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: М8О-215Б-23

Студент: Авраменко Д.А.

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 01.11.24

Москва, 2024

# Постановка задачи

**Вариант 1.**

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.

Отсортировать массив целых чисел при помощи битонической сортировки

# Общий метод и алгоритм решения

Использованные системные вызовы:

- pthread_create(pthread_t *thread, const pthread_attr_t *attr, void *(*start_routine) (void *), void *arg); – Создает новый поток, возвращает 0 при успехе
- pthread_join(pthread_t thread, void **retval); - Ожидает завершения указанного потока. Блокирует вызывающий поток до завершения целевого потока
- pthread_mutex_lock(pthread_mutex_t *mutex); - Блокирует мьютекс. Если мьютекс уже заблокирован, поток блокируется до освобождения
- pthread_mutex_unlock(pthread_mutex_t *mutex); - Разблокирует мьютекс. Позволяет другим потокам захватить мьютекс

Ключевые особенности:

1. Алгоритм работает только с массивами, размер которых является **степенью двойки**
2. Использует многопоточность через POSIX threads (pthread)
3. Контролирует количество активных потоков через мьютекс

Работа:

1. Инициализация:

- Принимает два параметра: размер массива и максимальное число потоков
- Создает случайный массив заданного размера
- Инициализирует пул потоков

2. Битоническая сортировка

bitonicSort:

- Рекурсивно разделяет массив на две части
- Сортирует левую половину по возрастанию (dir = 1)
- Сортирует правую половину по убыванию (dir = 0)
- После этого объединяет части используя bitonicMerge

bitonicMerge:

- Сравнивает и меняет местами элементы на определенном расстоянии
- Рекурсивно обрабатывает получившиеся подпоследовательности

- Использует оптимизацию: для маленьких подмассивов (< 1024) работает без создания новых потоков

3. Управление потоками:

- Программа следит за количеством активных потоков через active_threads
- Если число активных потоков меньше max_threads, создаются новые потоки
- Если достигнут лимит, выполнение продолжается в текущем потоке

4. Проверка результатов:

- После сортировки проверяется корректность (каждый следующий элемент должен быть больше предыдущего)
- Измеряется время выполнения сортировки

## Замеры эффективности

Замеры проводились для 6 разных длин массивов. Количество потоков было от 1 до 64. Так как массив заполняется случайными значениями, то для каждого количества потоков проводилось 5 замеров и бралось среднее значение времени.

На моем процессоре доступно 16 физических потоков

График со всеми замерами. Оси с логарифмическими шкалами

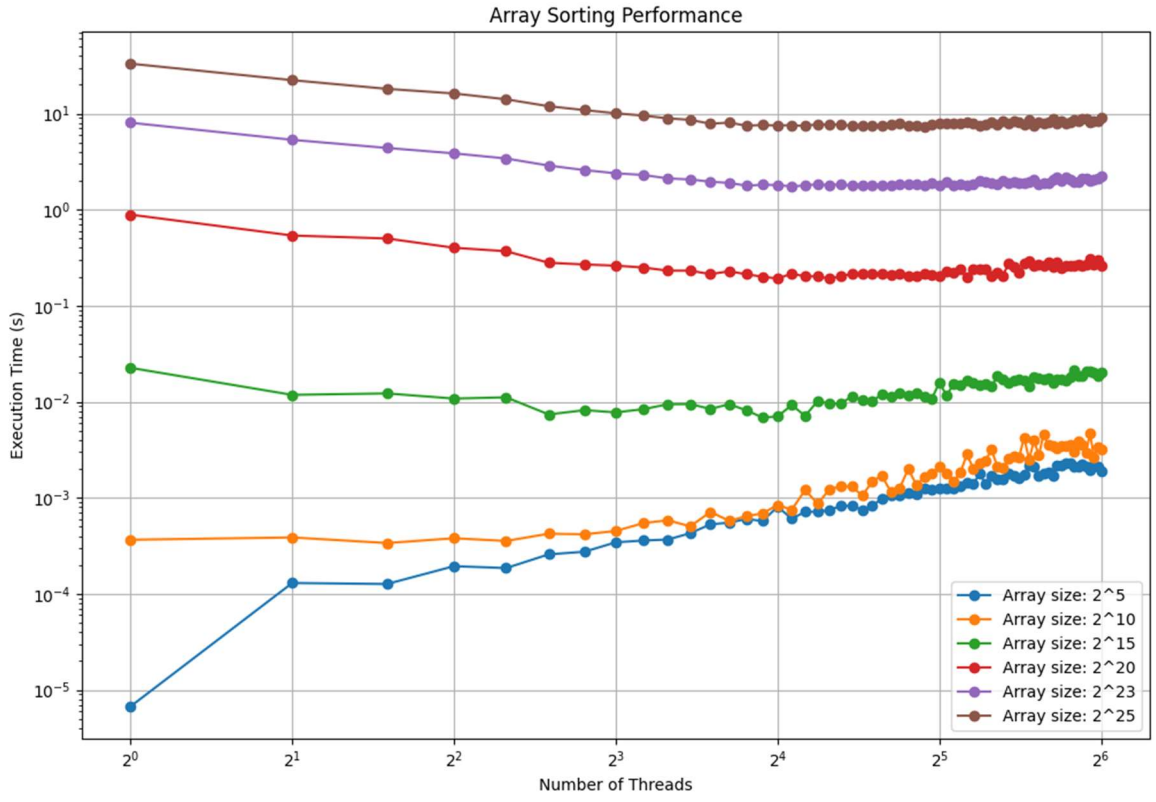


График для массива из 1024 элементов:

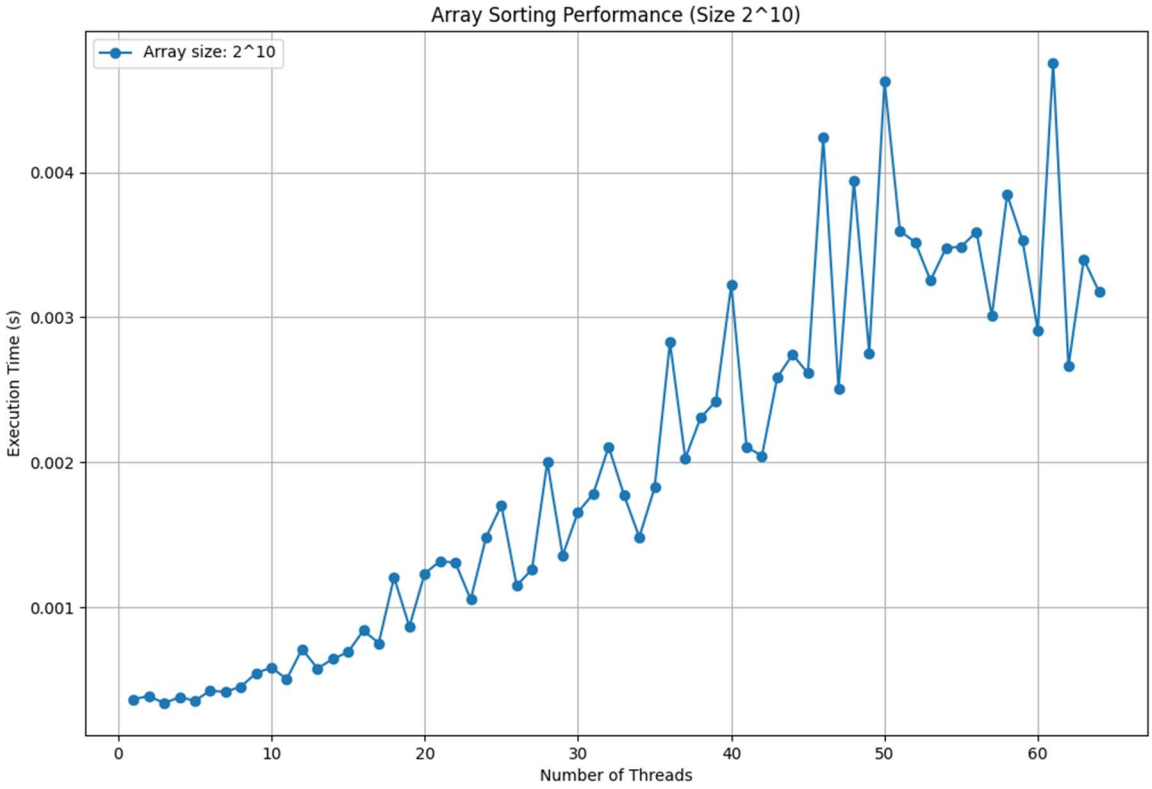



График для массива из 32_768 элементов:

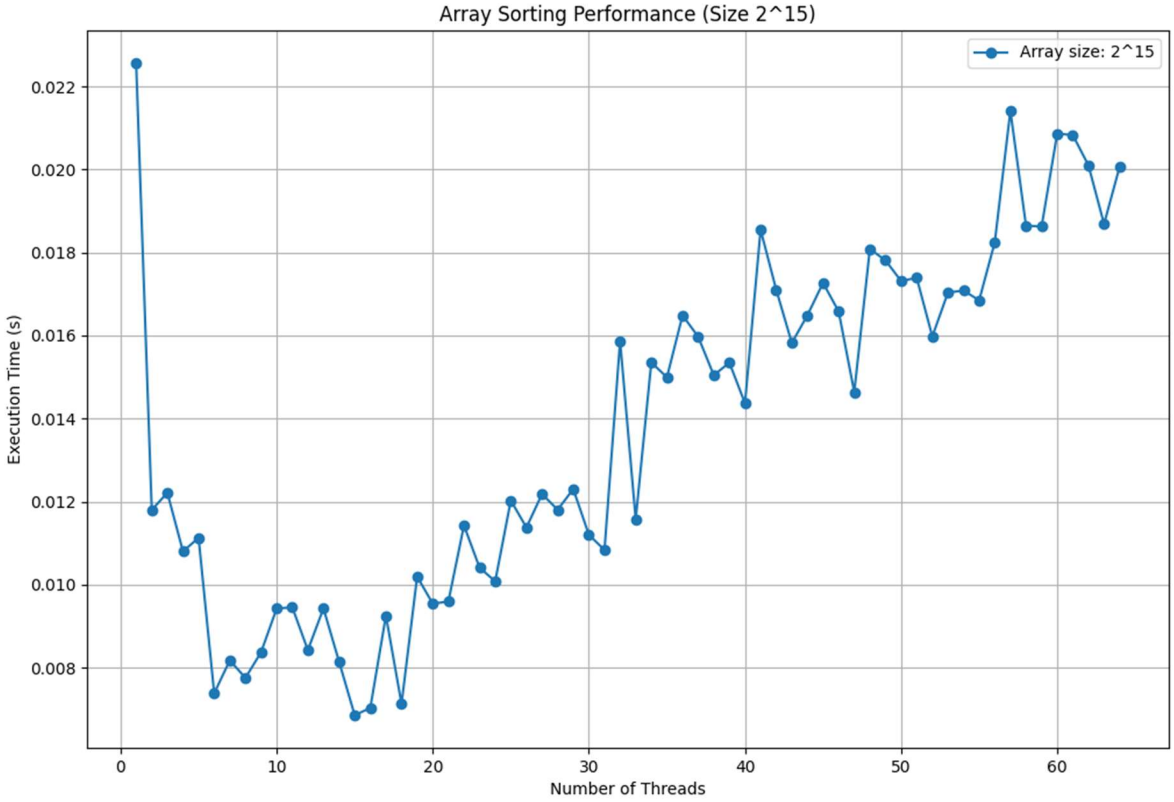


График для массива из 1_048_576 элементов:

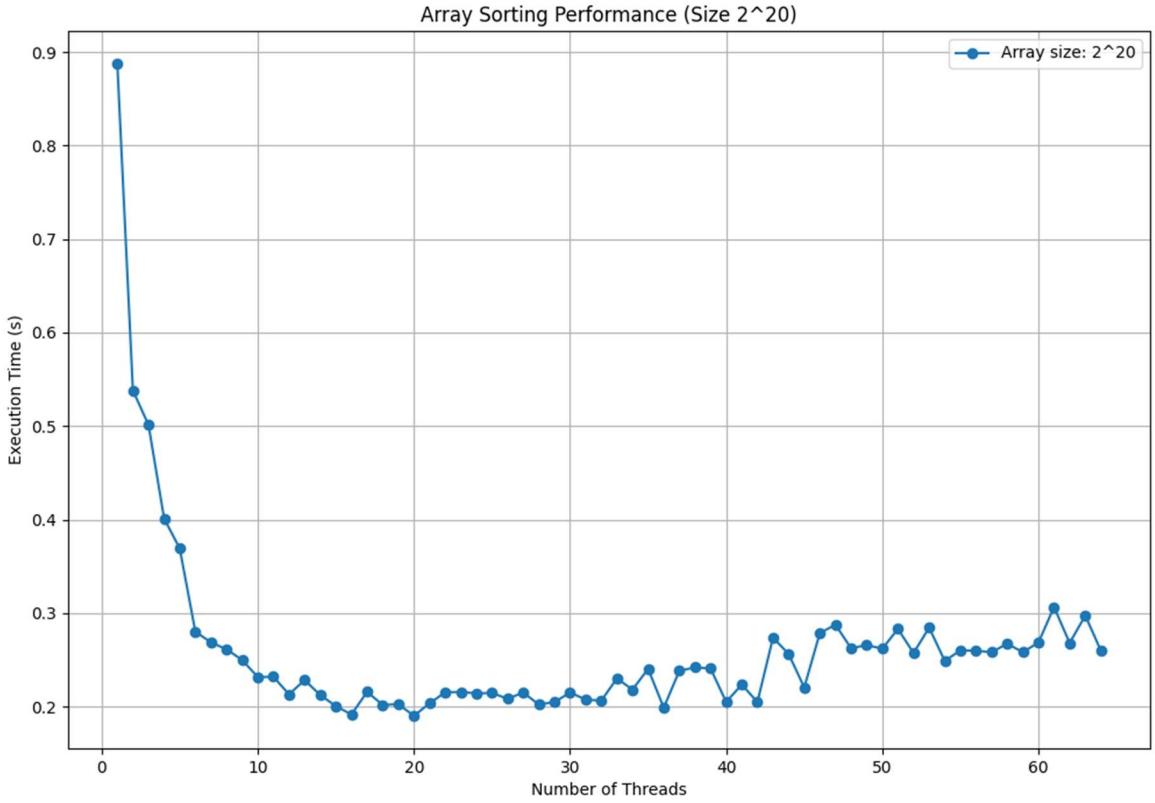



График для массива из 8_388_608 элементов:

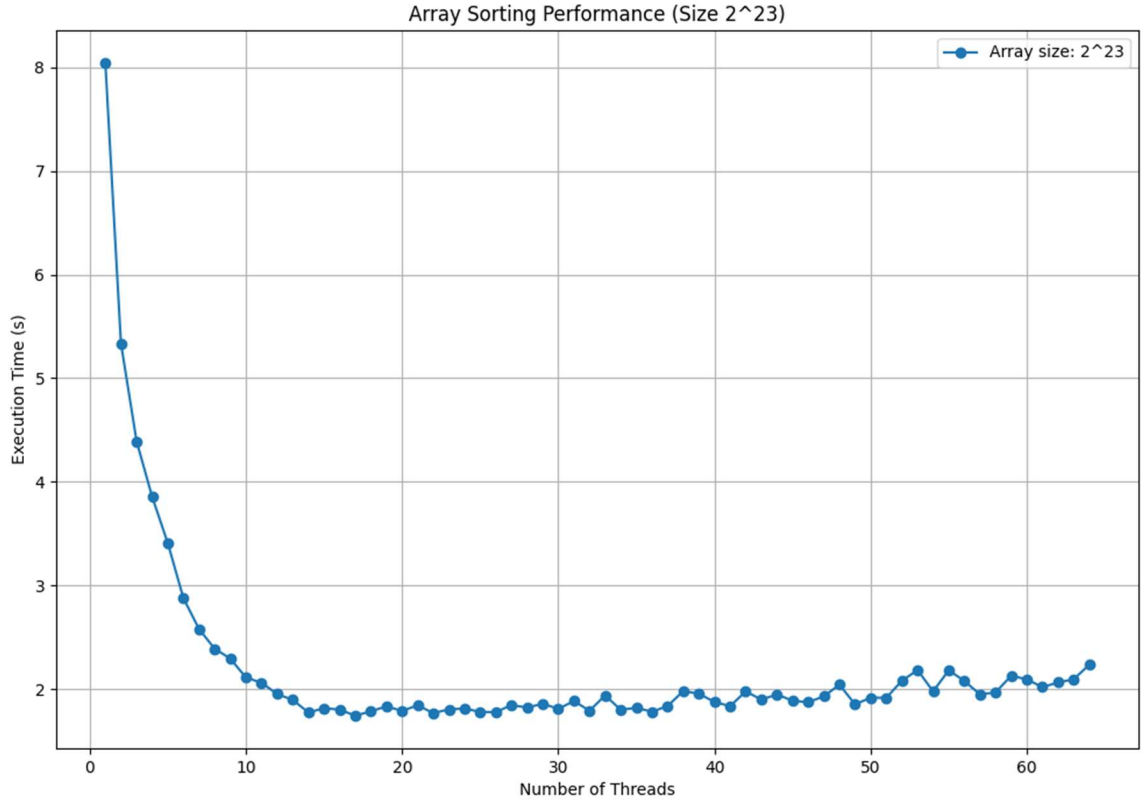
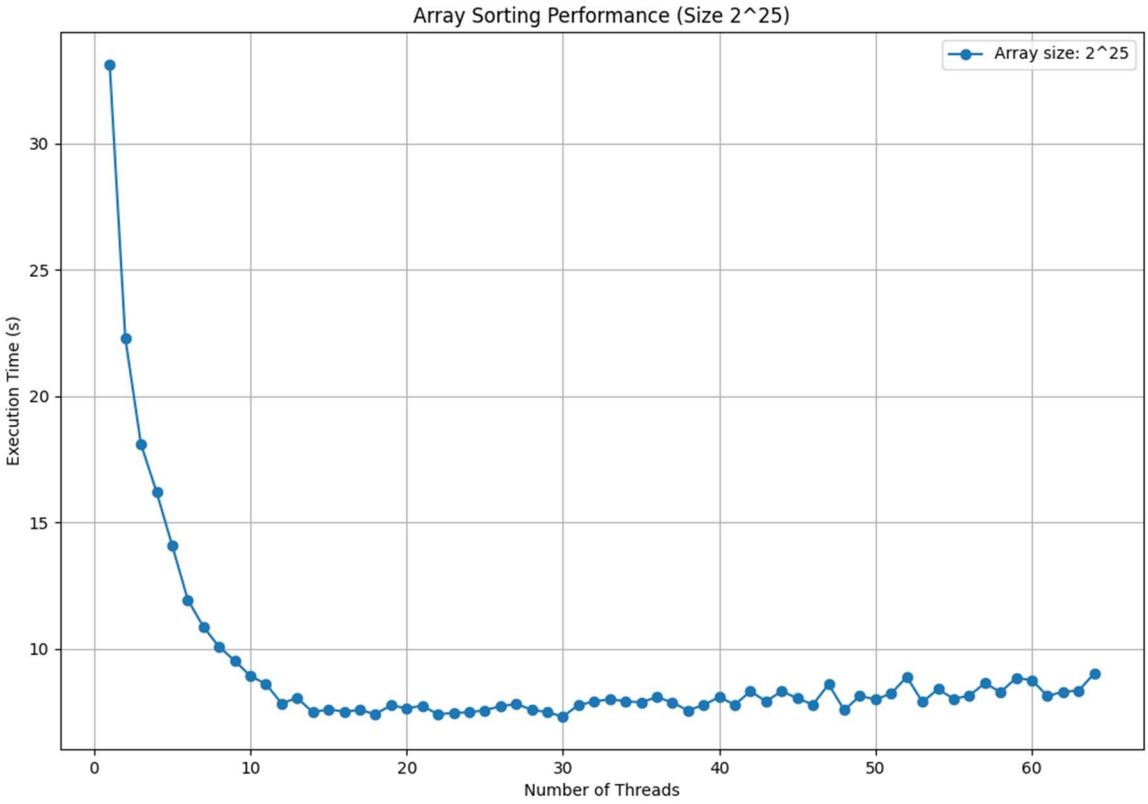
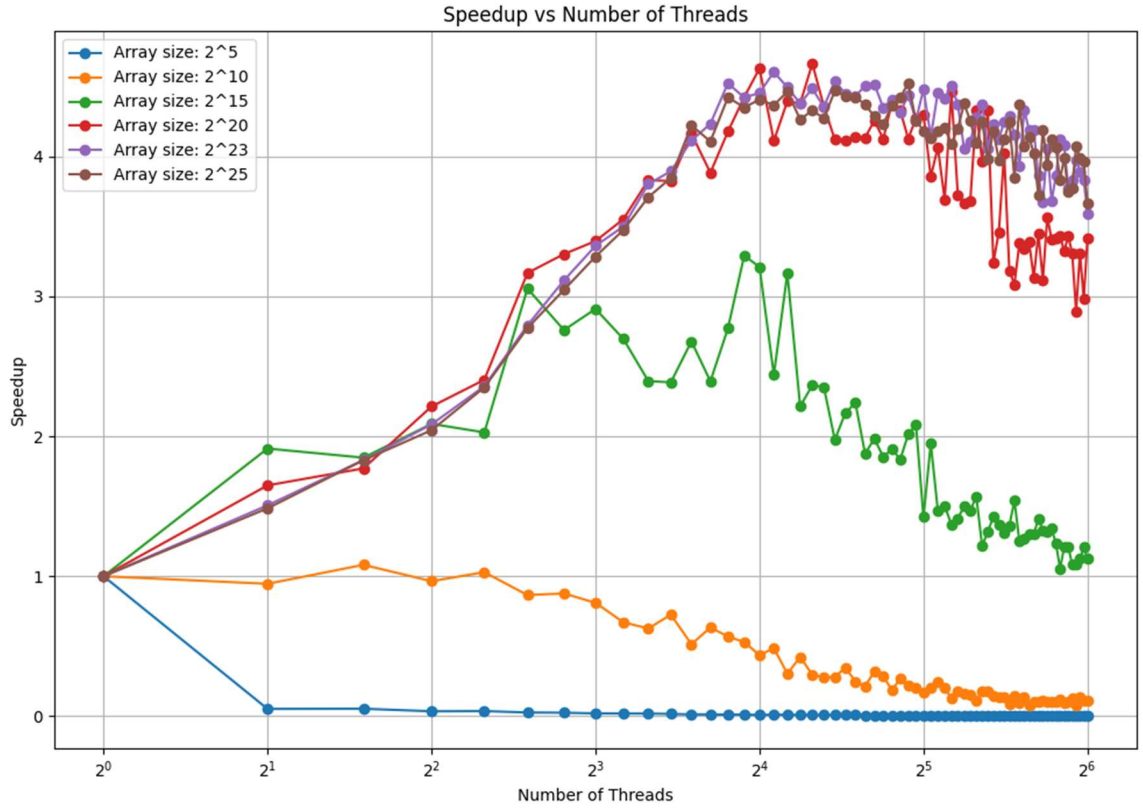


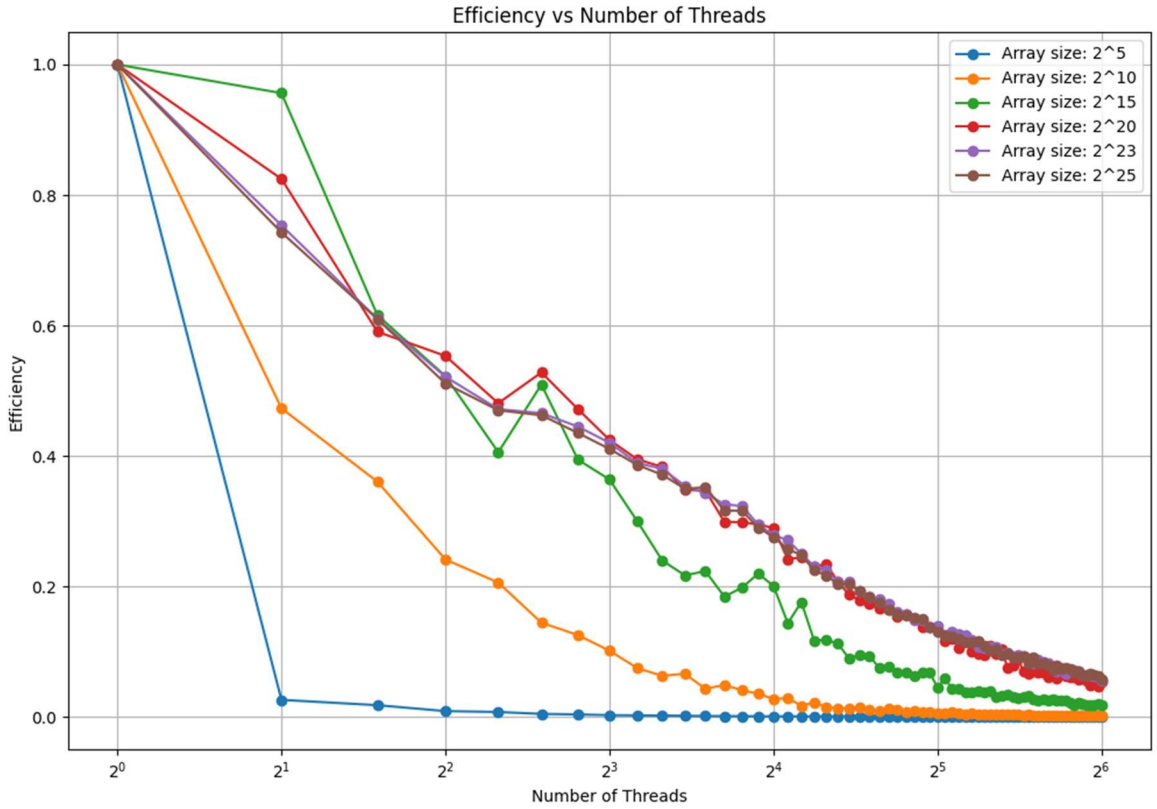
График для массива из 33_554_432 элементов:




Ускорение:



Эффективность:




**Маленькие массивы $2^{10}$ и $2^{15}$:**

Для них многопоточность дает негативный эффект - время выполнения растет с увеличением числа потоков. Это происходит потому, что накладные расходы на создание и управление потоками превышают выигрыш от параллельной обработки. Оверхед на синхронизацию потоков больше, чем время самой сортировки

**Средние массивы $2^{20}$:**

Заметно значительное улучшение производительности при использовании до ~15-16 потоков. После 16 потоков (что соответствует количеству физических ядер) производительность стабилизируется. Начальное время ~0.9с падает до ~0.2с при оптимальном количестве потоков

**Большие массивы $2^{23}$ и $2^{25}$:**

Очень похожая картина на всех размерах. Резкое улучшение производительности до 15-16 потоков. После достижения количества физических ядер график выходит на плато

Для $2^{25}$ время падает с ~33с до ~8с

**Увеличение времени при 45-64 потоках:**

Когда количество потоков становится значительно больше количества физических ядер (особенно после 45-50 потоков), время выполнения начинает постепенно расти. Как я понял, это происходит по нескольким причинам

1. Контекстное переключение (Context Switching):

Когда потоков намного больше, чем ядер, операционная система вынуждена часто переключаться между потоками. Каждое переключение контекста требует сохранения состояния текущего потока, загрузки состояния следующего потока, очистки и перезагрузки кэшей процессора

2. Деградация кэша:

Большое количество потоков приводит к более частой инвалидации кэша. Данные одного потока вытесняют данные другого из кэша и увеличивается количество кэш-промахов (cache misses)

3. Управление ресурсами:

Возрастают накладные расходы на планирование потоков, увеличивается конкуренция за общие ресурсы (память, шина данных), растет сложность синхронизации между потоками

4. Memory Bus Saturation:

При большом количестве потоков возрастает нагрузка на шину памяти и потоки начинают конкурировать за пропускную способность памяти

**Ускорение:**

Для 32 и 1024 ускорение отрицательное. Так как стоит время сортировки меньше, чем менеджеринг процессов. Даже при условии, что при размере <1024 сортировка считается рекурсивно.

Интересный паттерн у массива с длиной $2^{15}$. Сначала ускорение растет, потом падает. То есть сначала увеличение потоков стоит того, а потом накладные расходы перевешивают.

**Эффективность:**

Интересно, что для того же $2^{15}$ эффективность для двух потоков самая хорошая

**Выводы:**

- Параллельная сортировка эффективна только для достаточно больших массивов ($2^{20}$ элементов)

- Оптимальное количество потоков примерно равно (может чуть меньше) количеству физических ядер (16 в моем случае)

- Дальнейшее увеличение числа потоков не дает прироста

# Код программы

main.cpp

```cpp
#include <iostream>
#include <vector>
#include <pthread.h>
#include <cstdlib>
#include <chrono>

struct ThreadData {
    std::vector<int>* arr;
    int low;
    int cnt;
    int dir;
};

pthread_mutex_t thread_count_mutex = PTHREAD_MUTEX_INITIALIZER;
int max_threads;
int active_threads = 0;

// Пул потоков для переиспользования
pthread_t* thread_pool = nullptr;
int thread_pool_size = 0;

void compareAndSwap(std::vector<int>& arr, int i, int j, int dir) {
    if (dir == (arr[i] > arr[j])) {
        std::swap(arr[i], arr[j]);
    }
}

void* bitonicMerge(void* arg) {
    ThreadData* data = (ThreadData*)arg;
    std::vector<int>& arr = *(data->arr);
    int low = data->low;
    int cnt = data->cnt;
    int dir = data->dir;
```

```
    if (cnt > 1) {
        int k = cnt / 2;
        for (int i = low; i < low + k; i++) {
            compareAndSwap(arr, i, i + k, dir);
        }

        // Используем рекурсивный вызов для небольших размеров
        if (k < 1024) {
            ThreadData left_data = {data->arr, low, k, dir};
            ThreadData right_data = {data->arr, low + k, k, dir};
            bitonicMerge(&left_data);
            bitonicMerge(&right_data);
            return nullptr;
        }

        ThreadData left_data = {data->arr, low, k, dir};
        ThreadData right_data = {data->arr, low + k, k, dir};

        pthread_mutex_lock(&thread_count_mutex);
        if (active_threads + 1 <= max_threads) {
            active_threads += 1;
            pthread_mutex_unlock(&thread_count_mutex);

            pthread_t thread;
            pthread_create(&thread, nullptr, bitonicMerge, &left_data);
            bitonicMerge(&right_data);
            pthread_join(thread, nullptr);

            pthread_mutex_lock(&thread_count_mutex);
            active_threads -= 1;
            pthread_mutex_unlock(&thread_count_mutex);
        } else {
            pthread_mutex_unlock(&thread_count_mutex);
            bitonicMerge(&left_data);
            bitonicMerge(&right_data);
        }
    }
    return nullptr;
}

void* bitonicSort(void* arg) {
    ThreadData* data = (ThreadData*)arg;
    std::vector<int>& arr = *(data->arr);
    int low = data->low;
    int cnt = data->cnt;
    int dir = data->dir;

    if (cnt > 1) {
        int k = cnt / 2;

        ThreadData left_data = {data->arr, low, k, 1};
        ThreadData right_data = {data->arr, low + k, k, 0};

        pthread_t threads[2];

        pthread_mutex_lock(&thread_count_mutex);
        if (active_threads + 2 <= max_threads) {
            active_threads += 2;
            pthread_mutex_unlock(&thread_count_mutex);

            pthread_create(&threads[0], nullptr, bitonicSort, &left_data);
            pthread_create(&threads[1], nullptr, bitonicSort, &right_data);

            pthread_join(threads[0], nullptr);
            pthread_join(threads[1], nullptr);
```

```
            pthread_mutex_lock(&thread_count_mutex);
            active_threads -= 2;
            pthread_mutex_unlock(&thread_count_mutex);
        } else {
            pthread_mutex_unlock(&thread_count_mutex);
            bitonicSort(&left_data);
            bitonicSort(&right_data);
        }

        ThreadData merge_data = {data->arr, low, cnt, dir};
        bitonicMerge(&merge_data);
    }
    return nullptr;
}


int make_calculations(int n, int max_threads) {
    thread_pool_size = max_threads;
    thread_pool = new pthread_t[thread_pool_size];

    std::vector<int> arr(n);
    for (int i = 0; i < n; i++) {
        arr[i] = rand() % 1000;
    }

    std::cout << "Starting sorting array with length: " << n << "\n";
    std::cout << "Max threads: " << max_threads << "\n";

    auto start = std::chrono::high_resolution_clock::now();

    ThreadData initial_data = {&arr, 0, n, 1};
    bitonicSort(&initial_data);

    auto end = std::chrono::high_resolution_clock::now();
    std::chrono::duration<double> duration = end - start;

    std::cout << "Time taken: " << duration.count() << " seconds\n";

    for (int i = 1; i < n; i++) {
        if (arr[i] < arr[i-1]) {
            std::cout << "Sorting failed!\n";
            return 0;
        }
    }
    std::cout << "Sorting successful!\n";

    delete[] thread_pool;
    return 0;
}

int main(int argc, char* argv[]) {
    if (argc != 3) {
        std::cerr << "Usage: " << argv[0] << " <array_size> <max_threads>\n";
        return 1;
    }

    int n = std::atoi(argv[1]);
    max_threads = std::atoi(argv[2]);

    // Размер массива должен быть степенью 2
    if ((n & (n - 1)) != 0) {
        std::cerr << "Array size must be a power of 2\n";
        return 1;
    }

    make_calculations(n, max_threads);
```

```
    return 0;
}
```

Monitor_treads.sh

```
#!/bin/bash

# monitor_threads.sh

# Компилируем программу
g++ -pthread main.cpp -o bitonic_sort

# Запускаем программу в фоне
./bitonic_sort 2097152 4 &
PROG_PID=$!

echo "Program PID: $PROG_PID"
echo "Monitoring threads..."

# Мониторим каждую сотую секунды
while kill -0 $PROG_PID 2>/dev/null; do
    echo "$(date +%T%3N) - Number of threads: $(ls /proc/$PROG_PID/task | wc -l)"
    sleep 0.05
done

wait $PROG_PID
```

# Протокол работы программы

```
root@3520dd7dcbc8:/IdeaProjects/MAI_OS_Labs/Labs/Lab2/src# ./monitor_threads.sh
Program PID: 4406
Monitoring threads...
09:47:51421 - Number of threads: 1
Starting sorting array with length: 1048576
Max threads: 4
09:47:51480 - Number of threads: 5
09:47:51537 - Number of threads: 5
09:47:51595 - Number of threads: 5
09:47:51652 - Number of threads: 5
09:47:51708 - Number of threads: 5
09:47:51765 - Number of threads: 5
09:47:51825 - Number of threads: 5
09:47:51882 - Number of threads: 5
09:47:51939 - Number of threads: 5
09:47:51994 - Number of threads: 5
09:47:52050 - Number of threads: 5
09:47:52106 - Number of threads: 5
09:47:52162 - Number of threads: 5
09:47:52218 - Number of threads: 5
09:47:52274 - Number of threads: 5
09:47:52330 - Number of threads: 5
09:47:52386 - Number of threads: 4
09:47:52441 - Number of threads: 4
09:47:52497 - Number of threads: 4
09:47:52553 - Number of threads: 3
09:47:52609 - Number of threads: 4
09:47:52669 - Number of threads: 4
09:47:52726 - Number of threads: 5
```

```
09:47:52783 - Number of threads: 4
Time taken: 1.33603 seconds
Sorting successful!
```

**Strace:**

```
$ strace -f ./bitonic_sort 2048 4
```

```
root@3520dd7dcbc8:/IdeaProjects/MAI_OS_Labs/Labs/Lab2/src# strace -f ./bitonic_sort
2048 4
execve("./bitonic_sort", ["./bitonic_sort", "2048", "4"], 0x7fff8027ece8 /* 21 vars */)
= 0
brk(NULL)                               = 0xde3000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f9c996e2000
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=25258, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 25258, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f9c996db000
close(3)                                = 0
openat(AT_FDCWD, "/usr/local/lib64/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|0755, st_size=2530008, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 2543808, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f9c9946d000
mmap(0x7f9c99512000, 1216512, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0xa5000) = 0x7f9c99512000
mmap(0x7f9c9963b000, 581632, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1ce000) = 0x7f9c9963b000
mmap(0x7f9c996c9000, 57344, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x25c000) = 0x7f9c996c9000
mmap(0x7f9c996d7000, 12480, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7f9c996d7000
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"..., 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=907784, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 909560, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f9c9938e000
mmap(0x7f9c9939e000, 471040, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x10000) = 0x7f9c9939e000
mmap(0x7f9c99411000, 368640, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x83000) = 0x7f9c99411000
mmap(0x7f9c9946b000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0xdc000) = 0x7f9c9946b000
close(3)                                = 0
openat(AT_FDCWD, "/usr/local/lib64/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\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=906528, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 181160, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f9c99361000
mmap(0x7f9c99365000, 143360, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x4000) = 0x7f9c99365000
mmap(0x7f9c99388000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x27000)
= 0x7f9c99388000
mmap(0x7f9c9938c000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x2b000) = 0x7f9c9938c000
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\0\20t\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
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=1922136, ...}, 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, 1970000, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f9c99180000
mmap(0x7f9c991a6000, 1396736, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x26000) = 0x7f9c991a6000
mmap(0x7f9c992fb000, 339968, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x17b000) = 0x7f9c992fb000
mmap(0x7f9c9934e000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
```

```
3, 0x1ce000) = 0x7f9c9934e000
mmap(0x7f9c99354000, 53072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x7f9c99354000
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f9c9917e000
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f9c9917b000
arch_prctl(ARCH_SET_FS, 0x7f9c9917b740) = 0
set_tid_address(0x7f9c9917ba10)         = 4987
set_robust_list(0x7f9c9917ba20, 24)     = 0
rseq(0x7f9c9917c060, 0x20, 0, 0x53053053) = 0
mprotect(0x7f9c9934e000, 16384, PROT_READ) = 0
mprotect(0x7f9c9938c000, 4096, PROT_READ) = 0
mprotect(0x7f9c9946b000, 4096, PROT_READ) = 0
mprotect(0x7f9c996c9000, 45056, PROT_READ) = 0
mprotect(0x404000, 4096, PROT_READ)     = 0
mprotect(0x7f9c99714000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7f9c996db000, 25258)           = 0
futex(0x7f9c996d773c, FUTEX_WAKE_PRIVATE, 2147483647) = 0
getrandom("\x1b\x23\xaa\xcb\x56\x29\x7e\x02", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0xde3000
brk(0xe04000)                           = 0xe04000
newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0x1), ...},
AT_EMPTY_PATH) = 0
write(1, "Starting sorting array with leng"..., 41Starting sorting array with length:
2048
) = 41
write(1, "Max threads: 4\n", 15Max threads: 4
)        = 15
rt_sigaction(SIGRT_1, {sa_handler=0x7f9c992066a0, sa_mask=[],
sa_flags=SA_RESTORER|SA_ONSTACK|SA_RESTART|SA_SIGINFO, sa_restorer=0x7f9c991bc050},
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) =
0x7f9c9897a000
mprotect(0x7f9c9897b000, 8388608, PROT_READ|PROT_WRITE) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [QUIT], 8) = 0
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CL
ONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f9c9917a990,
parent_tid=0x7f9c9917a990, exit_signal=0, stack=0x7f9c9897a000, stack_size=0x7fff80,
tls=0x7f9c9917a6c0}strace: Process 4988 attached
 => {parent_tid=[4988]}, 88) = 4988
[pid  4988] rseq(0x7f9c9917afe0, 0x20, 0, 0x53053053 <unfinished ...>
[pid  4987] rt_sigprocmask(SIG_SETMASK, [QUIT],  <unfinished ...>
[pid  4988] <... rseq resumed>)         = 0
[pid  4987] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  4988] set_robust_list(0x7f9c9917a9a0, 24 <unfinished ...>
[pid  4987] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0
<unfinished ...>
[pid  4988] <... set_robust_list resumed>) = 0
[pid  4987] <... mmap resumed>)         = 0x7f9c98179000
[pid  4988] rt_sigprocmask(SIG_SETMASK, [QUIT],  <unfinished ...>
[pid  4987] mprotect(0x7f9c9817a000, 8388608, PROT_READ|PROT_WRITE <unfinished ...>
[pid  4988] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  4987] <... mprotect resumed>)     = 0
[pid  4988] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0
<unfinished ...>
[pid  4987] rt_sigprocmask(SIG_BLOCK, ~[],  <unfinished ...>
[pid  4988] <... mmap resumed>)         = 0x7f9c97978000
[pid  4987] <... rt_sigprocmask resumed>[QUIT], 8) = 0
[pid  4988] mprotect(0x7f9c97979000, 8388608, PROT_READ|PROT_WRITE <unfinished ...>
[pid  4987]
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CL
ONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f9c98979990,
parent_tid=0x7f9c98979990, exit_signal=0, stack=0x7f9c98179000, stack_size=0x7fff80,
```

```
tls=0x7f9c989796c0} <unfinished ...>
[pid  4988] <... mprotect resumed>)     = 0
[pid  4988] mmap(NULL, 134217728, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_NORESERVE, -
1, 0strace: Process 4989 attached
 <unfinished ...>
[pid  4987] <... clone3 resumed> => {parent_tid=[4989]}, 88) = 4989
[pid  4988] <... mmap resumed>)         = 0x7f9c8f978000
[pid  4987] rt_sigprocmask(SIG_SETMASK, [QUIT],  <unfinished ...>
[pid  4989] rseq(0x7f9c98979fe0, 0x20, 0, 0x53053053 <unfinished ...>
[pid  4987] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  4988] munmap(0x7f9c8f978000, 6848512 <unfinished ...>
[pid  4987] futex(0x7f9c9917a990, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 4988, NULL,
FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid  4989] <... rseq resumed>)         = 0
[pid  4988] <... munmap resumed>)       = 0
[pid  4989] set_robust_list(0x7f9c989799a0, 24 <unfinished ...>
[pid  4988] munmap(0x7f9c94000000, 60260352 <unfinished ...>
[pid  4989] <... set_robust_list resumed>) = 0
[pid  4988] <... munmap resumed>)       = 0
[pid  4989] rt_sigprocmask(SIG_SETMASK, [QUIT],  <unfinished ...>
[pid  4988] mprotect(0x7f9c90000000, 135168, PROT_READ|PROT_WRITE <unfinished ...>
[pid  4989] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  4988] <... mprotect resumed>)     = 0
[pid  4988] rt_sigprocmask(SIG_BLOCK, ~[], [QUIT], 8) = 0
[pid  4988]
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CL
ONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f9c98178990,
parent_tid=0x7f9c98178990, exit_signal=0, stack=0x7f9c97978000, stack_size=0x7fff80,
tls=0x7f9c981786c0}strace: Process 4990 attached
 <unfinished ...>
[pid  4990] rseq(0x7f9c98178fe0, 0x20, 0, 0x53053053 <unfinished ...>
[pid  4988] <... clone3 resumed> => {parent_tid=[4990]}, 88) = 4990
[pid  4990] <... rseq resumed>)         = 0
[pid  4988] rt_sigprocmask(SIG_SETMASK, [QUIT], NULL, 8) = 0
[pid  4990] set_robust_list(0x7f9c981789a0, 24 <unfinished ...>
[pid  4988] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0)
= 0x7f9c97177000
[pid  4988] mprotect(0x7f9c97178000, 8388608, PROT_READ|PROT_WRITE <unfinished ...>
[pid  4990] <... set_robust_list resumed>) = 0
[pid  4988] <... mprotect resumed>)     = 0
[pid  4989] rt_sigprocmask(SIG_BLOCK, ~[RT_1],  <unfinished ...>
[pid  4988] rt_sigprocmask(SIG_BLOCK, ~[],  <unfinished ...>
[pid  4990] rt_sigprocmask(SIG_SETMASK, [QUIT],  <unfinished ...>
[pid  4988] <... rt_sigprocmask resumed>[QUIT], 8) = 0
[pid  4989] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  4988]
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CL
ONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f9c97977990,
parent_tid=0x7f9c97977990, exit_signal=0, stack=0x7f9c97177000, stack_size=0x7fff80,
tls=0x7f9c979776c0}strace: Process 4991 attached
 <unfinished ...>
[pid  4990] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  4989] madvise(0x7f9c98179000, 8368128, MADV_DONTNEED <unfinished ...>
[pid  4991] rseq(0x7f9c97977fe0, 0x20, 0, 0x53053053 <unfinished ...>
[pid  4988] <... clone3 resumed> => {parent_tid=[4991]}, 88) = 4991
[pid  4991] <... rseq resumed>)         = 0
[pid  4988] rt_sigprocmask(SIG_SETMASK, [QUIT],  <unfinished ...>
[pid  4991] set_robust_list(0x7f9c979779a0, 24 <unfinished ...>
[pid  4988] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  4991] <... set_robust_list resumed>) = 0
[pid  4988] futex(0x7f9c98178990, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 4990, NULL,
FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid  4991] rt_sigprocmask(SIG_SETMASK, [QUIT], NULL, 8) = 0
[pid  4989] <... madvise resumed>)      = 0
[pid  4991] rt_sigprocmask(SIG_BLOCK, ~[RT_1],  <unfinished ...>
[pid  4990] futex(0x405300, FUTEX_WAKE_PRIVATE, 1 <unfinished ...>
```

```
[pid  4989] exit(0 <unfinished ...>
[pid  4991] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  4991] madvise(0x7f9c97177000, 8368128, MADV_DONTNEED <unfinished ...>
[pid  4990] <... futex resumed>)        = 0
[pid  4989] <... exit resumed>)         = ?
[pid  4991] <... madvise resumed>)      = 0
futex[pid  4991] <... exit resumed>)         = ?
[pid  4990] rt_sigprocmask(SIG_BLOCK, ~[RT_1],  <unfinished ...>
[pid  4991] +++ exited with 0 +++
[pid  4990] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  4990] madvise(0x7f9c97978000, 8368128, MADV_DONTNEED) = 0
[pid  4990] exit(0)                     = ?
[pid  4990] +++ exited with 0 +++
[pid  4988] <... futex resumed>)        = 0
[pid  4988] rt_sigprocmask(SIG_BLOCK, ~[RT_1], NULL, 8) = 0
[pid  4988] madvise(0x7f9c9897a000, 8368128, MADV_DONTNEED) = 0
[pid  4988] exit(0)                     = ?
[pid  4987] <... futex resumed>)        = 0
[pid  4988] +++ exited with 0 +++
rt_sigprocmask(SIG_BLOCK, ~[], [QUIT], 8) = 0
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CL
ONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7f9c98979990,
parent_tid=0x7f9c98979990, exit_signal=0, stack=0x7f9c98179000, stack_size=0x7fff80,
tls=0x7f9c989796c0}strace: Process 4992 attached
 => {parent_tid=[4992]}, 88) = 4992
[pid  4992] rseq(0x7f9c98979fe0, 0x20, 0, 0x53053053 <unfinished ...>
[pid  4987] rt_sigprocmask(SIG_SETMASK, [QUIT],  <unfinished ...>
[pid  4992] <... rseq resumed>)         = 0
[pid  4987] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  4992] set_robust_list(0x7f9c989799a0, 24) = 0
[pid  4992] rt_sigprocmask(SIG_SETMASK, [QUIT],  <unfinished ...>
[pid  4987] futex(0x7f9c98979990, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 4992, NULL,
FUTEX_BITSET_MATCH_ANY <unfinished ...>
[pid  4992] <... rt_sigprocmask resumed>NULL, 8) = 0
[pid  4992] rt_sigprocmask(SIG_BLOCK, ~[RT_1], NULL, 8) = 0
[pid  4992] madvise(0x7f9c98179000, 8368128, MADV_DONTNEED) = 0
[pid  4992] exit(0)                     = ?
[pid  4987] <... futex resumed>)        = 0
[pid  4992] +++ exited with 0 +++
write(1, "Time taken: 0.0109792 seconds\n", 30Time taken: 0.0109792 seconds
) = 30
write(1, "Sorting successful!\n", 20Sorting successful!
)   = 20
exit_group(0)                           = ?
+++ exited with 0 +++
```