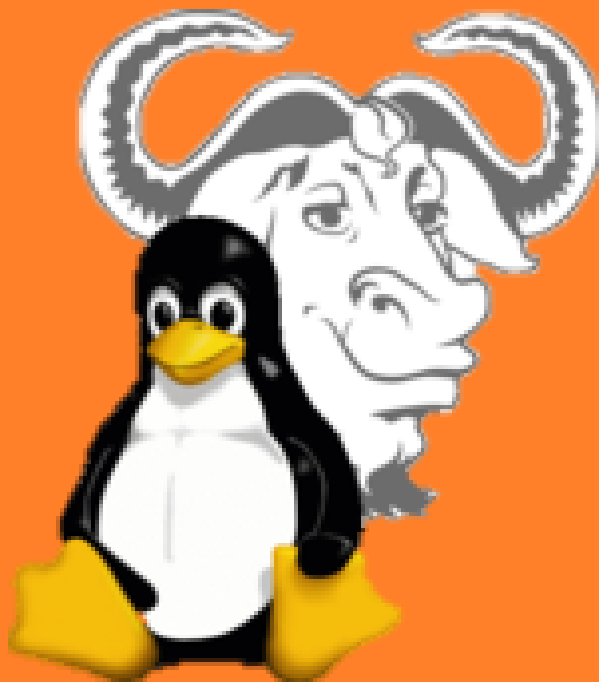


Лабораторная работа #2

# Компиляция и отладка простейшего приложения в Linux

*[GNU Compiler Collection & Project Debugger, gcc,  
g++, adb, \*.c, \*.cc, \*.out]*



# ЛАБОРАТОРНАЯ РАБОТА # 2

## Компиляция и отладка простейшего приложения в Linux

### Цель работы

Изучить встроенный инструментарий для разработки приложений под семейство ОС Linux и фундаментальные основы системного программирования с использованием компиляторов **gcc/g++**, отладка **gdb** и других для проектирования, компиляции, отладки и запуска приложений на языке программирования C/C++.

### Требования

- 1) Разработать модульное консольное приложение на C/C++ согласно варианту задания. Для компиляции, компоновки и выполнения программы использовать стандартный компилятор Linux **gcc/g++**, а для отладки - **gdb**.
- 2) Размерность массива задаётся пользователем на стадии выполнения программы. Для повышения производительности программы необходимо использовать динамическое выделение памяти.
- 3) Во время работы программы должны выводить на экран исходные и конечные данные.
- 4) При выполнении задания запрещается использовать интегрированные средства разработки (*Integrated Development Environment, IDE*). Рекомендуется задействовать любой текстовый редактор (к примеру, **gedit**) и набор компиляторов *GNU Compiler Collection (GCC)*, в частности, компиляторы языков программирования C/C++ **gcc/g++**.
- 5) При разработке программ необходимо придерживаться соглашения по написанию кода на C/C++ (*Code-Convention*).

## Основное задание

Ввести массив вещественных чисел размером  $N$ . Найти его наибольший и наименьший элементы и поменять их местами. Найти сумму и произведение всех элементов массива.

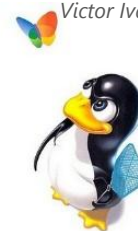
## Индивидуальное задание

- 1) В одномерном массиве, состоящем из  $n$  вещественных элементов, вычислить: сумму отрицательных элементов массива и произведение элементов массива, расположенных между максимальным и минимальным элементами.
- 2) В одномерном массиве, состоящем из  $n$  целых элементов, вычислить: произведение элементов массива с четными номерами и сумму элементов массива, расположенных между первым и последним нулевыми элементами.
- 3) В одномерном массиве, состоящем из  $n$  вещественных элементов, вычислить: максимальный элемент массива и сумму элементов массива, расположенных до последнего положительного элемента.
- 4) В одномерном массиве, состоящем из  $n$  целых элементов, вычислить: номер максимального элемента массива и произведение элементов массива, расположенных между первым и вторым нулевыми элементами.
- 5) В одномерном массиве, состоящем из  $n$  вещественных элементов, вычислить: максимальный по модулю элемент массива и сумму элементов массива, расположенных между первым и вторым положительными элементами.
- 6) В одномерном массиве, состоящем из  $n$  вещественных элементов, вычислить: номер максимального по модулю элемента массива и сумму элементов массива, расположенных после первого положительного элемента.
- 7) В одномерном массиве, состоящем из вещественных элементов, вычислить: количество элементов массива, больших  $S$  и произведение элементов массива, расположенных после максимального по модулю элемента.
- 8) В одномерном массиве, состоящем из  $k$  целых элементов, вычислить: количество положительных элементов массива и сумму элементов массива, расположенных после последнего элемента, равного нулю.
- 9) В одномерном массиве, состоящем из  $n$  вещественных элементов, вычислить: произведение отрицательных элементов массива и сумму положительных элементов массива, расположенных до максимального элемента.

- 10) В одномерном массиве, состоящем из  $n$  вещественных элементов, вычислить: минимальный элемент массива и сумму элементов массива, расположенных между первым и последним положительными элементами.
- 11) В одномерном массиве, состоящем из  $n$  вещественных элементов, вычислить: количество элементов массива, лежащих в диапазоне от  $A$  до  $B$  и сумму элементов массива, расположенных после максимального элемента.
- 12) В одномерном массиве, состоящем из  $n$  вещественных элементов, вычислить: произведение положительных элементов массива и сумму элементов массива, расположенных до минимального элемента.
- 13) В одномерном массиве, состоящем из  $n$  вещественных элементов, вычислить: сумму элементов массива с нечетными номерами и сумму элементов массива, расположенных между первым и последним отрицательными элементами.
- 14) В одномерном массиве, состоящем из  $n$  вещественных элементов, вычислить: номер минимального элемента массива и сумму элементов массива, расположенных между первым и вторым отрицательными элементами.
- 15) В одномерном массиве, состоящем из  $n$  вещественных элементов, вычислить: количество элементов массива, равных 0 и сумму элементов массива, расположенных после минимального элемента.

*Best of LUCK with it, and remember to HAVE FUN while you're learning :)*

Victor Ivanchenko



## Контрольные вопросы

1. Опишите базовый инструментарий системного программиста, проектирующий и разрабатывающий свои приложения на языке C/C++.
2. Что такое исходный (*source*) код, а что такое исполняемый (*executable, binary*) код?
3. Опишите в общем процесс разработки и исполнения программы под Linux? Что происходит за «сценой»?
4. Зачем нужен и какие действия выполняет препроцессор с исходным кодом программы?
5. Что такое компиляция (*compilation*)? С помощью чего и какой команды можно только скомпилировать код? Каким достоинством обладают компиляторы языка программирования C/C++?
6. Что такое компоновка (линковка, сборка)?
7. Как запустить под семейство ОС Linux исполняемый файл на выполнение? От куда (с каких директорий) по умолчанию пытается запустить ОС Linux программы?
8. Что такое **GCC**? Опишите наиболее востребованные опции компилятора **gcc/g++**. Чем отличается **gcc** и **g++**?
9. Опишите основные команды отладчика **gdb** (выполнение программы, трассировка стека, просмотр переменных, вывод листинга программы, установка точек останова, вставка исправлений с помощью отладчика).
10. С помощью каких горячих клавиш в дистрибутиве ОС Ubuntu Linux можно вызвать наиболее востребованные команды и программы, к примеру, Linux-терминал?

## Пример выполнения индивидуального задания

Задание;

В одномерном векторе (массиве), состоящем из N вещественных элементов, вычислить сумму отрицательных и положительных элементов.

- 1) Разработка программы осуществляется под управлением ОС Ubuntu Linux версии 16.04 с графической оболочкой GNOME (см. рис. 1).

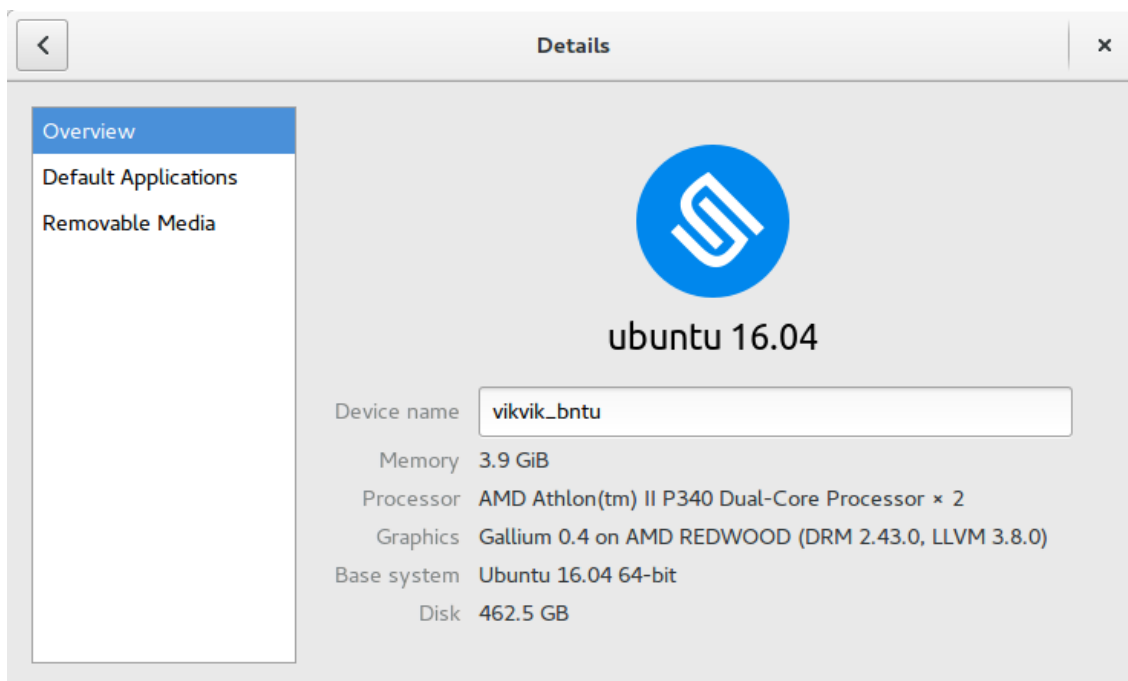
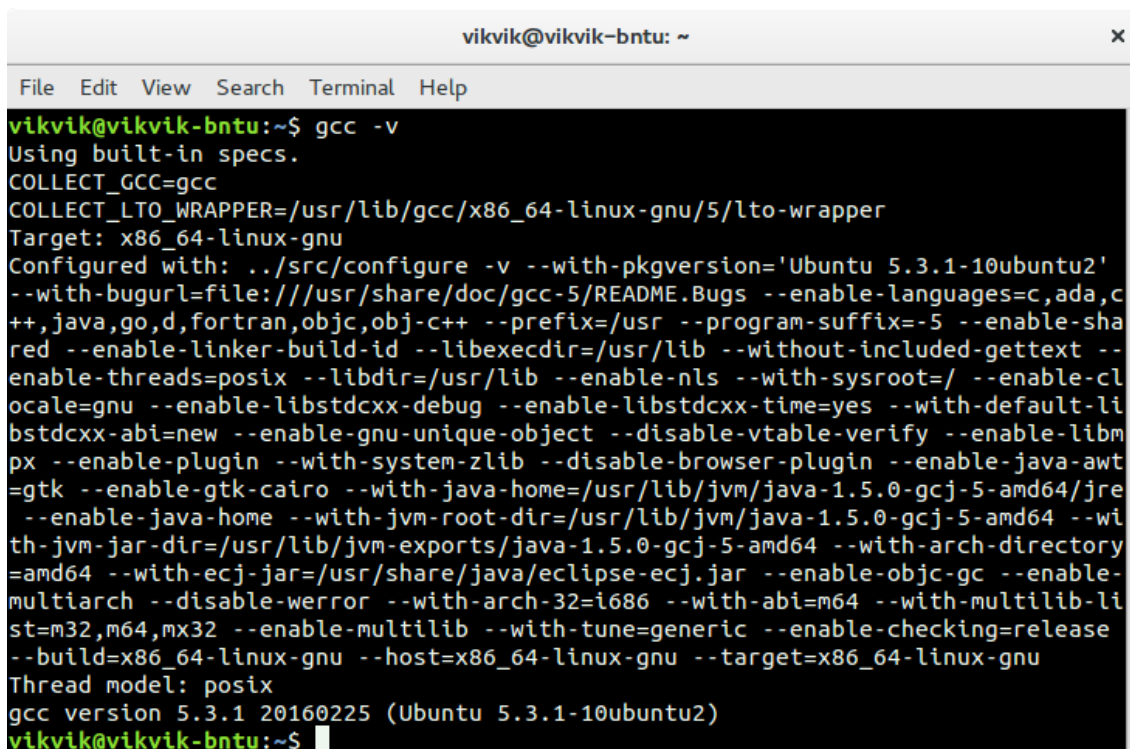


Рисунок 1 – Характеристики компьютера и версия ОС

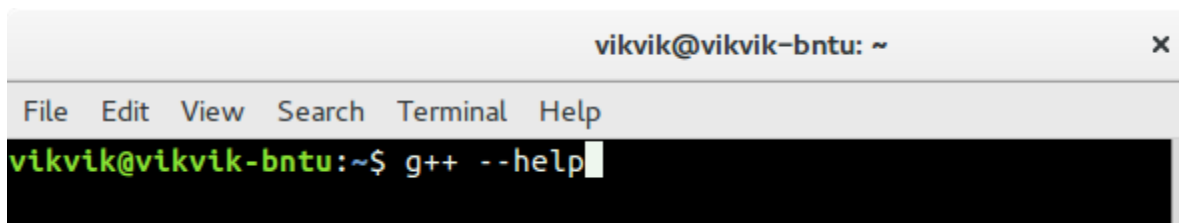
- 2) Для компиляции и сборки программы был использован gcc-компилятор, который устанавливается по умолчанию почти на всех дистрибутивах Linux (версию компилятора 5.3.1 см. на рис. 2).



```
vikvik@vikvik-bntu: ~  
File Edit View Search Terminal Help  
vikvik@vikvik-bntu:~$ gcc -v  
Using built-in specs.  
COLLECT_GCC=gcc  
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux-gnu/5/lto-wrapper  
Target: x86_64-linux-gnu  
Configured with: ../src/configure -v --with-pkgversion='Ubuntu 5.3.1-10ubuntu2'  
--with-bugurl=file:///usr/share/doc/gcc-5/README.Bugs --enable-languages=c,ada,c  
++,java,go,d,fortran,objc,obj-c++ --prefix=/usr --program-suffix=-5 --enable-sha  
red --enable-linker-build-id --libexecdir=/usr/lib --without-included-gettext --  
enable-threads=posix --libdir=/usr/lib --enable-nls --with-sysroot=/ --enable-cl  
ocale=gnu --enable-libstdcxx-debug --enable-libstdcxx-time=yes --with-default-li  
bstdcxx-abi=new --enable-gnu-unique-object --disable-vtable-verify --enable-libm  
px --enable-plugin --with-system-zlib --disable-browser-plugin --enable-java-awt  
=gtk --enable-gtk-cairo --with-java-home=/usr/lib/jvm/java-1.5.0-gcj-5-amd64/jre  
--enable-java-home --with-jvm-root-dir=/usr/lib/jvm/java-1.5.0-gcj-5-amd64 --wi  
th-jvm-jar-dir=/usr/lib/jvm-exports/java-1.5.0-gcj-5-amd64 --with-arch-director  
y=amd64 --with-ecj-jar=/usr/share/java/eclipse-ecj.jar --enable-objc-gc --enable-  
multiarch --disable-werror --with-arch-32=i686 --with-abi=m64 --with-multilib-li  
st=m32,m64,mx32 --enable-multilib --with-tune=generic --enable-checking=release  
--build=x86_64-linux-gnu --host=x86_64-linux-gnu --target=x86_64-linux-gnu  
Thread model: posix  
gcc version 5.3.1 20160225 (Ubuntu 5.3.1-10ubuntu2)  
vikvik@vikvik-bntu:~$
```

Рисунок 2 – Версия компилятора gcc для компиляции и сборки проекта

- 3) Для тех, кто хочет компилировать приложение с помощью C++ компилятора, должен установить утилиту `g++`, т.к. данный компилятор не входит в стандартную поставку многих дистрибутивов Linux. Чтобы проверить, установлен ли данный компилятор в системе, достаточно вызвать терминал (terminal) и ввести команду **`g++ --help`** (см. рис. 3). Если компилятор присутствует в системе, то должен быть выведен список параметров, которые могут использоваться вместе с вызовом компилятора (см. рис. 5). Если данного списка нет, то это значит, что компилятор не установлен. Для его установки, необходимо ввести команду **`sudo apt-get install g++`** (в терминале будет выдана соответствующая подсказка). Во время установки потребуется ввести пароль и дать согласие на скачивание и установку соответствующего пакета (см. рис. 4).



```
vikvik@vikvik-bntu: ~  
File Edit View Search Terminal Help  
vikvik@vikvik-bntu:~$ g++ --help
```

Рисунок 3 – Вызов справки компилятора g++

```

vikvik@vikvik-bntu: ~
File Edit View Search Terminal Help
The program 'g++' is currently not installed. You can install it by typing:
sudo apt install g++
vikvik@vikvik-bntu:~$ sudo apt install g++
[sudo] password for vikvik:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  cpp-5 g++-5 gcc-5 gcc-5-base libasan2 libatomic1 libcc1-0 libcilkrts5 libgcc-5-dev
  libgfortran3 libgomp1 libitm1 liblsan0 libmpx0 libquadmath0 libstdc++6 libtsan0
  libstdc++-5-dev libstdc++6 libubsan0
Suggested packages:
  gcc-5-locales g++-multilib g++-5-multilib gcc-5-multilib libgcc1-dbg libgomp1-dbg
  libasan2-dbg liblsan0-dbg libtsan0-dbg libubsan0-dbg libcilkrts5-dbg libmpx0-dbg
  libquadmath0-dbg libstdc++-5-doc
The following NEW packages will be installed:
  g++ g++-5 libstdc++-5-dev
The following packages will be upgraded:
  cpp-5 gcc-5 gcc-5-base libasan2 libatomic1 libcc1-0 libcilkrts5 libgcc-5-dev
  libgfortran3 libgomp1 libitm1 liblsan0 libmpx0 libquadmath0 libstdc++6
  libtsan0 libubsan0
17 upgraded, 3 newly installed, 0 to remove and 352 not upgraded.
Need to get 33.7 MB/97.9 MB of archives.
After this operation, 164 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://by.archive.ubuntu.com/ubuntu xenial/main amd64 g++-5 amd64 5.3.1-10ubuntu2 [1,427 kB]
Get:2 http://by.archive.ubuntu.com/ubuntu xenial/main amd64 g++-5 amd64 5.3.1-10ubuntu2 [32.3 MB]
57% [2 g++-5 15.0 MB/32.3 MB 46%]

```

Рисунок 4 – Процесс скачки и установки компилятора g++

- 4) После установки соответствующего компилятора g++, повторите ввод команды для вызова справочной информации о компиляторе g++: **g++ --help**. В результате на экране должен появиться список параметров (см. рис. 5).
- 5) Для удобства навигации по папкам и файлам дополнительно поставим файловый менеджер, к примеру, Double Commander. Для этого открываем Ubuntu Software Center и в строку поиска вводим данное название программы или что-то похожее на начало слова "command"..., а затем из списка поиска находим соответствующую программу и нажимаем кнопку «Install» («Установить»). Окно менеджера см. на рис. 7.
- 6) Если Вам не нравится стандартный текстовый редактор **gedit**, то можно скачать любой другой, к примеру, текстовый редактор Sublime Text: ресурс для скачивания: [www.sublimetext.com](http://www.sublimetext.com), см. рис. 6, или через Ubuntu Software Center.



```

vikvik@vikvik-bntu: ~
File Edit View Search Terminal Help
-Wp,<options>      Pass comma-separated <options> on to the preprocessor
-WL,<options>      Pass comma-separated <options> on to the linker
-Xassembler <arg>  Pass <arg> on to the assembler
-Xpreprocessor <arg> Pass <arg> on to the preprocessor
-Xlinker <arg>     Pass <arg> on to the linker
-save-temps        Do not delete intermediate files
-save-temps=<arg>  Do not delete intermediate files
-no-canonical-prefixes Do not canonicalize paths when building relative
                    prefixes to other gcc components
-pipe              Use pipes rather than intermediate files
-time             Time the execution of each subprocess
-specs=<file>       Override built-in specs with the contents of <file>
-std=<standard>     Assume that the input sources are for <standard>
--sysroot=<directory> Use <directory> as the root directory for headers
                    and libraries
-B <directory>     Add <directory> to the compiler's search paths
-v                Display the programs invoked by the compiler
-###              Like -v but options quoted and commands not executed
-E                Preprocess only; do not compile, assemble or link
-S                Compile only; do not assemble or link
-c                Compile and assemble, but do not link
-o <file>          Place the output into <file>
-pie              Create a position independent executable
-shared           Create a shared library
-x <language>      Specify the language of the following input files
                    Permissible languages include: c c++ assembler none
                    'none' means revert to the default behavior of
                    guessing the language based on the file's extension

Options starting with -g, -f, -m, -O, -W, or --param are automatically
passed on to the various sub-processes invoked by g++.  In order to pass
other options on to these processes the -W<letter> options must be used.

For bug reporting instructions, please see:
<file:///usr/share/doc/gcc-5/README.Bugs>.
vikvik@vikvik-bntu:~$

```

Рисунок 5 – Список параметров утилиты gcc/g++

- 7) После установки соответствующего инструментария создадим вложенные папки «*Operation System*» и «*Lab2*» в главной папке текущей учётной записи пользователя, в нашем случае «*Home*» → «vikvik», а затем соответствующий исходный код программы и сохраняем его в файле с именем **lab2.c** (см. рис. 7) - файл, в котором описана главная функция программы **main** и функции бизнес-логики выполнения задания (содержимое файла см. на рис. 8 и 9, а также в ПРИЛОЖЕНИИ А).
- 8) Для компиляции программы запустим терминал Linux и с помощью команды **cd** перейдём в соответствующую папку проекта (см. рис. 7).

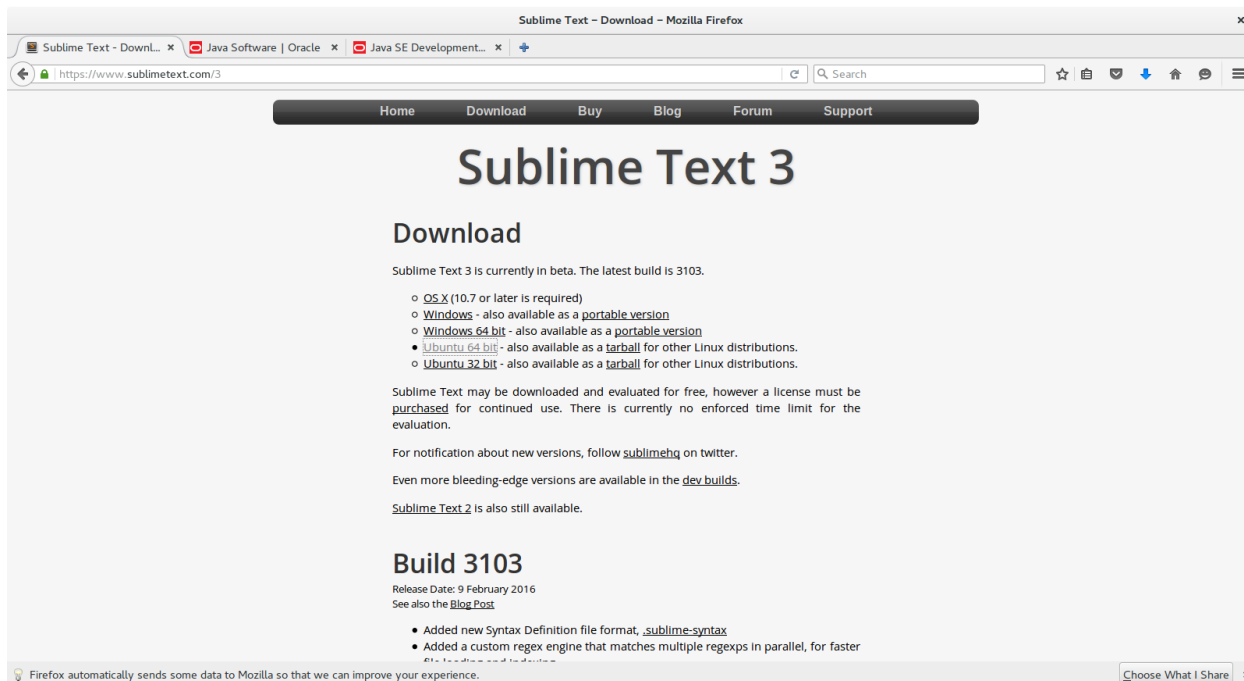


Рисунок 6 – Главная страница для скачивания текстового редактора Sublime Text

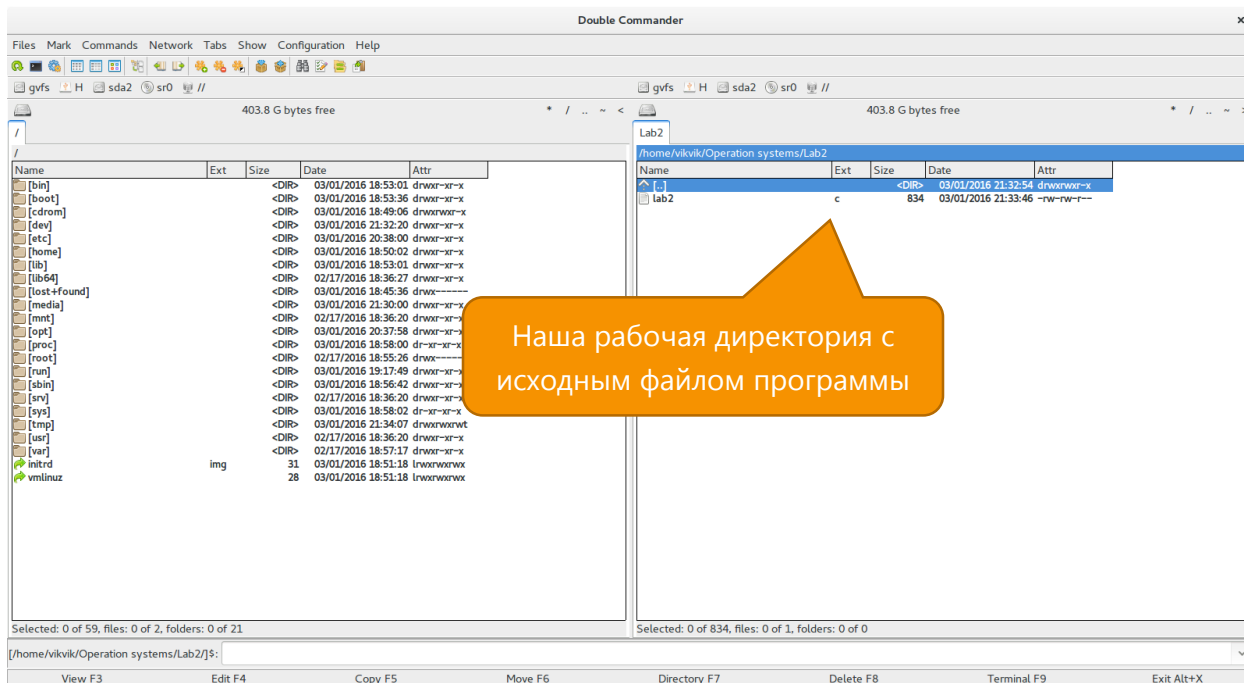


Рисунок 7 – Текущая рабочая директория разработки и выполнения задания

- 9) Для компиляции программы использовал компилятор g++ с вводом соответствующих параметров (см. рис. 2);

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int getSumOfNegativeArrayElements(int*, int);
5 int getSumOfPositiveArrayElements(int*, int);
6
7 int main(){
8     int size;
9     int *array = NULL;
10
11     printf("Input the array size: ");
12     scanf("%d", &size);
13
14     array = malloc(size * sizeof(int));
15
16     printf("Input array elements: ");
17     int i = 0;
18     for (; i < size; i++) {
19         scanf("%d", (array + i));
20     }
21
22     int positiveSum = getSumOfPositiveArrayElements(array, size);
23     int negativeSum = getSumOfNegativeArrayElements(array, size);
24
25     printf("\nResult:");
26     printf("\n\tsum of positive elements = %d", positiveSum);
27     printf("\n\tsum of negative elements = %d\n\n", negativeSum);
28
29     return 0;
30 }
31

```

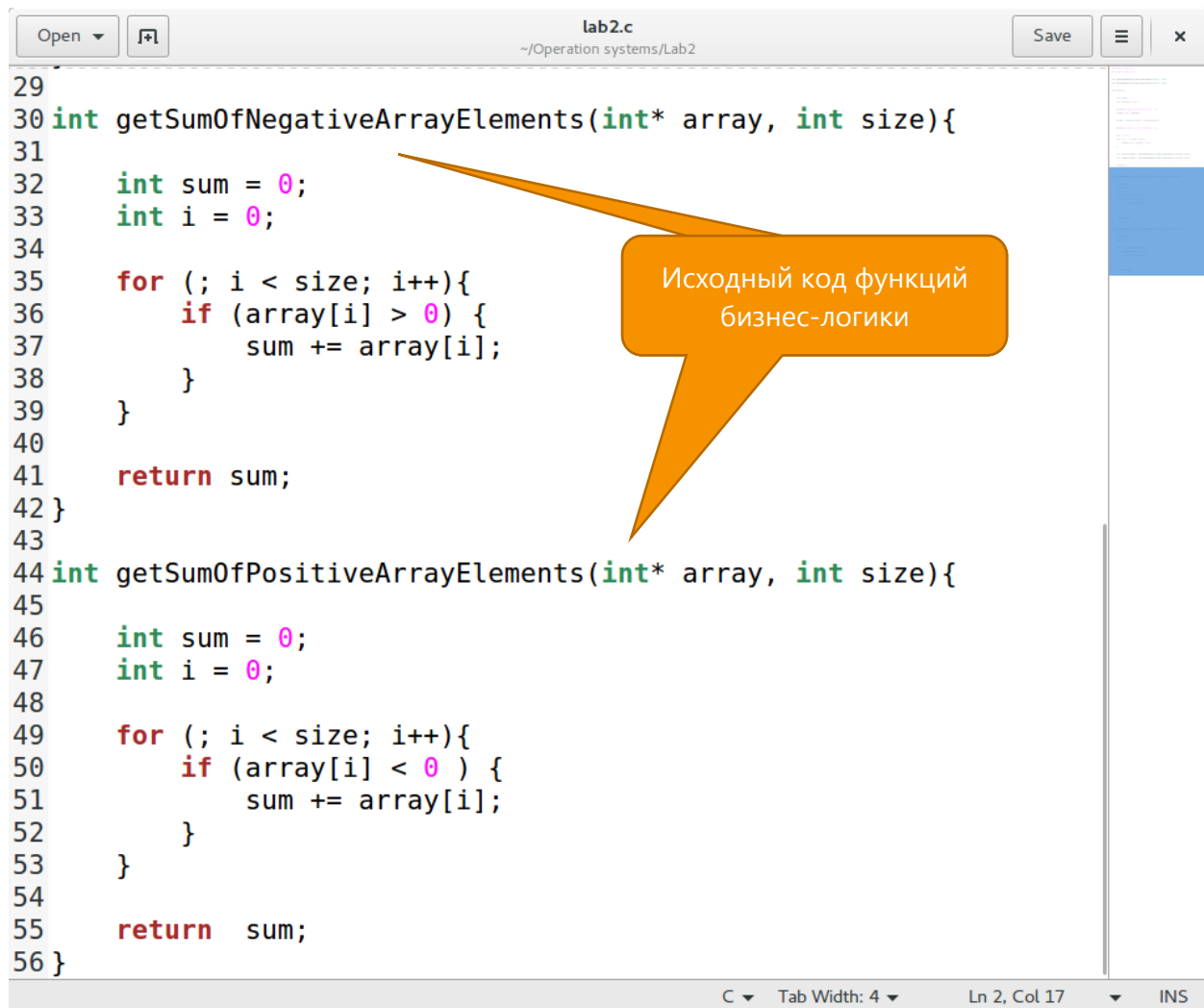
Стартовая функция любой C/C++ программы

Целевой массив целочисленных данных

Динамическое выделение памяти под массив с использованием C-функции **malloc(...)**

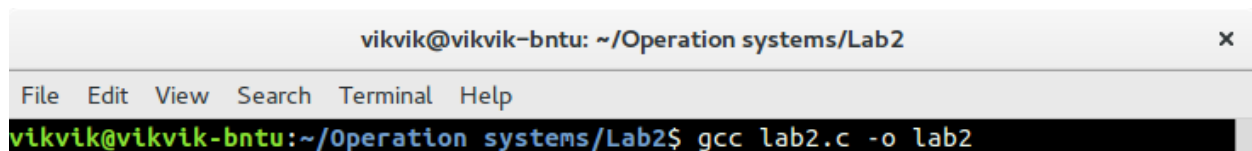
Рисунок 8 – Исходный код главной функции **main**

- 10) После перехода в нужный каталог напишем следующую команду для компиляции и создания исполняемой программы с именем **lab2**: **gcc lab2.c -o lab2** (см. рис. 10).
- 11) Если код написан синтаксически правильно (как в нашем случае 😊), то в результате в текущей рабочей директории должен быть создан исполняемый файл **lab2**. Для его запуска и демонстрации работоспособности программы введём команду **./lab2**, а затем соответствующие данные для тестирования приложения (результат работы программы см. на рис. 11).
- 12) Для отладки программы воспользовались отладчиком **gdb**.



```
29
30 int getSumOfNegativeArrayElements(int* array, int size){
31
32     int sum = 0;
33     int i = 0;
34
35     for (; i < size; i++){
36         if (array[i] > 0) {
37             sum += array[i];
38         }
39     }
40
41     return sum;
42 }
43
44 int getSumOfPositiveArrayElements(int* array, int size){
45
46     int sum = 0;
47     int i = 0;
48
49     for (; i < size; i++){
50         if (array[i] < 0) {
51             sum += array[i];
52         }
53     }
54
55     return sum;
56 }
```

Рисунок 9 – Исходный код функций бизнес-логики



```
vikvik@vikvik-bntu: ~/Operation systems/Lab2
File Edit View Search Terminal Help
vikvik@vikvik-bntu:~/Operation systems/Lab2$ gcc lab2.c -o lab2
```

Рисунок 10 – Компиляция и создание выполняемой программы lab2

```
vikvik@vikvik-bntu: ~/Operation systems/Lab2
File Edit View Search Terminal Help
vikvik@vikvik-bntu:~/Operation systems/Lab2$ gcc lab2.c -o lab2
vikvik@vikvik-bntu:~/Operation systems/Lab2$ ./lab2
Input the array size: 10
Input array elements: 5 4 -3 2 -1 -9 -6 4 -7 0

Result: sum of positive elements = -26  sum of negative elements = 15

vikvik@vikvik-bntu:~/Operation systems/Lab2$ gcc lab2.c -o lab2
vikvik@vikvik-bntu:~/Operation systems/Lab2$ ./lab2
Input the array size: 10
Input array elements: 6 -5 -7 4 3 2 -8 0 1 -9

Result:
    sum of positive elements = -29
    sum of negative elements = 16

vikvik@vikvik-bntu:~/Operation systems/Lab2$
```

Рисунок 11 – Запуск и результат работы программы lab2

## Исходный код программы lab2

```
#include <stdio.h>
#include <stdlib.h>

int getSumOfNegativeArrayElements(int*, int);
int getSumOfPositiveArrayElements(int*, int);

int main(){
    int size;
    int *array = NULL;

    printf("Input the array size: ");
    scanf("%d", &size);

    array = malloc(size * sizeof(int));

    printf("Input array elements: ");
    int i = 0;
    for (; i < size; i++) {
        scanf("%d", (array + i));
    }

    int positiveSum = getSumOfPositiveArrayElements(array, size);
    int negativeSum = getSumOfNegativeArrayElements(array, size);

    printf("\nResult:");
    printf("\n\tsum of positive elements = %d", positiveSum);
    printf("\n\tsum of negative elements = %d\n\n", negativeSum);

    free(array);

    return 0;
}

int getSumOfNegativeArrayElements(int* array, int size){
    int sum = 0;
    int i = 0;

    for (; i < size; i++){
        if (array[i] > 0) {
            sum += array[i];
        }
    }

    return sum;
}

int getSumOfPositiveArrayElements(int* array, int size){
    int sum = 0;
    int i = 0;

    for (; i < size; i++){
        if (array[i] < 0) {
            sum += array[i];
        }
    }

    return sum;
}
```