000  <... mprotect resumed>)           = 0
     9000  rt_sigprocmask(SIG_BLOCK,~[],[],8) = 0
     9000
```

```
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|
=>{parent_tid=[9019]},88) = 9019
     9019  rseq(0x77387d881fe0,0x20,0,0x53053053 <unfinished ...>
     9000  rt_sigprocmask(SIG_SETMASK,[], <unfinished ...>
     9019  <... rseq resumed>)               = 0
     9000  <... rt_sigprocmask resumed>NULL,8) = 0
     9019  set_robust_list(0x77387d8819a0,24 <unfinished ...>
     9000
mmap(NULL,8392704,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,-1,0
<unfinished ...>
     9019  <... set_robust_list resumed>)    = 0
     9000  <... mmap resumed>)               = 0x77387c880000
     9019  rt_sigprocmask(SIG_SETMASK,[], <unfinished ...>
     9000  mprotect(0x77387c881000,8388608,PROT_READ|PROT_WRITE
<unfinished ...>
     9019  <... rt_sigprocmask resumed>NULL,8) = 0
     9000  <... mprotect resumed>)           = 0
     9000  rt_sigprocmask(SIG_BLOCK,~[],[],8) = 0
     9000
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|
=>{parent_tid=[9020]},88) = 9020
     9020  rseq(0x77387d080fe0,0x20,0,0x53053053 <unfinished ...>
     9000  rt_sigprocmask(SIG_SETMASK,[], <unfinished ...>
     9020  <... rseq resumed>)               = 0
     9000  <... rt_sigprocmask resumed>NULL,8) = 0
     9020  set_robust_list(0x77387d0809a0,24 <unfinished ...>
     9000
mmap(NULL,8392704,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK,-1,0
<unfinished ...>
     9020  <... set_robust_list resumed>)    = 0
     9000  <... mmap resumed>)               = 0x77387c07f000
     9020  rt_sigprocmask(SIG_SETMASK,[], <unfinished ...>
     9000  mprotect(0x77387c080000,8388608,PROT_READ|PROT_WRITE
<unfinished ...>
     9020  <... rt_sigprocmask resumed>NULL,8) = 0
     9000  <... mprotect resumed>)           = 0
     9000  rt_sigprocmask(SIG_BLOCK,~[],[],8) = 0
     9000
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|
=>{parent_tid=[9021]},88) = 9021
     9021  rseq(0x77387c87ffe0,0x20,0,0x53053053 <unfinished ...>
     9000  rt_sigprocmask(SIG_SETMASK,[], <unfinished ...>
     9021  <... rseq resumed>)               = 0
     9000  <... rt_sigprocmask resumed>NULL,8) = 0
     9021  set_robust_list(0x77387c87f9a0,24 <unfinished ...>
     9000
futex(0x773880086990,FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME,9014,NULL,FUTEX_BITSET_MA
<unfinished ...>
```

```
     9021  <... set_robust_list resumed>)    = 0
     9021  rt_sigprocmask(SIG_SETMASK,[],NULL,8) = 0
     9021  mmap(NULL,134217728,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0
<unfinished ...>
     9020  mmap(NULL,134217728,PROT_NONE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0
<unfinished ...>
     9021  <... mmap resumed>)               = 0x773874000000
     9020  <... mmap resumed>)               = 0x77386c000000
     9021  munmap(0x773878000000,67108864 <unfinished ...>
     9020  munmap(0x773870000000,67108864 <unfinished ...>
     9021  <... munmap resumed>)             = 0
     9020  <... munmap resumed>)             = 0
     9021  mprotect(0x773874000000,135168,PROT_READ|PROT_WRITE
<unfinished ...>
     9020  mprotect(0x77386c000000,135168,PROT_READ|PROT_WRITE
<unfinished ...>
     9021  <... mprotect resumed>)           = 0
     9020  <... mprotect resumed>)           = 0
     9021  rt_sigprocmask(SIG_BLOCK,~[RT_1], <unfinished ...>
     9020  rt_sigprocmask(SIG_BLOCK,~[RT_1], <unfinished ...>
     9021  <... rt_sigprocmask resumed>NULL,8) = 0
     9020  <... rt_sigprocmask resumed>NULL,8) = 0
     9021  madvise(0x77387c07f000,8368128,MADV_DONTNEED <unfinished ...>
     9020  madvise(0x77387c880000,8368128,MADV_DONTNEED <unfinished ...>
     9021  <... madvise resumed>)            = 0
     9020  <... madvise resumed>)            = 0
     9021  exit(0 <unfinished ...>
     9020  exit(0 <unfinished ...>
     9021  <... exit resumed>)               = ?
     9020  <... exit resumed>)               = ?
     9021  +++ exited with 0 +++
     9020  +++ exited with 0 +++
     9015  rt_sigprocmask(SIG_BLOCK,~[RT_1],NULL,8) = 0
     9015  madvise(0x77387f085000,8368128,MADV_DONTNEED) = 0
     9015  exit(0)                           = ?
     9015  +++ exited with 0 +++
     9019  rt_sigprocmask(SIG_BLOCK,~[RT_1],NULL,8) = 0
     9019  madvise(0x77387d081000,8368128,MADV_DONTNEED) = 0
     9019  exit(0)                           = ?
     9019  +++ exited with 0 +++
     9017  rt_sigprocmask(SIG_BLOCK,~[RT_1],NULL,8) = 0
     9017  madvise(0x77387e083000,8368128,MADV_DONTNEED) = 0
     9017  exit(0)                           = ?
     9017  +++ exited with 0 +++
     9018  rt_sigprocmask(SIG_BLOCK,~[RT_1],NULL,8) = 0
     9018  madvise(0x77387d882000,8368128,MADV_DONTNEED) = 0
     9018  exit(0)                           = ?
     9018  +++ exited with 0 +++
```

```
     9014  rt_sigprocmask(SIG_BLOCK,~[RT_1], <unfinished ...>
     9016  rt_sigprocmask(SIG_BLOCK,~[RT_1],NULL,8) = 0
     9014  <... rt_sigprocmask resumed>NULL,8) = 0
     9016  madvise(0x77387e884000,8368128,MADV_DONTNEED) = 0
     9016  exit(0)                           = ?
     9016  +++ exited with 0 +++
     9014  madvise(0x77387f886000,8368128,MADV_DONTNEED) = 0
     9014  exit(0)                           = ?
     9000  <... futex resumed>)              = 0
     9014  +++ exited with 0 +++
     9000  munmap(0x77387f886000,8392704)   = 0
     9000  munmap(0x77387f085000,8392704)   = 0
     9000  munmap(0x77387e884000,8392704)   = 0
     9000  munmap(0x77387e083000,8392704)   = 0
     9000  fstat(1,{st_mode=S_IFCHR|0620,st_rdev=makedev(0x88,0),...}) =
0
     9000  write(1,"8
\320\277\320\276\321\202\320\276\320\272\320\276\320\262:\n",18) = 18
     9000  write(1,"
\320\274\320\270\320\275\320\270\320\274\321\203\320\274: 1\n",20) = 20
     9000  write(1,"
\320\274\320\260\320\272\321\201\320\270\320\274\321\203\320\274:
1000\n",25) = 25
     9000  write(1,"  \320\262\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"...,52)
= 52
     9000  write(1,"\n",1)                 = 1
     9000
write(1,"\320\276\320\264\320\275\320\276\320\277\320\276\321\202\320\276\321\207\320
\320\277\320\276\320\270\321"...,37) = 37
     9000  write(1,"
\320\274\320\270\320\275\320\270\320\274\321\203\320\274: 1\n",20) = 20
     9000  write(1,"
\320\274\320\260\320\272\321\201\320\270\320\274\321\203\320\274:
1000\n",25) = 25
     9000  write(1,"  \320\262\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"...,53)
= 53
     9000  write(1,"\n",1)                 = 1
     9000
write(1,"\321\203\321\201\320\272\320\276\321\200\320\265\320\275\320\270\320\265:
3.86x\n",26) = 26
     9000
write(1,"\321\215\321\204\321\204\320\265\320\272\321\202\320\270\320\262\320\275\320
48.2"...,35) = 35
     9000
write(1,"\320\274\320\275\320\276\320\263\320\276\320\277\320\276\321\202\320\276\321
\320\262\320\265\321"...,70) = 70
```

```
9000  munmap(0x773880087000,400003072) = 0
9000  exit_group(0)                     = ?
9000  +++ exited with 0 +++
```

Листинг 4: *Strace логи*