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: Самарский Я.В.

Преподаватель: Миронов Е.С.

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

Дата: 14.11.24

Москва, 2024

**Постановка задачи**

**Вариант 5.**

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 четно-нечетной сортировки Бетчера

**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\_init(&thread\_count\_mutex, nullptr); - инициализация мьютекса для синхронизации доступа к глобальной переменной current\_threads.
* pthread\_mutex\_lock(&thread\_count\_mutex); - блокировка мьютекса для безопасного доступа к переменной current\_threads.
* pthread\_mutex\_unlock(&thread\_count\_mutex); - разблокировка мьютекса после завершения доступа к переменной current\_threads.
* pthread\_mutex\_destroy(&thread\_count\_mutex); - уничтожение мьютекса после завершения работы программы.

Ключевые особенности:

1. Размер массива должен быть степенью двойки

2. Количество потоков также должно быть степенью двойки

3. Использует многопоточность через POSIX threads (pthread)

Работа:

1. Инициализация:

- Принимает два параметра: размер массива и максимальное число потоков

- Создается мьютекс для контроля доступа к числу потоков

- Создает случайный массив заданного размера

2. Четно-нечетная сортировка:

oddEvenMergeSort:

- Разделяет массив на две части

- Рекурсивно сортирует левую половину

- Рекурсивно сортирует правую половину

- После этого объединяет части используя bitonicMerge

oddEvenMerge:

- Рекурсивно сливает массив, увеличивая промежуток между сортируемыми элементами в два раза

- Рекурсивно обрабатывает получившиеся подпоследовательности

3. Управление потоками осуществляется с помощью счетчика потоков, доступ к которому контролируется с помощью мьютекса. oddEvenMergeSort запрашивает у счетчика возможность создать новый поток для сортировки левого подмассива.

4. Проверка результатов:

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

- Измеряется время выполнения сортировки

**Замеры эффективности**

Замеры проводились для 5 разных длин массивов. Количество потоков было от 1 до 64. Массив заполняется случайными значениями. Для каждого количества потоков проводилось 7 замеров, убирались минимальный и максимальный результат и бралось среднее значение времени.

На моем процессоре доступно 6 физических ядер и 12 логических ядер.

График со всеми замерами. Оси с логарифмическими шкалами

График ускорения (массив из 222 элементов):

График эффективности (массив из 222 элементов):

Ускорение показывает во сколько раз применение параллельного алгоритма уменьшает время решения задачи по сравнению с последовательным алгоритмом. Ускорение определяется величиной SN=T1/TN, где Т1 - время выполнения на одном потоке, TN - время выполнения на N потоках.

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

**Выводы:**

Параллельная сортировка имеет смысл на достаточно больших данных, где время на создание потоков компенсируется сэкономленным временем на сортировку. Число потоков больше кол-ва логических ядер также не даёт прироста производительности, а наоборот увеличивает время сортировки из-за того, что созданные потоки не могут выполняться одновременно.

**Код программы**

main.cpp

|  |
| --- |
| #include <iostream>  #include <vector>  #include <chrono>  #include "sort.h"  int main(int argc, char\* argv[]) {  if (argc != 3) {  std::cerr << "Usage: " << argv[0] << " <array\_size> <threads\_count>\n";  return 1;  }  int arraySize = std::atoi(argv[1]);  int threads = std::atoi(argv[2]);  // Размер массива должен быть степенью 2  if ((arraySize & (arraySize - 1)) != 0) {  std::cerr << "Array size must be a power of 2\n";  return 1;  }  std::vector<int> originalArray = createRandomValuesVector(arraySize);  std::vector<int> oeSorted = originalArray;  std::cout << "Starting sorting array with length: " << arraySize << "\n";  std::cout << "Max threads: " << threads << std::endl;  ThreadsLimiter threadsLimiter(threads);  auto start = std::chrono::high\_resolution\_clock::now();  oddEvenMergeSort(oeSorted, 0, oeSorted.size(), threadsLimiter);  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 < arraySize; i++) {  if (oeSorted[i] < oeSorted[i-1]) {  std::cout << "Sorting failed!\n";  return 0;  }  }  std::cout << "Sorting successful!\n";  return 0;  } |

sort.cpp

|  |
| --- |
| #include "sort.h"  void \*parallelSort(void \*u\_arg) {  auto \*arg = static\_cast<ParallelSortArg \*>(u\_arg);  oddEvenMergeSort(arg->vector, arg->left, arg->right, arg->threadsLimiter);  arg->threadsLimiter.releaseThread();  return nullptr;  }  void oddEvenMergeSort(std::vector<int> &a, int startIndex, int length, ThreadsLimiter& threadsLimiter) {  if (length <= 1)  return;  int halfLength = length / 2;  if (threadsLimiter.lockThread()) {  ParallelSortArg parallelArg{  a,  startIndex,  halfLength,  threadsLimiter  };  pthread\_t thread;  pthread\_create(&thread, nullptr, parallelSort, &parallelArg);  oddEvenMergeSort(a, startIndex + halfLength, halfLength, threadsLimiter);  pthread\_join(thread, nullptr);  } else {  oddEvenMergeSort(a, startIndex, halfLength, threadsLimiter);  oddEvenMergeSort(a, startIndex + halfLength, halfLength, threadsLimiter);  }  oddEvenMerge(a, startIndex, length, 1);  }  void oddEvenMerge(std::vector<int> &a, int startIndex, int length, int step) {  int doubleStep = step \* 2;  if (doubleStep < length) {  oddEvenMerge(a, startIndex, length, doubleStep);  oddEvenMerge(a, startIndex + step, length, doubleStep);  for (int i = startIndex + step; i + step < startIndex + length; i += doubleStep) {  compareAndExchange(a, i, i + step);  }  } else {  compareAndExchange(a, startIndex, startIndex + step);  }  }  void compareAndExchange(std::vector<int> &vector, int aIndex, int bIndex) {  if (vector[aIndex] > vector[bIndex]) {  std::swap(vector[aIndex], vector[bIndex]);  }  }  std::vector<int> createRandomValuesVector(size\_t size) {  std::vector<int> array(size);  for (size\_t i = 0; i < size; i++) {  array[i] = std::rand() % 1024;  }  return array;  }  bool ThreadsLimiter::lockThread() {  bool result = false;  pthread\_mutex\_lock(&mutex);  if (currentCount < maxCount) {  currentCount++;  result = true;  }  pthread\_mutex\_unlock(&mutex);  return result;  }  void ThreadsLimiter::releaseThread() {  pthread\_mutex\_lock(&mutex);  currentCount--;  pthread\_mutex\_unlock(&mutex);  }  ThreadsLimiter::ThreadsLimiter(int maxCount) : maxCount(maxCount) {  pthread\_mutex\_init(&mutex, nullptr);  }  ThreadsLimiter::~ThreadsLimiter() {  if (disposed)  return;  pthread\_mutex\_destroy(&mutex);  disposed = true;  } |

sort.h

|  |
| --- |
| #pragma once  #include <iostream>  #include <vector>  #include <pthread.h>  struct ThreadsLimiter {  int currentCount = 1; // Первый поток - основной (который использует этот класс)  int maxCount;  bool lockThread();  void releaseThread();  [[maybe\_unused]] explicit ThreadsLimiter(int maxCount);  ~ThreadsLimiter();  private:  pthread\_mutex\_t mutex{};  bool disposed = false;  };  void oddEvenMergeSort(std::vector<int> &a, int startIndex, int length, ThreadsLimiter& threadsLimiter);  void oddEvenMerge(std::vector<int> &a, int startIndex, int length, int step);  void compareAndExchange(std::vector<int> &vector, int aIndex, int bIndex);  std::vector<int> createRandomValuesVector(size\_t size);  struct ParallelSortArg {  std::vector<int> &vector;  int left;  int right;  ThreadsLimiter &threadsLimiter;  }; |

**Протокол работы программы**

user@DESKTOP-KC5QDB8:~/projects/mai\_os/lab2$ ./cmake-build-release/mai\_os 128 4

Starting sorting array with length: 128

Max threads: 4

Time taken: 0.000348404 seconds

Sorting successful!

**Strace:**

$ strace -f ./cmake-build-release/mai\_os 128 4

user@DESKTOP-KC5QDB8:~/projects/mai\_os/lab2$ strace -f ./cmake-build-release/mai\_os 128 4

execve("./cmake-build-release/mai\_os", ["./cmake-build-release/mai\_os", "128", "4"], 0x7ffcd5748a48 /\* 32 vars \*/) = 0

brk(NULL)                               = 0x55ea6f0f9000

arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7fffcf6d6330) = -1 EINVAL (Invalid argument)

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f96a202a000

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=18463, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 18463, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f96a2025000

close(3)                                = 0

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libstdc++.so.6", O\_RDONLY|O\_CLOEXEC) = 3

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

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=2260296, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 2275520, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f96a1df9000

mprotect(0x7f96a1e93000, 1576960, PROT\_NONE) = 0

mmap(0x7f96a1e93000, 1118208, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x9a000) = 0x7f96a1e93000

mmap(0x7f96a1fa4000, 454656, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1ab000) = 0x7f96a1fa4000

mmap(0x7f96a2014000, 57344, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x21a000) = 0x7f96a2014000

mmap(0x7f96a2022000, 10432, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f96a2022000

close(3)                                = 0

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libgcc\_s.so.1", O\_RDONLY|O\_CLOEXEC) = 3

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

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=125488, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 127720, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f96a1dd9000

mmap(0x7f96a1ddc000, 94208, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x3000) = 0x7f96a1ddc000

mmap(0x7f96a1df3000, 16384, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1a000) = 0x7f96a1df3000

mmap(0x7f96a1df7000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1d000) = 0x7f96a1df7000

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\0I\17\357\204\3$\f\221\2039x\324\224\323\236S"..., 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) = 0x7f96a1bb0000

mprotect(0x7f96a1bd8000, 2023424, PROT\_NONE) = 0

mmap(0x7f96a1bd8000, 1658880, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7f96a1bd8000

mmap(0x7f96a1d6d000, 360448, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1bd000) = 0x7f96a1d6d000

mmap(0x7f96a1dc6000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x215000) = 0x7f96a1dc6000

mmap(0x7f96a1dcc000, 52816, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f96a1dcc000

close(3)                                = 0

openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libm.so.6", O\_RDONLY|O\_CLOEXEC) = 3

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

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=940560, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 942344, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f96a1ac9000

mmap(0x7f96a1ad7000, 507904, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xe000) = 0x7f96a1ad7000

mmap(0x7f96a1b53000, 372736, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x8a000) = 0x7f96a1b53000

mmap(0x7f96a1bae000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xe4000) = 0x7f96a1bae000

close(3)                                = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f96a1ac7000

arch\_prctl(ARCH\_SET\_FS, 0x7f96a1ac83c0) = 0

set\_tid\_address(0x7f96a1ac8690)         = 125299

set\_robust\_list(0x7f96a1ac86a0, 24)     = 0

rseq(0x7f96a1ac8d60, 0x20, 0, 0x53053053) = 0

mprotect(0x7f96a1dc6000, 16384, PROT\_READ) = 0

mprotect(0x7f96a1bae000, 4096, PROT\_READ) = 0

mprotect(0x7f96a1df7000, 4096, PROT\_READ) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f96a1ac5000

mprotect(0x7f96a2014000, 45056, PROT\_READ) = 0

mprotect(0x55ea6e766000, 4096, PROT\_READ) = 0

mprotect(0x7f96a2064000, 8192, PROT\_READ) = 0

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

munmap(0x7f96a2025000, 18463)           = 0

getrandom("\xaa\x34\xe6\xa0\xb8\x62\x9d\x53", 8, GRND\_NONBLOCK) = 8

brk(NULL)                               = 0x55ea6f0f9000

brk(0x55ea6f11a000)                     = 0x55ea6f11a000

futex(0x7f96a202277c, FUTEX\_WAKE\_PRIVATE, 2147483647) = 0

newfstatat(1, "", {st\_mode=S\_IFCHR|0620, st\_rdev=makedev(0x88, 0x3), ...}, AT\_EMPTY\_PATH) = 0

write(1, "Starting sorting array with leng"..., 40Starting sorting array with length: 128

) = 40

write(1, "Max threads: 4\n", 15Max threads: 4

)        = 15

rt\_sigaction(SIGRT\_1, {sa\_handler=0x7f96a1c41870, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_ONSTACK|SA\_RESTART|SA\_SIGINFO, sa\_restorer=0x7f96a1bf2520}, 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) = 0x7f96a12c4000

mprotect(0x7f96a12c5000, 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=0x7f96a1ac4910, parent\_tid=0x7f96a1ac4910, exit\_signal=0, stack=0x7f96a12c4000, stack\_size=0x7fff00, tls=0x7f96a1ac4640}strace: Process 125300 attached

 => {parent\_tid=[125300]}, 88) = 125300

[pid 125300] rseq(0x7f96a1ac4fe0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 125299] rt\_sigprocmask(SIG\_SETMASK, [QUIT],  <unfinished ...>

[pid 125300] <... rseq resumed>)        = 0

[pid 125299] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 125300] set\_robust\_list(0x7f96a1ac4920, 24 <unfinished ...>

[pid 125299] mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0 <unfinished ...>

[pid 125300] <... set\_robust\_list resumed>) = 0

[pid 125299] <... mmap resumed>)        = 0x7f96a0ac3000

[pid 125300] rt\_sigprocmask(SIG\_SETMASK, [QUIT],  <unfinished ...>

[pid 125299] mprotect(0x7f96a0ac4000, 8388608, PROT\_READ|PROT\_WRITE <unfinished ...>

[pid 125300] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 125299] <... mprotect resumed>)    = 0

[pid 125300] mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0 <unfinished ...>

[pid 125299] rt\_sigprocmask(SIG\_BLOCK, ~[],  <unfinished ...>

[pid 125300] <... mmap resumed>)        = 0x7f96a02c2000

[pid 125299] <... rt\_sigprocmask resumed>[QUIT], 8) = 0

[pid 125300] mprotect(0x7f96a02c3000, 8388608, PROT\_READ|PROT\_WRITE <unfinished ...>

[pid 125299] clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, child\_tid=0x7f96a12c3910, parent\_tid=0x7f96a12c3910, exit\_signal=0, stack=0x7f96a0ac3000, stack\_size=0x7fff00, tls=0x7f96a12c3640} <unfinished ...>

[pid 125300] <... mprotect resumed>)    = 0

strace: Process 125301 attached

[pid 125300] mmap(NULL, 134217728, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_NORESERVE, -1, 0 <unfinished ...>

[pid 125299] <... clone3 resumed> => {parent\_tid=[125301]}, 88) = 125301

[pid 125301] rseq(0x7f96a12c3fe0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 125299] rt\_sigprocmask(SIG\_SETMASK, [QUIT],  <unfinished ...>

[pid 125300] <... mmap resumed>)        = 0x7f96982c2000

[pid 125299] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 125301] <... rseq resumed>)        = 0

[pid 125299] futex(0x7f96a12c3910, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 125301, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

[pid 125300] munmap(0x7f96982c2000, 64217088 <unfinished ...>

[pid 125301] set\_robust\_list(0x7f96a12c3920, 24 <unfinished ...>

[pid 125300] <... munmap resumed>)      = 0

[pid 125301] <... set\_robust\_list resumed>) = 0

[pid 125300] munmap(0x7f96a0000000, 2891776 <unfinished ...>

[pid 125301] rt\_sigprocmask(SIG\_SETMASK, [QUIT],  <unfinished ...>

[pid 125300] <... munmap resumed>)      = 0

[pid 125301] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 125300] mprotect(0x7f969c000000, 135168, PROT\_READ|PROT\_WRITE <unfinished ...>

[pid 125301] rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1],  <unfinished ...>

[pid 125300] <... mprotect resumed>)    = 0

[pid 125301] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 125300] rt\_sigprocmask(SIG\_BLOCK, ~[],  <unfinished ...>

[pid 125301] madvise(0x7f96a0ac3000, 8368128, MADV\_DONTNEED <unfinished ...>

[pid 125300] <... rt\_sigprocmask resumed>[QUIT], 8) = 0

[pid 125301] <... madvise resumed>)     = 0

[pid 125300] clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, child\_tid=0x7f96a0ac2910, parent\_tid=0x7f96a0ac2910, exit\_signal=0, stack=0x7f96a02c2000, stack\_size=0x7fff00, tls=0x7f96a0ac2640} <unfinished ...>

[pid 125301] exit(0strace: Process 125302 attached

)                    = ?

[pid 125302] rseq(0x7f96a0ac2fe0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 125301] +++ exited with 0 +++

[pid 125300] <... clone3 resumed> => {parent\_tid=[125302]}, 88) = 125302

[pid 125299] <... futex resumed>)       = 0

[pid 125302] <... rseq resumed>)        = 0

[pid 125299] futex(0x7f96a1ac4910, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 125300, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

[pid 125300] rt\_sigprocmask(SIG\_SETMASK, [QUIT],  <unfinished ...>

[pid 125302] set\_robust\_list(0x7f96a0ac2920, 24 <unfinished ...>

[pid 125300] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 125302] <... set\_robust\_list resumed>) = 0

[pid 125300] rt\_sigprocmask(SIG\_BLOCK, ~[],  <unfinished ...>

[pid 125302] rt\_sigprocmask(SIG\_SETMASK, [QUIT],  <unfinished ...>

[pid 125300] <... rt\_sigprocmask resumed>[QUIT], 8) = 0

[pid 125302] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 125300] clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, child\_tid=0x7f96a12c3910, parent\_tid=0x7f96a12c3910, exit\_signal=0, stack=0x7f96a0ac3000, stack\_size=0x7fff00, tls=0x7f96a12c3640} <unfinished ...>

[pid 125302] rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], strace: Process 125303 attached

NULL, 8) = 0

[pid 125303] rseq(0x7f96a12c3fe0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 125302] madvise(0x7f96a02c2000, 8368128, MADV\_DONTNEED <unfinished ...>

[pid 125303] <... rseq resumed>)        = 0

[pid 125300] <... clone3 resumed> => {parent\_tid=[125303]}, 88) = 125303

[pid 125303] set\_robust\_list(0x7f96a12c3920, 24 <unfinished ...>

[pid 125302] <... madvise resumed>)     = 0

[pid 125303] <... set\_robust\_list resumed>) = 0

[pid 125300] rt\_sigprocmask(SIG\_SETMASK, [QUIT],  <unfinished ...>

[pid 125303] rt\_sigprocmask(SIG\_SETMASK, [QUIT],  <unfinished ...>

[pid 125302] exit(0 <unfinished ...>

[pid 125303] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 125300] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 125303] mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0 <unfinished ...>

[pid 125302] <... exit resumed>)        = ?

[pid 125303] <... mmap resumed>)        = 0x7f969b7ff000

[pid 125300] futex(0x7f96a12c3910, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 125303, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

[pid 125303] mprotect(0x7f969b800000, 8388608, PROT\_READ|PROT\_WRITE <unfinished ...>

[pid 125302] +++ exited with 0 +++

[pid 125303] <... mprotect resumed>)    = 0

[pid 125303] mmap(NULL, 134217728, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_NORESERVE, -1, 0) = 0x7f96937ff000

[pid 125303] munmap(0x7f96937ff000, 8392704) = 0

[pid 125303] munmap(0x7f9698000000, 58716160) = 0

[pid 125303] mprotect(0x7f9694000000, 135168, PROT\_READ|PROT\_WRITE) = 0

[pid 125303] rt\_sigprocmask(SIG\_BLOCK, ~[], [QUIT], 8) = 0

[pid 125303] clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, child\_tid=0x7f969bfff910, parent\_tid=0x7f969bfff910, exit\_signal=0, stack=0x7f969b7ff000, stack\_size=0x7fff00, tls=0x7f969bfff640}strace: Process 125304 attached

 <unfinished ...>

[pid 125304] rseq(0x7f969bffffe0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 125303] <... clone3 resumed> => {parent\_tid=[125304]}, 88) = 125304

[pid 125304] <... rseq resumed>)        = 0

[pid 125303] rt\_sigprocmask(SIG\_SETMASK, [QUIT],  <unfinished ...>

[pid 125304] set\_robust\_list(0x7f969bfff920, 24 <unfinished ...>

[pid 125303] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 125304] <... set\_robust\_list resumed>) = 0

[pid 125303] futex(0x7f969bfff910, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 125304, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

[pid 125304] rt\_sigprocmask(SIG\_SETMASK, [QUIT], NULL, 8) = 0

[pid 125304] rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], NULL, 8) = 0

[pid 125304] madvise(0x7f969b7ff000, 8368128, MADV\_DONTNEED) = 0

[pid 125304] exit(0)                    = ?

[pid 125304] +++ exited with 0 +++

[pid 125303] <... futex resumed>)       = 0

[pid 125303] rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], NULL, 8) = 0

[pid 125303] madvise(0x7f96a0ac3000, 8368128, MADV\_DONTNEED) = 0

[pid 125303] exit(0)                    = ?

[pid 125303] +++ exited with 0 +++

[pid 125300] <... futex resumed>)       = 0

[pid 125300] rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], NULL, 8) = 0

[pid 125300] madvise(0x7f96a12c4000, 8368128, MADV\_DONTNEED) = 0

[pid 125300] exit(0)                    = ?

[pid 125300] +++ exited with 0 +++

<... futex resumed>)                    = 0

write(1, "Time taken: 0.00536197 seconds\n", 31Time taken: 0.00536197 seconds

) = 31

write(1, "Sorting successful!\n", 20Sorting successful!

)   = 20

exit\_group(0)                           = ?

+++ exited with 0 +++