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

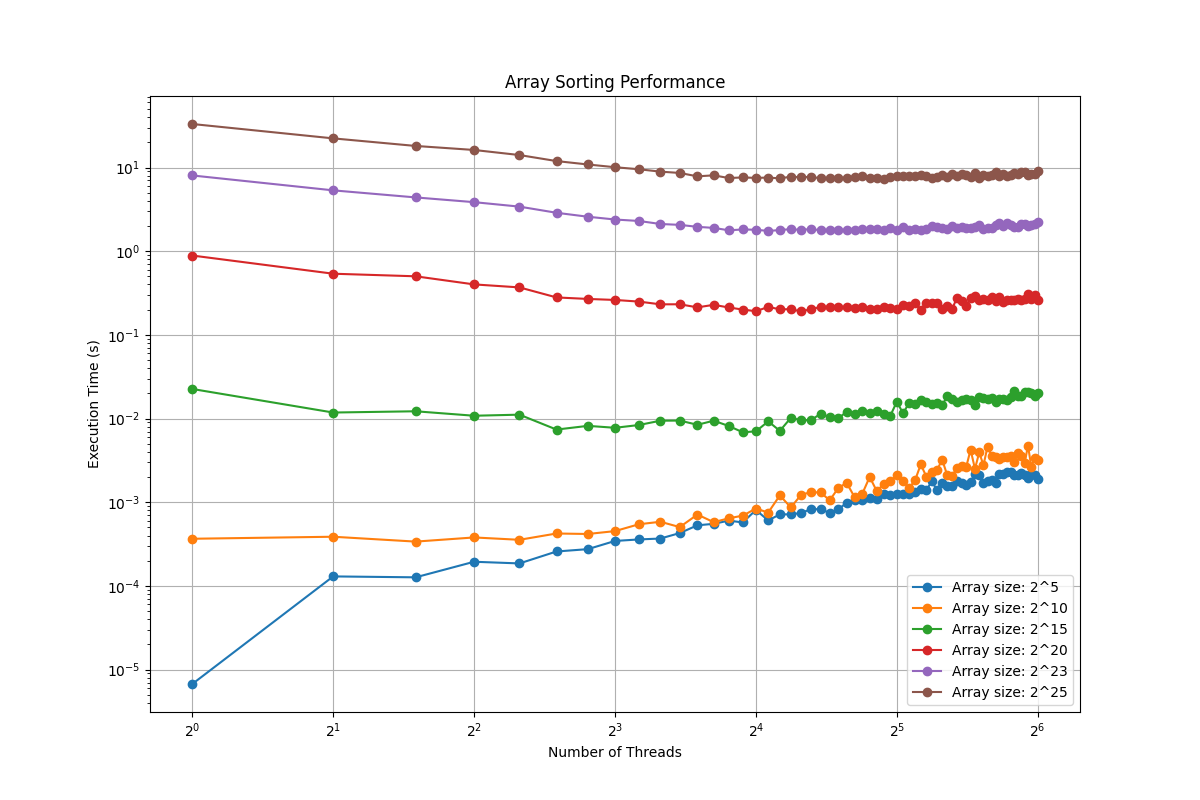


График для массива из 1024 элементов:

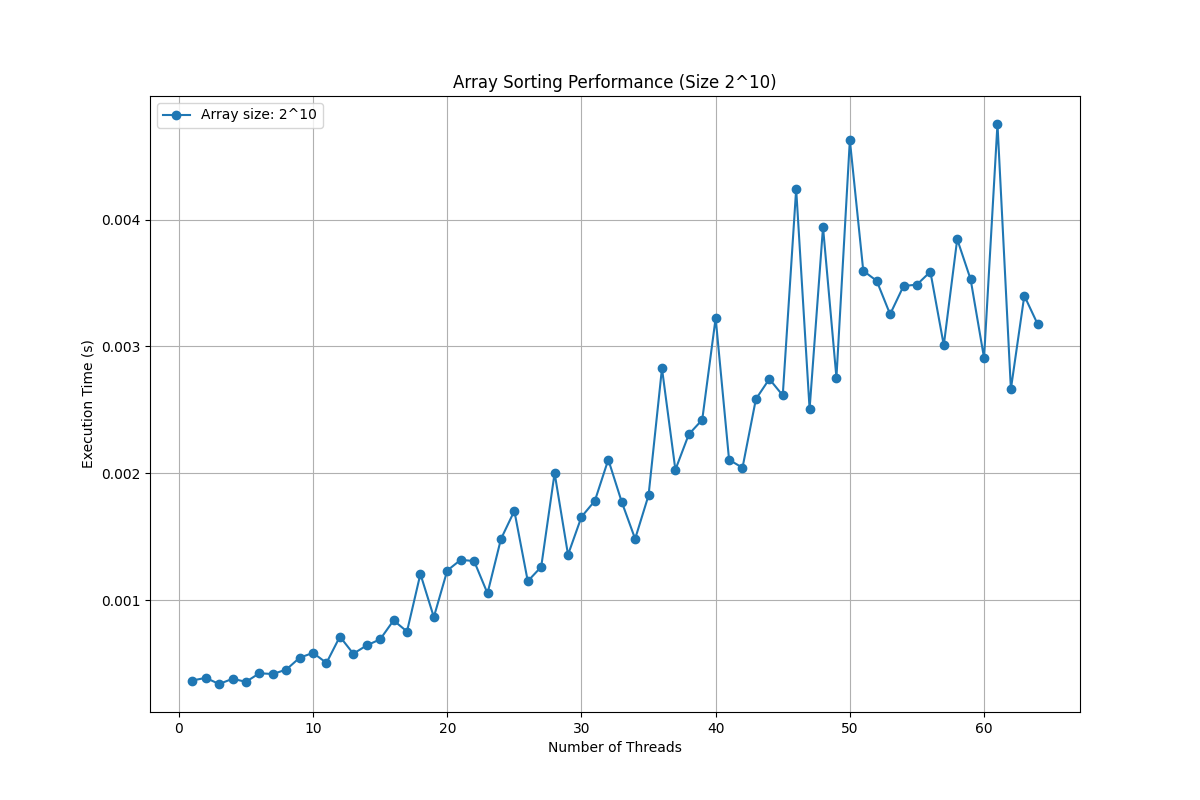


График для массива из 32\_768 элементов:

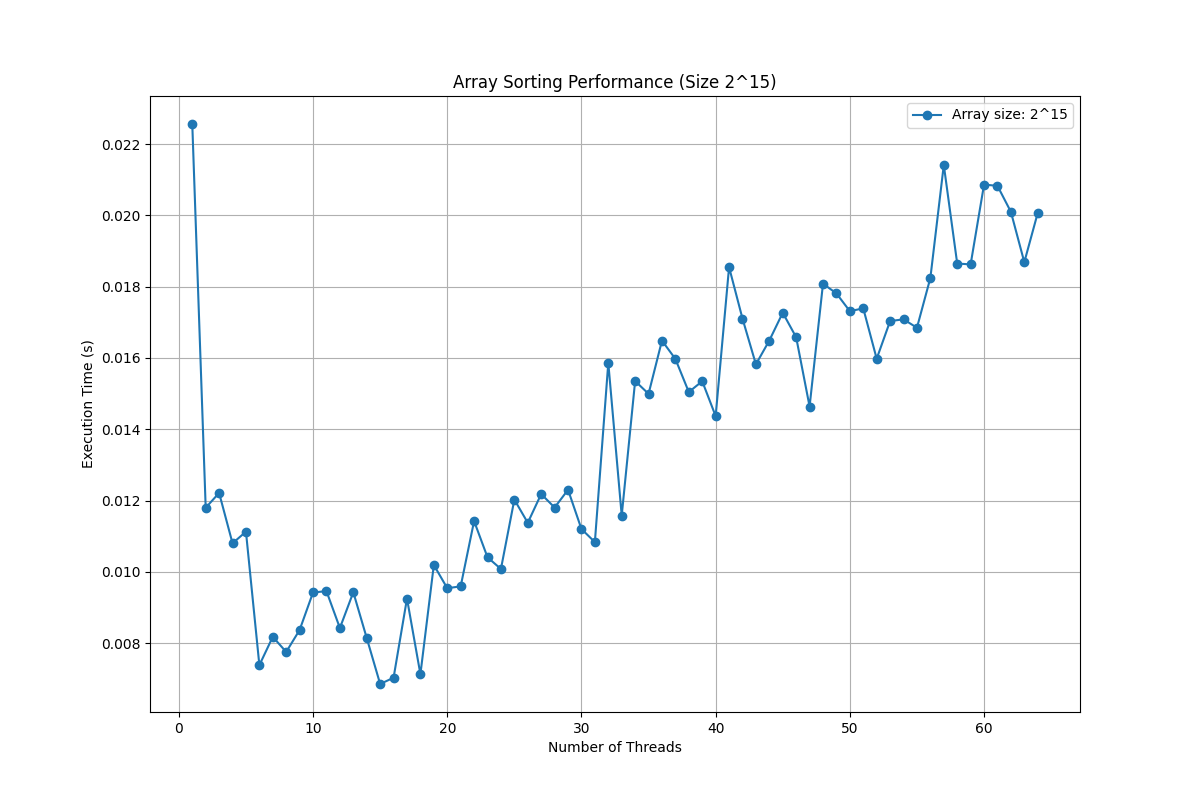


График для массива из 1\_048\_576 элементов:

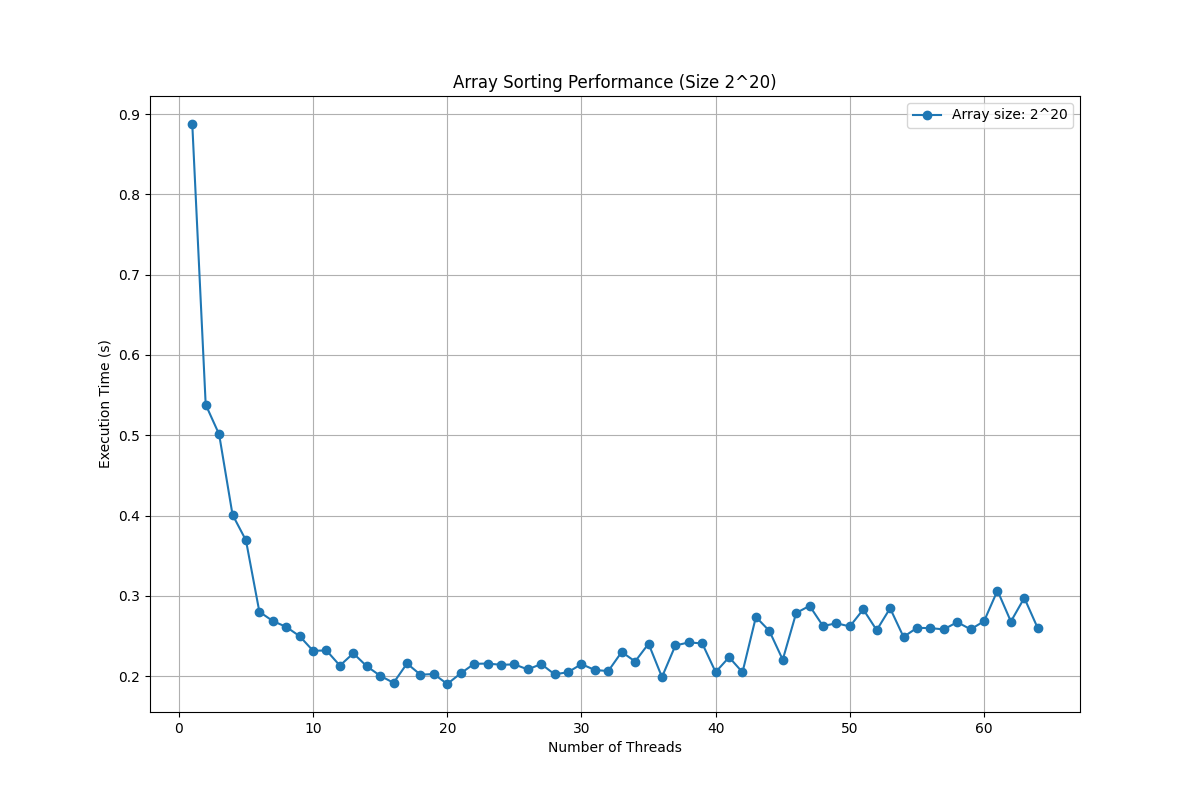


График для массива из 8\_388\_608 элементов:

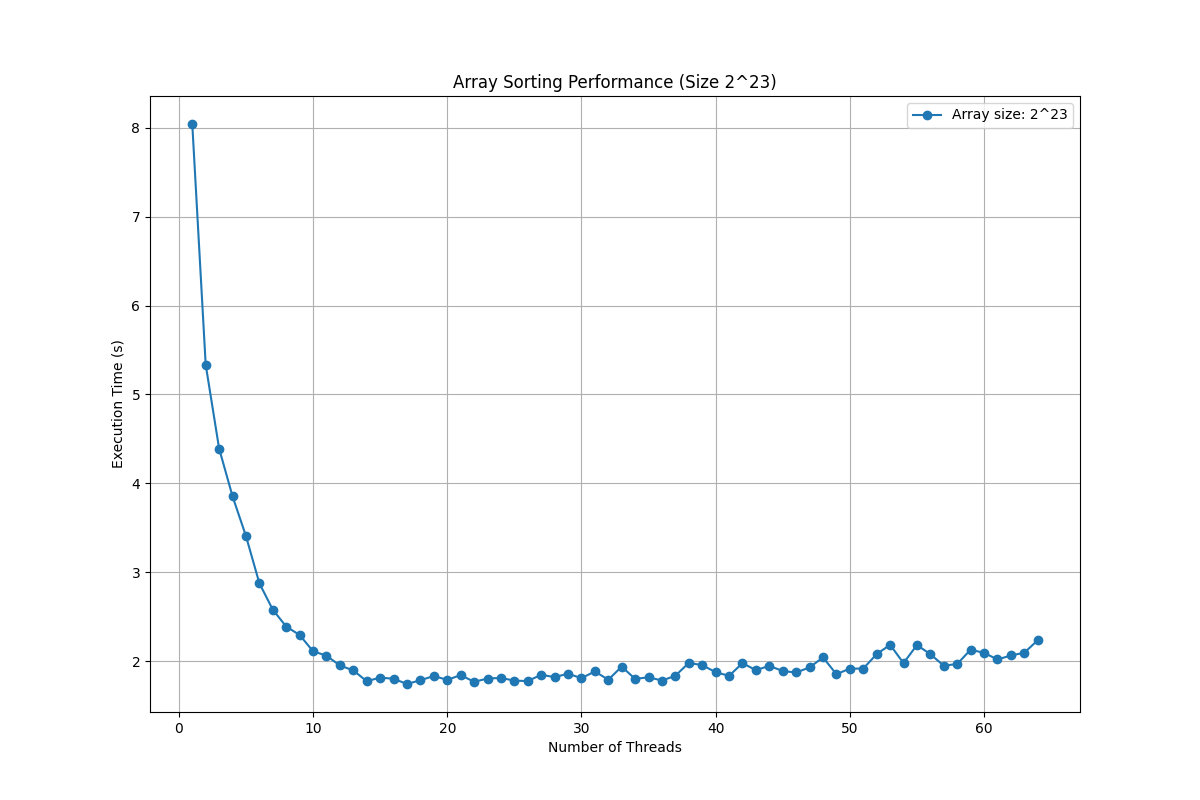
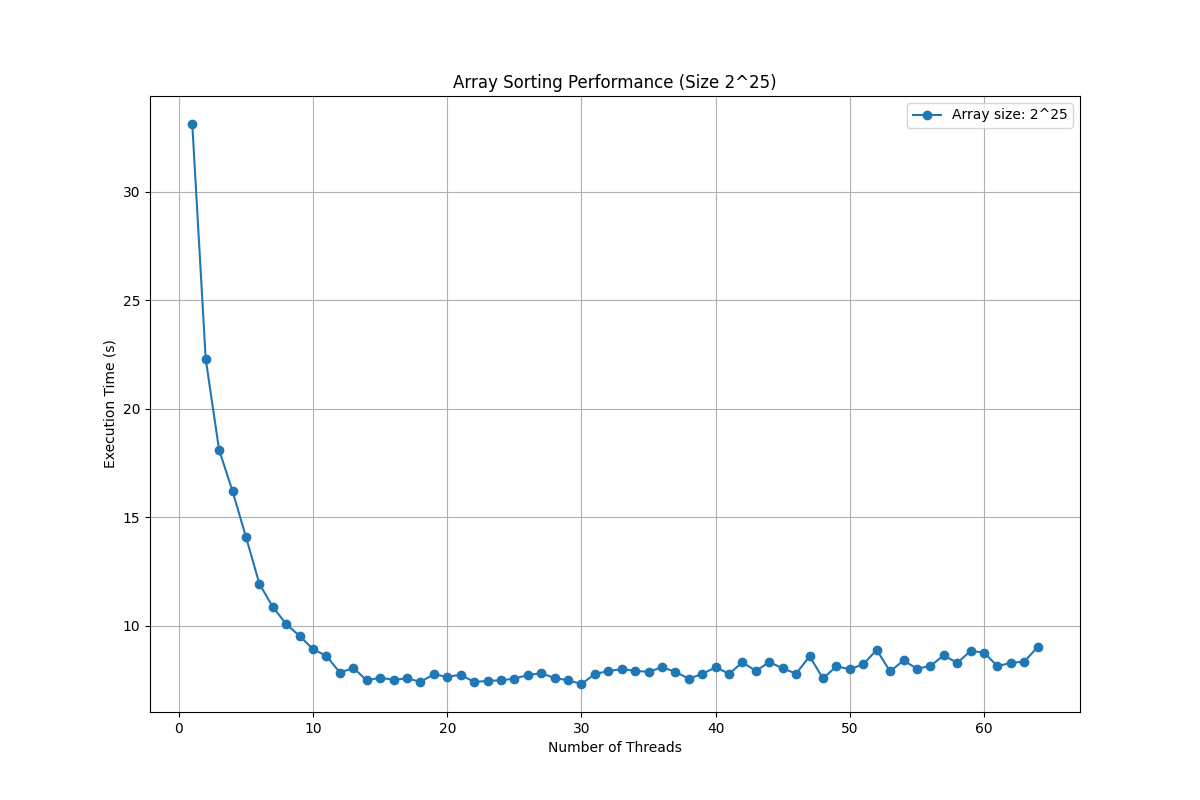
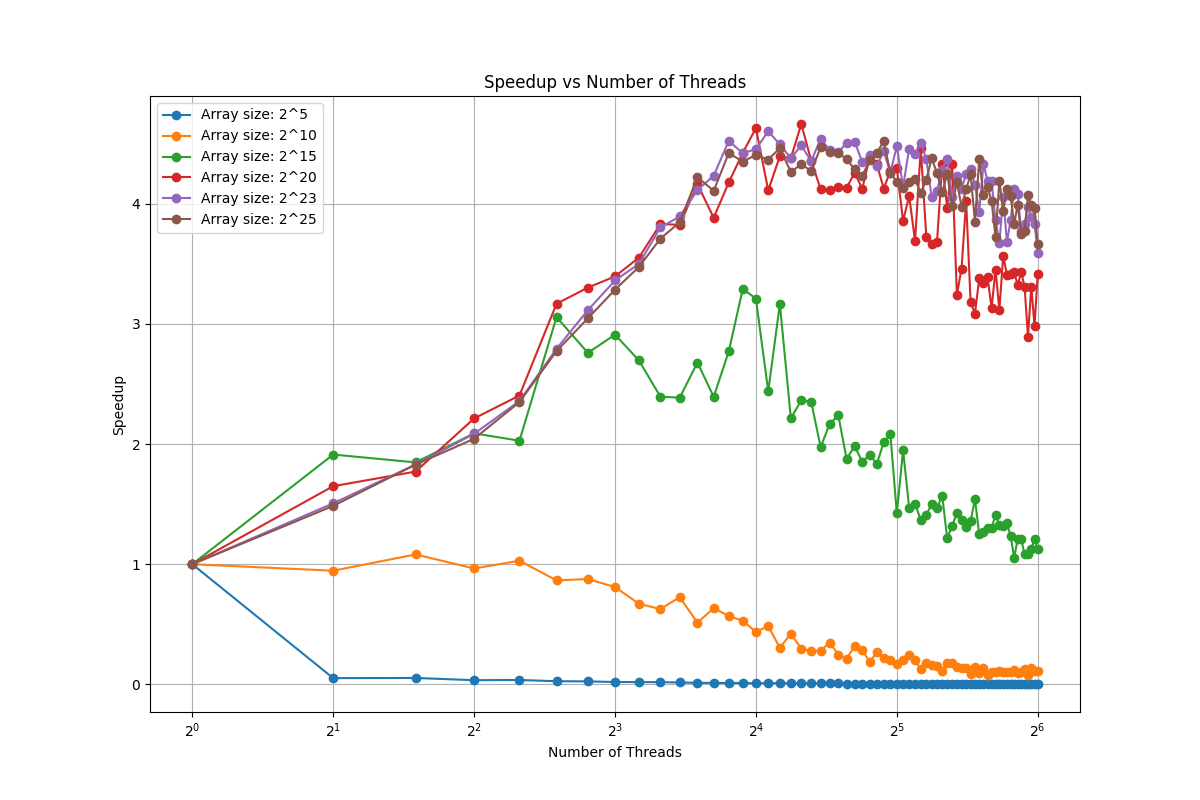


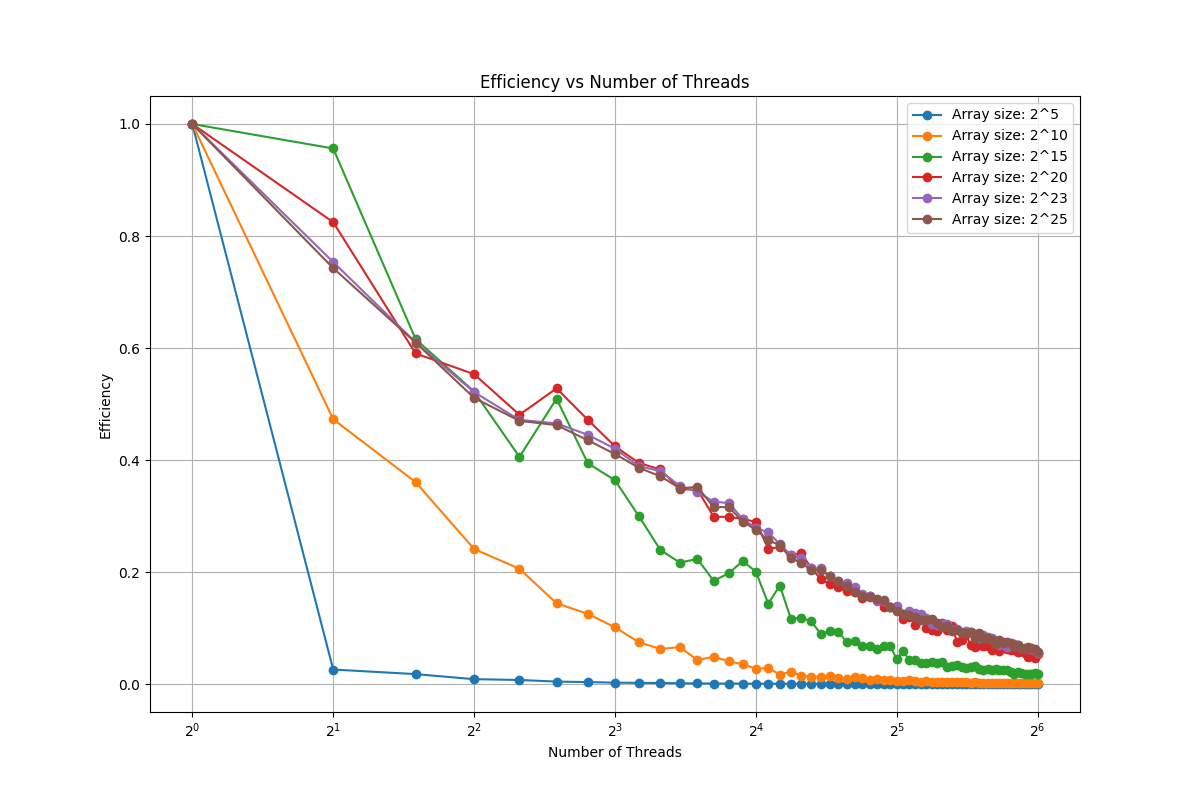
График для массива из 33\_554\_432 элементов:



Ускорение:



Эффективность:



**Маленькие массивы и :**

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 :**

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 и :**

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

Для 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 . Сначала ускорение растет, потом падает. То есть сначала увеличение потоков стоит того, а потом накладные расходы перевешивают.

**Эффективность:**

Интересно, что для того же эффективность для двух потоков самая хорошая

**Выводы:**

- Параллельная сортировка эффективна только для достаточно больших массивов ( 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

|  |
| --- |
| #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\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\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\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=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|CLONE\_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|CLONE\_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|CLONE\_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|CLONE\_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|CLONE\_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 +++