# Лабораторная работа №3

## **Задание 1**

### **Необходимые знания**

**1. Аргументы командной строки**

Аргументы командной строки — это необязательные строковые аргументы, передаваемые операционной системой в программу при её запуске. Программа может их использовать в качестве входных данных, либо игнорировать. Подобно тому, как параметры одной функции предоставляют данные для параметров другой функции, так и аргументы командной строки предоставляют возможность людям или программам предоставлять входные данные для программы.

Чтобы принять аргументы командной строки, используются два специальных встроенных аргумента: argc и argv.

Int main (int argc, char \*argv[]) argc (англ. «argument count» = «количество аргументов») — это целочисленный параметр, содержащий количество аргументов, переданных в программу. argc всегда будет как минимум 1, так как первым аргументом всегда является имя самой программы.

Каждый аргумент командной строки, который предоставляет пользователь, заставит argc увеличиться на 1.

Argv (англ. «argument values» = «значения аргументов») — это место, где хранятся фактические значения аргументов. Длина этого массива — argc.

**2.Сборка с помощью gcc (clang)**

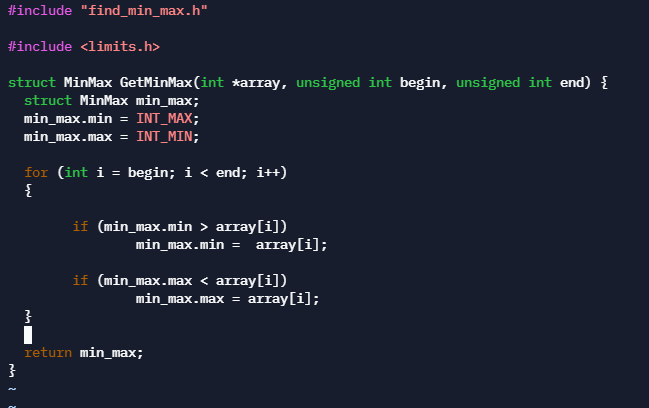
Gcc– это свободно доступный оптимизирующий компилятор для языков C, C++.

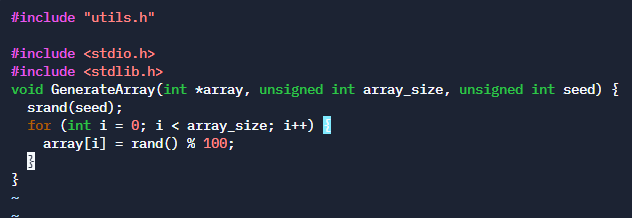
Программа gcc, запускаемая из командной строки, представяляет собой надстройку над группой компиляторов.

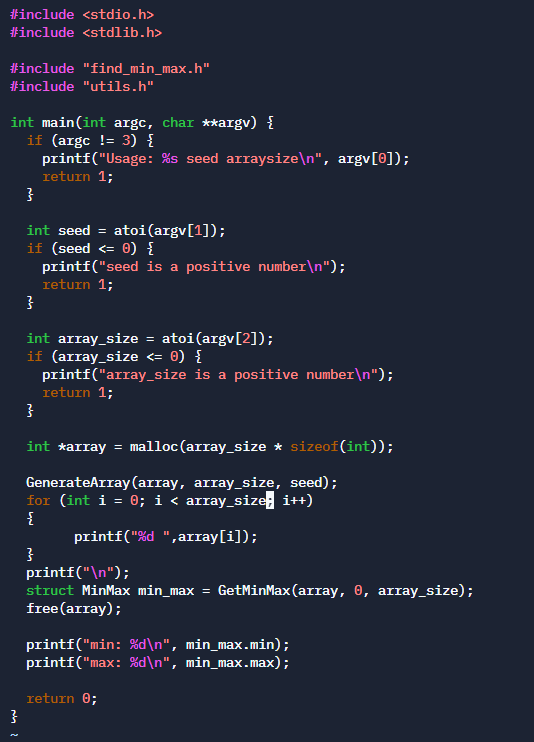
В зависимости от расширений имен файлов, передаваемых в качестве параметров, и дополнительных опций, gcc  запускает необходимые препроцессоры, компиляторы, линкеры.

Clang может использоваться как прямая замена gcc и предлагает несколько крутых инструментов статического анализа.

*Написать функцию GetMinMax в find\_max\_min.c, которая ищет минимальный и максимальный элементы массива, на заданном промежутке. Разобраться, что делает программа в sequiential\_min\_max.c, скомпилировать, проверить, что написанный вами GetMinMax работает правильно.*



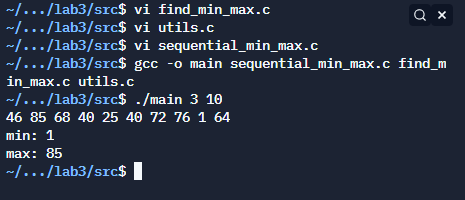




argv[0] is the name of the program, or an empty string if the name is not available.

argv[1] is **seed** − This is an integer value to be used as seed by the pseudo-random number generator algorithm

argv[2] is array\_size – количество элементов массива.



**Задание 2 – 3**

### **Необходимые знания**

**1.Аргументы командной строки.**

Аргументы командной строки — это необязательные строковые аргументы, передаваемые операционной системой в программу при её запуске. Программа может их использовать в качестве входных данных, либо игнорировать. Подобно тому, как параметры одной функции предоставляют данные для параметров другой функции, так и аргументы командной строки предоставляют возможность людям или программам предоставлять входные данные для программы.

Чтобы принять аргументы командной строки, используются два специальных встроенных аргумента: argc и argv.

Int main (int argc, char \*argv[]) argc (англ. «argument count» = «количество аргументов») — это целочисленный параметр, содержащий количество аргументов, переданных в программу. argc всегда будет как минимум 1, так как первым аргументом всегда является имя самой программы.

Каждый аргумент командной строки, который предоставляет пользователь, заставит argc увеличиться на 1.

Argv (англ. «argument values» = «значения аргументов») — это место, где хранятся фактические значения аргументов. Длина этого массива — argc.

**2.Системный вызов fork.**

Системный вызов fork запускает новый процесс, который является точной копией существующего, простым копированием сегментов с исполняемым кодом и данными. Вновь созданный процесс продолжает работу с того же места, что и изначальный, таким образом, оба процесса продолжают исполнять один и тот же код.

**3.Системный вызов pipe.**

Системный вызов pipe предназначен для создания pip'а внутри операционной системы.

Параметр fd является указателем на массив из двух целых переменных. При нормальном завершении вызова в первый элемент массива – fd[0] – будет занесен файловый дескриптор, соответствующий выходному потоку данных pip’а и позволяющий выполнять только операцию чтения, а во второй элемент массива – fd[1] – будет занесен файловый дескриптор, соответствующий входному потоку данных и позволяющий выполнять только операцию записи.

**4.Работа с файлами в Си.**

Для программиста открытый файл представляется как последовательность считываемых или записываемых данных. При открытии файла с ним связывается поток ввода-вывода. Выводимая информация записывается в поток, вводимая информация считывается из потока.

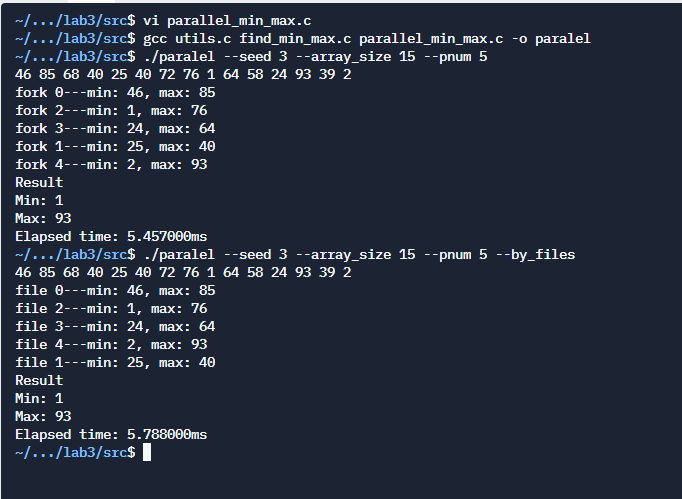
Когда поток открывается для ввода-вывода, он связывается со стандартной структурой типа FILE, которая определена в stdio.h. Структура FILE содержит необходимую информацию о файле.

Открытие файла осуществляется с помощью функции fopen(), которая возвращает указатель на структуру типа FILE, который можно использовать для последующих операций с файлом.

*Завершить программу parallel\_min\_max.c, так, чтобы задача нахождения минимума и максимума в массиве решалась параллельно. Если выставлен аргумент by\_files для синхронизации процессов использовать файлы (задание 2), в противном случае использовать pipe (задание 3).*

*(code in github)*

https://github.com/lehoanggia2k/os\_lab\_2019/tree/hg

**

**Задание 4**

### **Необходимые знания**

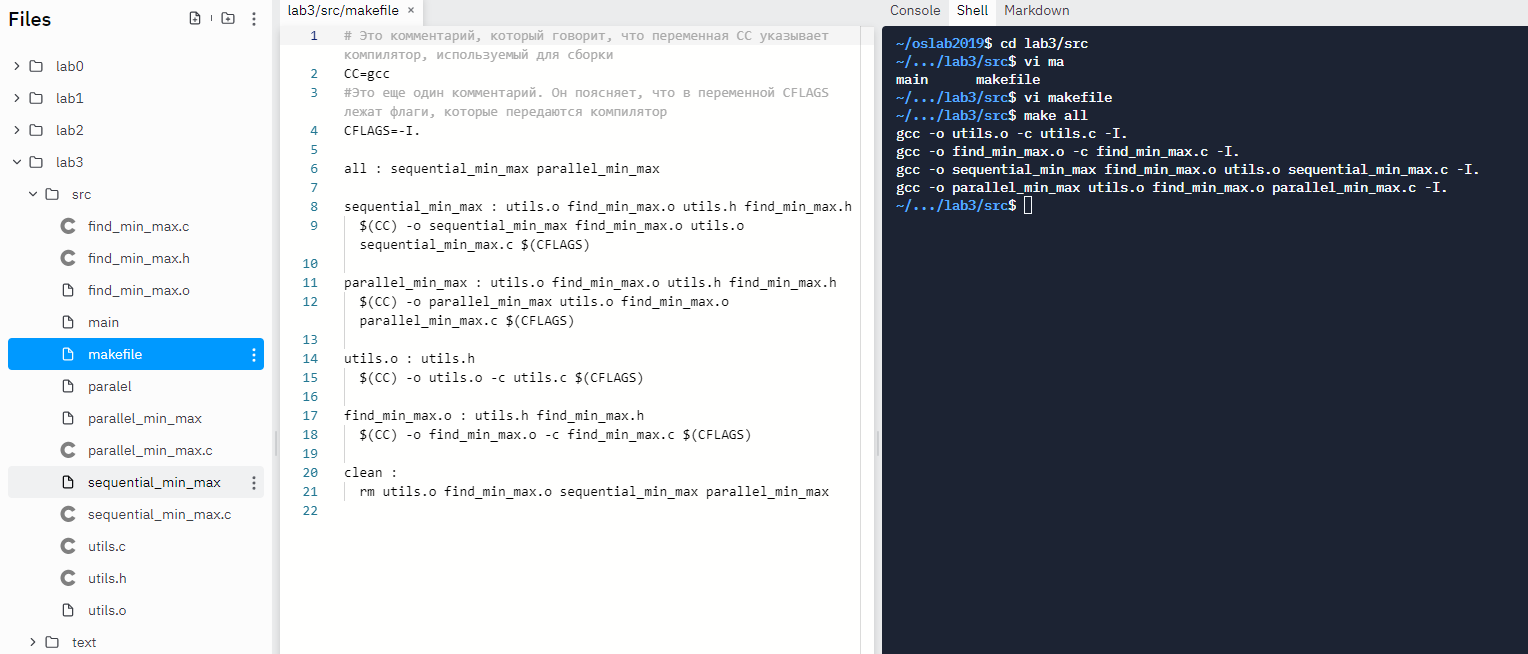
**1.Как работают Makefile'ы.**

[make](http://ru.wikipedia.org/wiki/Make) – утилита предназначенная для автоматизации преобразования файлов из одной формы в другую. Правила преобразования задаются в скрипте с именем Makefile, который должен находиться в корне рабочей директории проекта. Сам скрипт состоит из набора правил, которые в свою очередь описываются:  
1) целями (то, что данное правило делает);

2) реквизитами (то, что необходимо для выполнения правила и получения целей);

3) командами (выполняющими данные преобразования).

*Изучить все targets в makefile, будьте готовы объяснить, за что они отвечают. Используя makefile, собрать получившиеся решения. Добавьте target all, отвечающий за сборку всех программ.*

**

**Задание 5**

### **Необходимые знания**

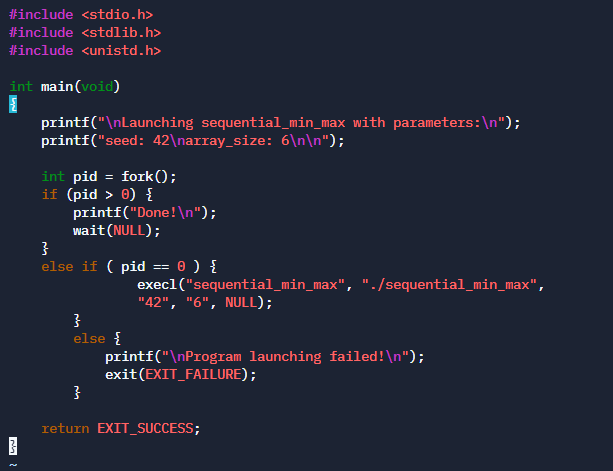
1.Системный вызов exec.

Системный вызов exec повторно инициализирует процесс, подменяя его указанной программой - программа меняется, а процесс остается.

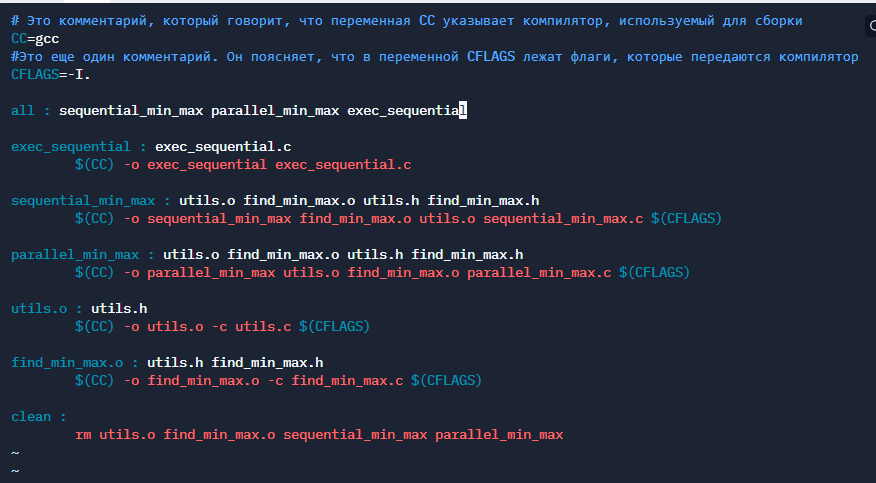
Системный вызов exec – единственный способ запуска программ в Unix, а системный вызов fork – единственный способ запустить новый процесс.

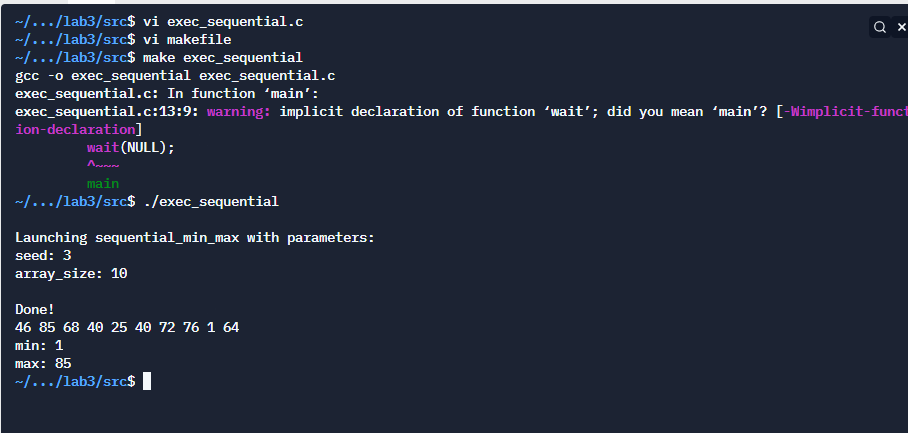
***Написать программу, которая запускает в отдельном процессе ваше приложение sequiential\_min\_max. Добавить его сборку в ваш makefile.***

***File exec\_sequential.c***

******

***File makefie***

******

******