

# **Отчёта по лабораторной работе 13**

**Средства, применяемые при разработке программного обеспечения в  
ОС типа UNIX/Linux**

Абд эль хай мохамад

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
4.1	Сделать два файла .c и один заголовочный файл .h . . . . .	8
4.1.1	Создал файлы с помощью команды touch . . . . .	8
4.2	Скопировать данные внутрь них . . . . .	9
4.2.1	Используя :vsp из текстового редактора neovim. Вы можете увидеть три файла. . . . .	9
4.3	Скомпилировать .c файлы с gcc . . . . .	9
4.3.1	Шаги компиляции: . . . . .	9
4.3.2	затем выполнил программу ./calcul . . . . .	10
4.4	Создавать Makefile и использовать его . . . . .	10
4.4.1	Скопировал данные в файл Makefile . . . . .	10
4.5	Отлаживать программа с GDB . . . . .	11
4.5.1	Последовательность команд, используемых при отладке программы с помощью gdb . . . . .	11
<b>5</b>	<b>Выводы</b>	<b>15</b>
	<b>Список литературы</b>	<b>16</b>

## Список иллюстраций

4.1	Creat . . . . .	8
4.2	3 Files . . . . .	9
4.3	GCC . . . . .	10
4.4	Makefile . . . . .	10
4.5	GDB . . . . .	12
4.6	Splint . . . . .	13
4.7	Splint . . . . .	14

## Список таблиц

# 1 Цель работы

Приобрести простейшие навыки разработки, анализа, тестирования и отладки приложений в ОС типа UNIX/Linux на примере создания на языке программирования С калькулятора с простейшими функциями

## 2 Задание

- Сделать два файла .c и один заголовочный файл .h
- Скопировать данные внутрь них и скомпилировать файлы .c
- Скомпилировать .c файлы с gcc
- Создавать Makefile и использовать его для компиляции программы
- Отлаживать программа с GDB
- Сканировать программу с помощью splint

## 3 Теоретическое введение

Gdb — отладчик для C (и C++). Это позволяет вам делать такие вещи, как запуск программы до определенной точки, затем останавливаться и распечатывать значения определенных переменных в этой точке, или выполнять программу по одной строке за раз и распечатывать значения каждой переменной после выполнения каждой. линия.

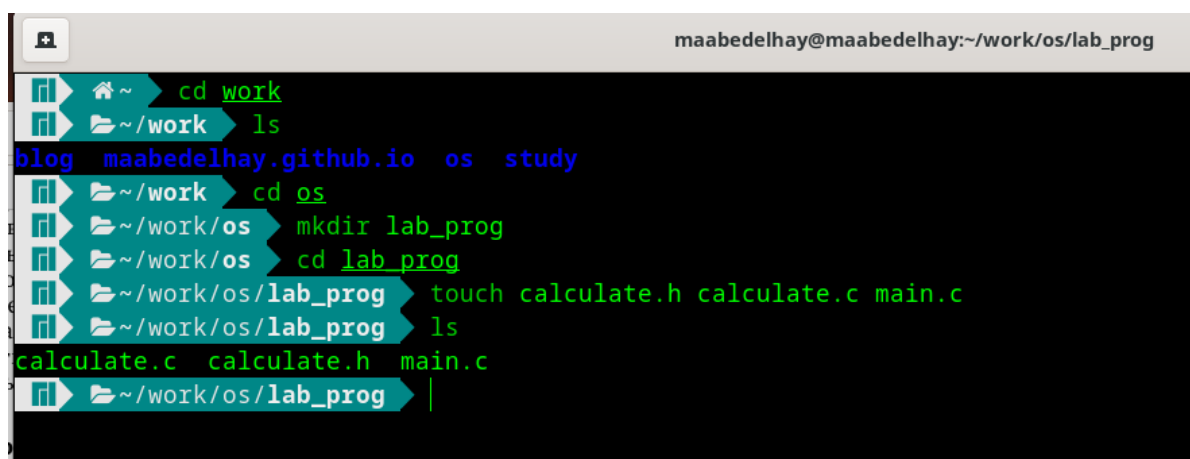
Splint — это инструмент для статической проверки программ на языке C на наличие уязвимостей в системе безопасности и ошибок кода. С минимальными усилиями Splint можно использовать как лучший ворс. Если приложить дополнительные усилия для добавления аннотаций к программам, Splint может выполнить более строгую проверку, чем любой стандартный lint.

## 4 Выполнение лабораторной работы

### 4.1 Сделать два файла .c и один заголовочный файл .h

#### 4.1.1 Создал файлы с помощью команды touch

```
touch calculate.h calculate.c main.c
```



```
maabedelhay@maabedelhay:~/work/os/lab_prog
~
└─> cd work
└─> ls
blog maabedelhay.github.io os study
~/work
└─> cd os
~/work/os
└─> mkdir lab_prog
~/work/os
└─> cd lab_prog
~/work/os/lab_prog
└─> touch calculate.h calculate.c main.c
~/work/os/lab_prog
└─> ls
calculate.c calculate.h main.c
~/work/os/lab_prog
```

Рис. 4.1: Creat



## 4.2 Скопировать данные внутрь них

### 4.2.1 Используя :vsp из текстового редактора neovim. Вы можете увидеть три файла.

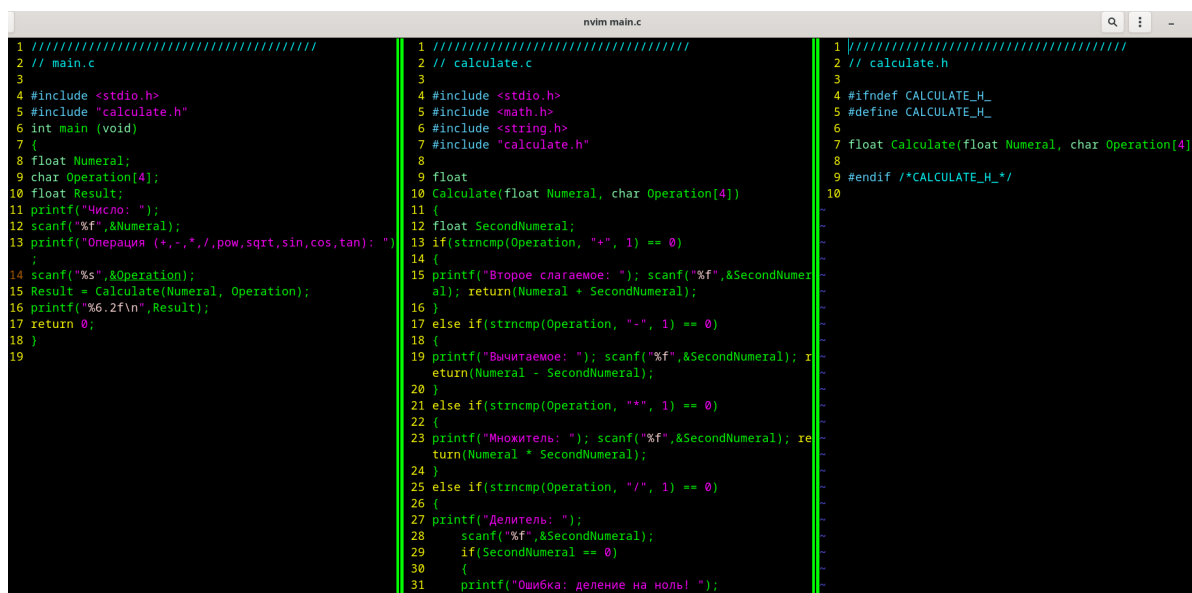


Рис. 4.2: 3 Files

## 4.3 Скомпилировать .c файлы с gcc

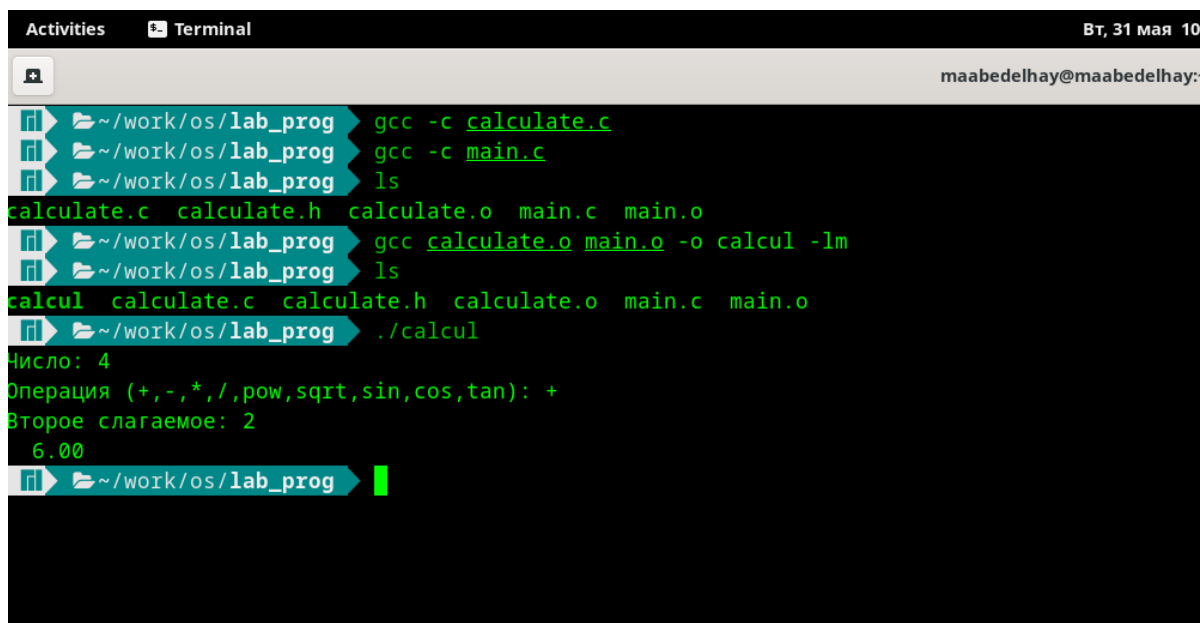
### 4.3.1 Шаги компиляции:

```
gcc -c calculate.c
```

```
gcc -c main.c
```

```
gcc main.o calculate.o -o calcul -lm
```

### 4.3.2 затем выполнил программу ./calcul

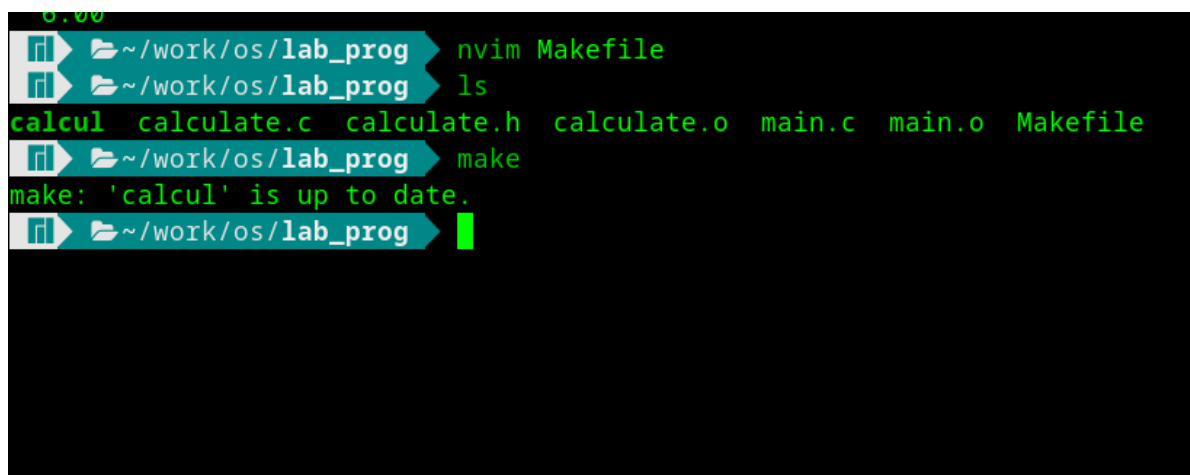


```
Activities Terminal ВТ, 31 мая 10
maabeldelhay@maabeldelhay:~
~/work/os/lab_prog$ gcc -c calculate.c
~/work/os/lab_prog$ gcc -c main.c
~/work/os/lab_prog$ ls
calculate.c calculate.h calculate.o main.c main.o
~/work/os/lab_prog$ gcc calculate.o main.o -o calcul -lm
~/work/os/lab_prog$ ls
calcul calculate.c calculate.h calculate.o main.c main.o
~/work/os/lab_prog$ ./calcul
Число: 4
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): +
Второе слагаемое: 2
6.00
~/work/os/lab_prog$
```

Рис. 4.3: GCC

## 4.4 Создавать Makefile и использовать его

### 4.4.1 Скопировал данные в файл Makefile



```
6.00
~/work/os/lab_prog$ nvim Makefile
~/work/os/lab_prog$ ls
calcul calculate.c calculate.h calculate.o main.c main.o Makefile
~/work/os/lab_prog$ make
make: 'calcul' is up to date.
~/work/os/lab_prog$
```

Рис. 4.4: Makefile

## 4.5 Отлаживать программа с GDB

### 4.5.1 Последовательность команд, используемых при отладке программы с помощью gdb

```
gdb ./calcul
```

```
run
```

```
list
```

```
list 12,5
```

```
list calculate.c:20,29
```

```
list calculate.c:20,27
```

```
break 21
```

```
run 5
```

```

GNU gdb (GDB) 12.1
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./calcul...
(gdb) run
Starting program: /home/maabeldelhay/work/os/lab_prog/calcul
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/usr/lib/libthread_db.so.1".
Число: 1
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): +
Второе слагаемое: 1
    2.00
[Inferior 1 (process 12896) exited normally]
(gdb) list
1      ../sysdeps/x86_64/start.S: No such file or directory.
(gdb) list 12,15
12      in ../sysdeps/x86_64/start.S
(gdb) list calculate.c:20,29
No source file named calculate.c.
(gdb) list calculate.c:20,27
No source file named calculate.c.
(gdb) break 21
Breakpoint 1 at 0x5555555550c0: file ../sysdeps/x86_64/start.S, line 57.
(gdb) info breakpoints
Num      Type             Disp Enb Address                  What
1        breakpoint       keep y   0x00005555555550c0  ../sysdeps/x86_64/start.S:57
(gdb) █

```

Рис. 4.5: GDB

## Сканировать программу с помощью splint

### Сканирование завершилось успешно и выдало 18 предупреждений о коде.

```

~/work/os/lab_prog splint main.c calculate.c +bounds -paramuse -varuse
Splint 3.1.2a --- May 25 2020

calculate.h:7:37: Function parameter Operation declared as manifest array (size
                    constant is meaningless)
    A formal parameter is declared as an array with size. The size of the array
    is ignored in this context, since the array formal parameter is treated as a
    pointer. (Use -fixedformalarray to inhibit warning)
main.c: (in function main)
main.c:12:1: Return value (type int) ignored: scanf("%f", &Num...
    Result returned by function call is not used. If this is intended, can cast
    result to (void) to eliminate message. (Use -retvalint to inhibit warning)
main.c:14:12: Format argument 1 to scanf (%s) expects char * gets char [4] *:
                    &Operation
    Type of parameter is not consistent with corresponding code in format string.
    (Use -formattype to inhibit warning)
    main.c:14:9: Corresponding format code
main.c:14:1: Return value (type int) ignored: scanf("%s", &Ope...
calculate.c:10:31: Function parameter Operation declared as manifest array
                    (size constant is meaningless)
calculate.c: (in function Calculate)
calculate.c:15:46: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:19:35: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:23:33: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:28:2: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:29:5: Dangerous equality comparison involving float types:
                    SecondNumeral == 0
    Two real (float, double, or long double) values are compared directly using
    == or != primitive. This may produce unexpected results since floating point
    representations are inexact. Instead, compare the difference to FLT_EPSILON
    or DBL_EPSILON. (Use -realcompare to inhibit warning)
calculate.c:32:8: Return value type double does not match declared type float:
                    (HUGE_VAL)
    To allow all numeric types to match, use +relaxtypes.
calculate.c:40:2: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:41:8: Return value type double does not match declared type float:
                    (pow(Numeral, SecondNumeral))
calculate.c:44:8: Return value type double does not match declared type float:
                    (pow(Numeral, 3))

```

Рис. 4.6: Splint

```
To allow all numeric types to match, use +relaxtypes.
calculate.c:40:2: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:41:8: Return value type double does not match declared type float:
    (pow(Numeral, SecondNumeral))
calculate.c:44:8: Return value type double does not match declared type float:
    (sqrt(Numeral))
calculate.c:46:8: Return value type double does not match declared type float:
    (sin(Numeral))
calculate.c:48:8: Return value type double does not match declared type float:
    (cos(Numeral))
calculate.c:50:8: Return value type double does not match declared type float:
    (tan(Numeral))
calculate.c:54:8: Return value type double does not match declared type float:
    (HUGE_VAL)

Finished checking --- 18 code warnings
~/work/os/lab_prog
```

Рис. 4.7: Splint

## **5 Выводы**

Работал с отладчиком компилятора и сканером статического кода в среде Linux.

## **Список литературы**