# Задача №1

Необходимо написать программу, которая будет сортировать по возрастанию большой файл беззнаковых 32-х разрядных целых чисел. При этом:

- Размер входного файла значительно больше объема доступной оперативной памяти, которой есть всего 128 Мб.

- Числа в файле записаны в бинарном виде.

- Есть достаточно дискового пространства для хранения результата сортировки в отдельном файле и для хранения промежуточных результатов.

- Программа будет компилироваться при помощи g++-5.3.0 с опциями -std=c++14 -D\_NDEBUG -O3 -lpthread.

- Файлы будут находиться на SSD диске. А компьютере стоит 2-х ядерный процессор.

- Входной файл будет находиться в той же директории что и исполняемый файл и будет называться input. Мы ожидаем там же увидеть отсортированный файл с именем output.

- Из вашей программы должен получиться исполняемый файл, а значит нужно чтобы в ней была функция main.

- Никаких дополнительных библиотек на компьютере не установлено. (Например, нет boost).

- Решение должно быть кросс-платформенным.[[1]](#footnote-1) [[2]](#footnote-2)

- Входной файл менять нельзя.

- Для того, чтобы у всех были равные условия, мы 2 раза запустим сортировку вхолостую, а затем еще 3 раза. Измерим время, за которое файл был отсортирован в течении каждой из 3-х попыток и усредним его.

- В качестве решения можно загрузить или один cpp-файл, или zip-архив c h- или cpp-файлами.

Система для проверки решения находится по адресу: [http://cpp-test.artec3d-cloud.com](http://cpp-test.artec3d-cloud.com/). Пожалуйста, не обращайте особого внимания на начисляемые очки, Вам достаточно сделать решение, которое проходит все тесты в рамках отведенного периода времени. Если у Вас возникнут какие-то проблемы с системой проверки, то обращайтесь, пожалуйста, к нам.

В дополнение ко всему вышесказанному, мы хотели бы попросить Вас оценить время, которое потребуется Вам на решение задачи и сообщить нам когда ждать от Вас результат. В качестве результата достаточно сообщить нам, что Вы закончили.

Спасибо и успеха!

# Learning

с++ кроссплатформенный   
[https://www.google.ru/search?q=%D1%81%2B%2B+%D0%BA%D1%80%D0%BE%D1%81%D1%81%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9&oq=%D1%81%2B%2B+%D0%BA%D1%80%D0%BE&aqs=chrome.4.69i57j69i65l3j0l2.5515j0j7&sourceid=chrome&ie=UTF-8](https://www.google.ru/search?q=с%2B%2B+кроссплатформенный&oq=с%2B%2B+кро&aqs=chrome.4.69i57j69i65l3j0l2.5515j0j7&sourceid=chrome&ie=UTF-8)

Кроссплатформенность   
[https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%BE%D1%81%D1%81%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D1%8C](https://ru.wikipedia.org/wiki/Кроссплатформенность)

POSIX <https://ru.wikipedia.org/wiki/POSIX>

g++ cross compile linux windows <https://www.google.ru/search?q=g%2B%2B+cross+compile+linux+windows&oq=g%2B%2B+cross&aqs=chrome.5.0j69i57j0l4.24790j0j7&sourceid=chrome&ie=UTF-8>

Cross-Compiling Under Linux <https://wiki.wxwidgets.org/Cross-Compiling_Under_Linux>

Install The Mingw Cross-Compiler <https://wiki.wxwidgets.org/Install_The_Mingw_Cross-Compiler>

Cross-compiling on Windows for Linux <http://metamod-p.sourceforge.net/cross-compiling.on.windows.for.linux.html>

Check Углубленное программирование на C/C++ <https://stepik.org/course/153/syllabus?module=1>

lpthread g++ <https://www.google.ru/search?q=lpthread+g%2B%2B&oq=lpthread+g%2B%2B&aqs=chrome..69i57j0l5.14222j0j7&sourceid=chrome&ie=UTF-8>

Difference between -pthread and -lpthread while compiling <https://stackoverflow.com/questions/23250863/difference-between-pthread-and-lpthread-while-compiling>

How to include -std=c++11 and -lpthread in makefile? <https://stackoverflow.com/questions/33689044/how-to-include-std-c11-and-lpthread-in-makefile>

with -lpthread, g++ compiler error, “undefined reference to ” semaphore calls such as `sem\_open' <https://stackoverflow.com/questions/11111291/with-lpthread-g-compiler-error-undefined-reference-to-semaphore-calls-su>

parallel programming c++ example   
<https://www.google.ru/search?q=parallel+programming+c%2B%2B+example&oq=parallel+programming+C%2B%2B&aqs=chrome.1.69i57j0l5.16905j0j7&sourceid=chrome&ie=UTF-8>

C++ Examples of Parallel Programming with OpenMP <https://people.sc.fsu.edu/~jburkardt/cpp_src/openmp/openmp.html>

parallel programming c++ <https://www.google.ru/search?q=parallel+programming+c%2B%2B&oq=parallel+programming+c%2B%2B&aqs=chrome..69i57j69i59j0l4.2966j0j7&sourceid=chrome&ie=UTF-8>

Многопоточность [https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D0%BF%D0%BE%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D1%81%D1%82%D1%8C](https://ru.wikipedia.org/wiki/Многопоточность)

Multithreading (computer architecture) <https://en.wikipedia.org/wiki/Multithreading_(computer_architecture)>

# [многопоточное программирование c++](https://www.google.ru/search?q=многопоточное+программирование+c%2B%2B&oq=многопоточное+программирование+&aqs=chrome.4.69i57j69i60j69i61l2j0l2.12414j0j7&sourceid=chrome&ie=UTF-8)

## [Изучение многопоточного программирования](https://ru.stackoverflow.com/questions/202738/Изучение-многопоточного-программирования)

Наверное, надо понять, что такое потоки (threads) вообще, как они создаются, как их можно прерывать, как они заканчиваются. Далее самое сложное - механизмы синхронизации: всякие мьютексы, семафоры, барьеры, условные переменные. Для начала разберите и реализуйте паттерн "производитель-потребитель" (producent-consumer). Тогда многое станет ясно.

Лучше, наверное, начать с низкоуровневых реализаций, с уровня API системы. Разные boost.Thread, C++11 Thread, Qt::Thread - ООП-обертки, удобные и более безопасные, но довольно прозрачно отражающие низкоуровневые принципы реализации.

Что почитать... Обычно в книгах по API ОС все это описывается. По общим вопросам у Танненбаума в книге по ОС есть довольно неплохой раздел. Есть книжка по C++11 Thread, в том числе на русском.

### [producer-consumer C++](https://www.google.ru/search?q=producent-consumer+C%2B%2B&oq=producent-consumer+C%2B%2B&aqs=chrome..69i57j0l5.4670j0j7&sourceid=chrome&ie=UTF-8)

[производитель-потребитель C++](https://www.google.ru/search?q=производитель-потребитель+C%2B%2B&oq=производитель-потребитель+C%2B%2B&aqs=chrome..69i57j0l2.10637j0j7&sourceid=chrome&ie=UTF-8)

[3.6. Типовые задачи синхронизации](http://www.intuit.ru/studies/courses/4447/983/lecture/14923?page=4#sect14), [3.6.1. Задача "Производители-потребители"](http://www.intuit.ru/studies/courses/4447/983/lecture/14923?page=4#sect15)[[3]](#endnote-1)

[Многопоточное программирование. 3. Проблема производителя и потребителя](http://precious-cpp.blogspot.ru/2011/01/blog-post.html)

[Решение задачи производитель потребитель](http://life-prog.ru/view_os.php?id=50)

[Классические задачи синхронизации](http://itcentre.ru/programming/science-work/parallel-programming/368/)

[priority thread c++](https://www.google.ru/search?q=priority+thread+c%2B%2B&oq=priority+thread+c%2B%2B&aqs=chrome..69i57j0l5.5319j0j7&sourceid=chrome&ie=UTF-8)

[C++11 non-blocking producer/consumer](https://stackoverflow.com/questions/21414933/c11-non-blocking-producer-consumer)

I have a C++11 application with a high-priority thread that's producing data, and a low-priority thread that's consuming it (in my case, writing it to disk). I'd like to make sure the high-priority producer thread is never blocked, i.e. it uses only lock-free algorithms.

With a lock-free queue, I can push data to the queue from the producer thread, and poll it from the consumer thread, thus meeting my goals above. I'd like to modify my program so that the consumer thread blocks when inactive instead of polling.

[Producer–consumer problem](https://en.wikipedia.org/wiki/Producer–consumer_problem)

[A multi-threaded Producer Consumer with C++11](https://codereview.stackexchange.com/questions/84109/a-multi-threaded-producer-consumer-with-c11)

# [Углубленное программирование на C/C++](https://stepik.org/course/153/syllabus)

## [1.3 Организация и использование оперативной памяти. Часть 2 - 13](https://stepik.org/lesson/29438/step/13?unit=10323)

6:00 – Что ещё можно оптимизировать: Even/odd array elements.

[Векторизация вычислений](http://ssd.sscc.ru/sites/default/files/content/attach/317/lecture2014_10_vectorization.pdf)

## [1.4 Основы многопоточного программирования. Вопросы качества кода - 1](https://stepik.org/lesson/28284/step/1?course=Углубленное-программирование-на-CC%2B%2B&unit=9472)

0:18 – if else: the positive condition is first.

3:56 – if() vs. if(!)

[1.4 Основы многопоточного программирования. Вопросы качества кода - 6](https://stepik.org/lesson/28284/step/6?course=Углубленное-программирование-на-CC%2B%2B&unit=9472)

[Data segment](https://en.wikipedia.org/wiki/Data_segment)

[.bss](https://en.wikipedia.org/wiki/.bss)

## [1.4 Основы многопоточного программирования. Вопросы качества кода - 10](https://stepik.org/lesson/28284/step/10?course=Углубленное-программирование-на-CC%2B%2B&unit=9472)

Элементы многопоточного программирования: пример (1 / 2) - 10:00

include <pthread.h>

void pthread\_routine(void \*arg) {…}

## [1.4 Основы многопоточного программирования. Вопросы качества кода - 12](https://stepik.org/lesson/28284/step/12?course=Углубленное-программирование-на-CC%2B%2B&unit=9472)

03:43 – Взаимные исключения и ситуация гонок

08:29 – операция «закрыть» (условный, или неблокирующий вариант); см. 18:40 – Условные переменные.

09:41 – Обобщением мьютекса является семафор Дейкстры, реализованный в Pthreads как объект типа sem\_t.

[Синхронизация с помощью семафоров](http://wm-help.net/lib/b/book/1696396857/296)

[Книга: Основы программирования в Linux](http://wm-help.net/lib/b/book/1696396857/296)

15:51 – «попытаться и откатить»

[Аномалии многопоточных программ](https://stepik.org/lesson/Синхронизация-потоков-13490/step/10?unit=3646)

*22:07 – SUSPENDED to review* [*1.4 Основы многопоточного программирования. Вопросы качества кода - 12*](https://stepik.org/lesson/28284/step/12?course=Углубленное-программирование-на-CC%2B%2B&unit=9472)*.*

# [Многопоточное программирование на С/С++](https://stepik.org/course/149/syllabus?module=6)

My screenshots and abstracts of the course.

# My multi-threading examples.

## [memory\_order\_acquire](http://www.cplusplus.com/reference/atomic/memory_order/?kw=memory_order_acquire)

### [Lock-free структуры данных. Основы: Модель памяти](https://habrahabr.ru/post/197520/)

[Lock-free структуры данных. Извне: введение в libcds](https://habrahabr.ru/post/196834/)

[POSIX Threads](https://www.google.ru/search?q=POSIX+Threads&oq=POSIX+Threads&aqs=chrome..69i57j69i65j0l4.10143j0j7&sourceid=chrome&ie=UTF-8)

[POSIX thread (pthread) libraries](http://www.cs.cmu.edu/afs/cs/academic/class/15492-f07/www/pthreads.html)

[C++ Multithreading](https://www.tutorialspoint.com/cplusplus/cpp_multithreading.htm)

[Multithreaded Programming (POSIX pthreads Tutorial)](https://randu.org/tutorials/threads/)

[POSIX Threads Programming](https://computing.llnl.gov/tutorials/pthreads/)

[lock free multithreading](https://www.google.ru/search?q=lock+free+multithreading&oq=lock&aqs=chrome.5.69i57j69i61j69i60j69i65l2j35i39.6529j0j7&sourceid=chrome&ie=UTF-8)

[An Introduction to Lock-Free Programming](http://preshing.com/20120612/an-introduction-to-lock-free-programming/)

[Lockless Programming Considerations for Xbox 360 and Microsoft Windows](https://msdn.microsoft.com/en-us/library/windows/desktop/ee418650(v=vs.85).aspx)

[Part X. Parallel Programming](https://theboostcpplibraries.com/parallel-programming)

[Chapter 46. Boost.Lockfree](https://theboostcpplibraries.com/boost.lockfree)

[Programming with POSIX® Threads](file:///G:\_\_17\My_Documents\Miscellaneous\_18\Projects\C++\Examples\C++\Multithreading\Learning\butenhof\book\Butenhof_Programming_with_POSIX%25C2%25AE_Threads.html)\4.1 Pipeline

1. 1) Linux: epoll, FreeBSD: kequeue 2) #ifder … #else … #endif [https://stepik.org/lesson/12581/step/1?course=%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D0%BF%D0%BE%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D0%B5-%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5-%D0%BD%D0%B0-%D0%A1%D0%A1%2B%2B&unit=3008](https://stepik.org/lesson/12581/step/1?course=Многопоточное-программирование-на-СС%2B%2B&unit=3008) [↑](#footnote-ref-1)
2. 1) New process in Linux: fork. 2) What about std::thread in Linux & Win? [↑](#footnote-ref-2)
3. The OS standpoint will give me a deeper understanding. [↑](#endnote-ref-1)