

# **Отчёт по лабораторной работе №13**

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

Надежда Александровна Рогожина

# 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>8</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>10</b>
<b>5</b>	<b>Выводы</b>	<b>21</b>
	<b>Список литературы</b>	<b>22</b>

## Список иллюстраций

4.1	Создание подкаталога . . . . .	10
4.2	Создание файлов . . . . .	10
4.3	calculate.c . . . . .	11
4.4	calculate.h . . . . .	11
4.5	main.c . . . . .	12
4.6	Компиляция программы . . . . .	12
4.7	Создание Makefile . . . . .	13
4.8	make . . . . .	13
4.9	Запуск gdb . . . . .	14
4.10	list . . . . .	14
4.11	list 12,15 . . . . .	15
4.12	list calculate.c:20,29 . . . . .	15
4.13	Точка остановки . . . . .	16
4.14	Точки остановки . . . . .	16
4.15	run + backtrace . . . . .	16
4.16	print Numeral . . . . .	17
4.17	display Numeral . . . . .	17
4.18	Удаление точек остановки . . . . .	17
4.19	run . . . . .	18
4.20	splint calculate.c . . . . .	19
4.21	splint main.c . . . . .	20

# Список таблиц

3.1	Некоторые опции компиляции в gcc . . . . .	8
-----	--	---

# 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

1. В домашнем каталоге создайте подкаталог `~/work/os/lab_prog`.
2. Создайте в нём файлы: `calculate.h`, `calculate.c`, `main.c`.

Это будет примитивнейший калькулятор, способный складывать, вычитать, умножать и делить, возводить число в степень, брать квадратный корень, вычислять `sin`, `cos`, `tan`. При запуске он будет запрашивать первое число, операцию, второе число. После этого программа выведет результат и остановится.

3. Выполните компиляцию программы посредством `gcc`:

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

4. При необходимости исправьте синтаксические ошибки.
5. Создайте `Makefile`.
6. С помощью `gdb` выполните отладку программы `calcul` (перед использованием `gdb` исправьте `Makefile`):
  - Запустите отладчик GDB, загрузив в него программу для отладки:
  - Для запуска программы внутри отладчика введите команду `run`
  - Для постраничного (по 9 строк) просмотра исходного код используйте команду `list`

- Для просмотра строк с 12 по 15 основного файла используйте `list` с параметрами
  - Для просмотра определённых строк не основного файла используйте `list` с параметрами
  - Установите точку остановки в файле `calculate.c` на строке номер 21
  - Выведите информацию об имеющихся в проекте точка остановки
  - Запустите программу внутри отладчика и убедитесь, что программа останавливается в момент прохождения точки остановки
  - Посмотрите, чему равно на этом этапе значение переменной `Numeral`
  - Сравните с результатом вывода на экран после использования команды
  - Уберите точки остановки
7. С помощью утилиты `splint` попробуйте проанализировать коды файлов `calculate.c` и `main.c`.

### 3 Теоретическое введение

Стандартным средством для компиляции программ в ОС типа UNIX является GCC (GNU Compiler Collection). Это набор компиляторов для разного рода языков программирования (C, C++, Java, Фортран и др.). Работа с GCC производится при помощи одноимённой управляющей программы `gcc`, которая интерпретирует аргументы командной строки, определяет и осуществляет запуск нужного компилятора для входного файла.

Файлы с расширением (суффиксом) `.c` воспринимаются `gcc` как программы на языке C, файлы с расширением `.cc` или `.C` — как файлы на языке C++, а файлы с расширением `.o` считаются объектными.

Например, в табл. 3.1 приведено описание некоторых опций `gcc`.

Таблица 3.1: Некоторые опции компиляции в `gcc`

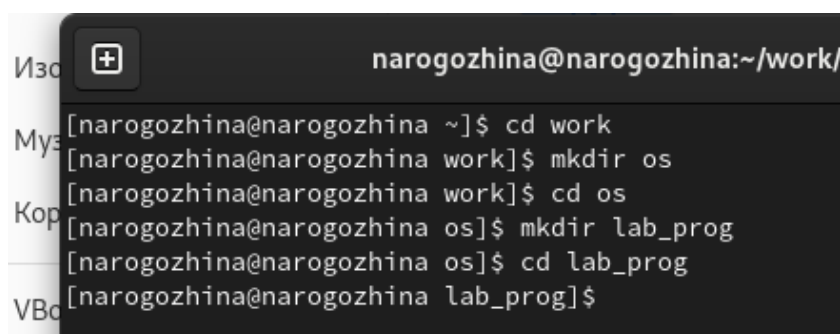
Опция	Описание
<code>-c</code>	Компиляция без компоновки — создаются объектные файлы <code>file.o</code>
<code>-o file-name</code>	Задать имя <code>file-name</code> создаваемому файлу
<code>-g</code>	Поместить в файл (объектный или исполняемый) отладочную информацию для отладчика <code>gdb</code>
<code>-MM</code>	Вывести зависимости от заголовочных файлов C и/или C++ программ в формате, подходящем для утилиты <code>make</code> ; при этом объектные или исполняемые файлы не будут созданы



Опция	Описание
-Wall	Вывод на экран сообщений об ошибках, возникших во время компиляции

## 4 Выполнение лабораторной работы

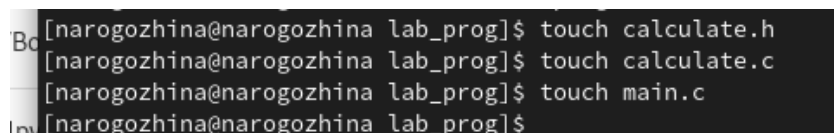
1. В домашнем каталоге создайте подкаталог ~/work/os/lab\_prog (рис. 4.1).



```
narogozhina@narogozhina:~/work/  
[narogozhina@narogozhina ~]$ cd work  
[narogozhina@narogozhina work]$ mkdir os  
[narogozhina@narogozhina work]$ cd os  
[narogozhina@narogozhina os]$ mkdir lab_prog  
[narogozhina@narogozhina os]$ cd lab_prog  
[narogozhina@narogozhina lab_prog]$
```

Рис. 4.1: Создание подкаталога

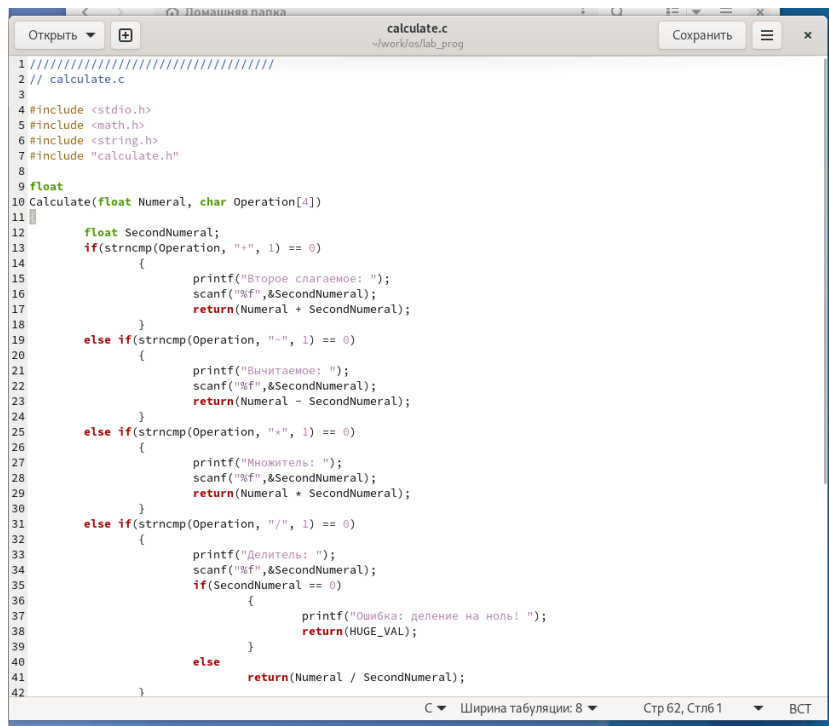
2. Создайте в нём файлы: calculate.h, calculate.c, main.c (рис. 4.2).



```
[narogozhina@narogozhina lab_prog]$ touch calculate.h  
[narogozhina@narogozhina lab_prog]$ touch calculate.c  
[narogozhina@narogozhina lab_prog]$ touch main.c  
[narogozhina@narogozhina lab_prog]$
```

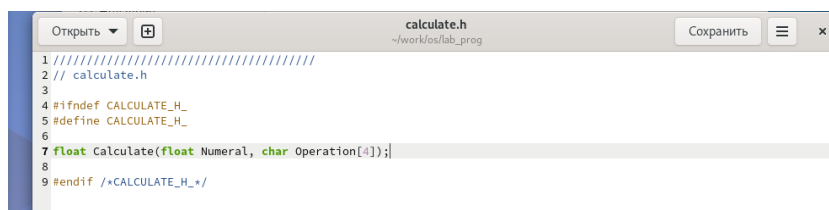
Рис. 4.2: Создание файлов

Это будет примитивнейший калькулятор, способный складывать, вычитать, умножать и делить, возводить число в степень, брать квадратный корень, вычислять  $\sin$ ,  $\cos$ ,  $\tan$ . При запуске он будет запрашивать первое число, операцию, второе число. После этого программа выведет результат и остановится (рис. 4.3, 4.4, 4.5).



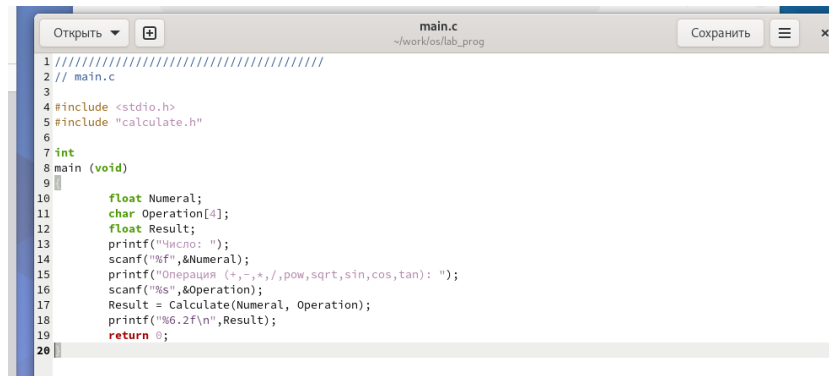
```
1 //////////////////////////////////////////////////
2 // calculate.c
3
4 #include <stdio.h>
5 #include <math.h>
6 #include <string.h>
7 #include "calculate.h"
8
9 float
10 Calculate(float Numeral, char Operation[4])
11 {
12     float SecondNumeral;
13     if(strncmp(Operation, "+", 1) == 0)
14     {
15         printf("Второе слагаемое: ");
16         scanf("%f", &SecondNumeral);
17         return(Numeral + SecondNumeral);
18     }
19     else if(strncmp(Operation, "-", 1) == 0)
20     {
21         printf("Вычитаемое: ");
22         scanf("%f", &SecondNumeral);
23         return(Numeral - SecondNumeral);
24     }
25     else if(strncmp(Operation, "*", 1) == 0)
26     {
27         printf("Множитель: ");
28         scanf("%f", &SecondNumeral);
29         return(Numeral * SecondNumeral);
30     }
31     else if(strncmp(Operation, "/", 1) == 0)
32     {
33         printf("Делитель: ");
34         scanf("%f", &SecondNumeral);
35         if(SecondNumeral == 0)
36         {
37             printf("Ошибка: деление на ноль! ");
38             return(HUGE_VAL);
39         }
40         else
41             return(Numeral / SecondNumeral);
42     }
```

Рис. 4.3: calculate.c



```
1 //////////////////////////////////////////////////
2 // calculate.h
3
4 #ifndef CALCULATE_H_
5 #define CALCULATE_H_
6
7 float Calculate(float Numeral, char Operation[4]);
8
9 #endif /*CALCULATE_H_*/
```

Рис. 4.4: calculate.h



```
1 //////////////////////////////////////////////////
2 // main.c
3
4 #include <stdio.h>
5 #include "calculate.h"
6
7 int
8 main (void)
9 {
10     float Numeral;
11     char Operation[4];
12     float Result;
13     printf("Число: ");
14     scanf("%f",&Numeral);
15     printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");
16     scanf("%s",&Operation);
17     Result = Calculate(Numeral, Operation);
18     printf("%6.2f\n",Result);
19     return 0;
20 }
```

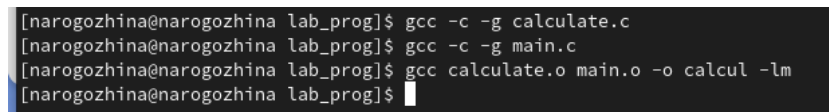
Рис. 4.5: main.c

3. Выполните компиляцию программы посредством gcc (рис. 4.6):

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

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

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



```
[narogozhina@narogozhina lab_prog]$ gcc -c -g calculate.c
[narogozhina@narogozhina lab_prog]$ gcc -c -g main.c
[narogozhina@narogozhina lab_prog]$ gcc calculate.o main.o -o calcul -lm
[narogozhina@narogozhina lab_prog]$
```

Рис. 4.6: Компиляция программы

4. При необходимости исправьте синтаксические ошибки.

Ошибок не было :)

5. Создайте Makefile (рис. 4.7).



Рис. 4.7: Создание Makefile

В данном Makefile целями являются:

- calcul - зависит от: calculate.o, main.o
- calculate.o - зависит от: calculate.c, calculate.h
- main.o - зависит от: main.c, calculate.h

На строках 10, 13, 16 прописаны команды, которые относятся к 9, 12 и 15 строкам соответственно.

6. С помощью gdb выполните отладку программы calcul (перед использованием gdb исправьте Makefile):

Первоначально я запустила Makefile (рис. 4.8), но так как обновления не требовалось, приступила к выполнению дальнейшего задания.

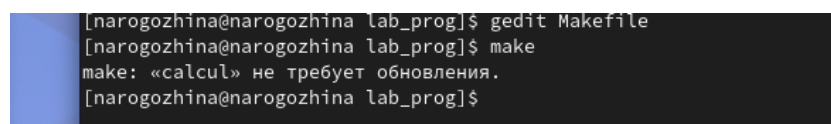


Рис. 4.8: make

- Запустите отладчик GDB, загрузив в него программу для отладки. Для запуска программы внутри отладчика введите команду run (рис. 4.9):

```

make: «calcul» не требует обновления.
[narogozhina@narogozhina lab_prog]$ gdb ./calcul
GNU gdb (GDB) Fedora Linux 13.1-2.fc37
Copyright (C) 2023 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-redhat-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/narogozhina/work/os/lab_prog/calcul

This GDB supports auto-downloading debuginfo from the following URLs:
  <https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 5
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): *
Множитель: 15
75.00
[Inferior 1 (process 3685) exited normally]
(gdb)

```

Рис. 4.9: Запуск gdb

- Для постраничного (по 9 строк) просмотра исходного код используйте команду `list` (рис. 4.10):

```

75.00
[Inferior 1 (process 3685) exited normally]
(gdb) list
1      //////////////////////////////////////////
2      // main.c
3
4      #include <stdio.h>
5      #include "calculate.h"
6
7      int
8      main (void)
9      {
10         float Numeral;
(gdb)

```

Рис. 4.10: list

- Для просмотра строк с 12 по 15 основного файла используйте list с параметрами (рис. 4.11):

```
9      {
10          float Numeral;
(gdb) list 12,15
12          float Result;
13          printf("Число: ");
14          scanf("%f",&Numeral);
15          printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");
(gdb)
```

Рис. 4.11: list 12,15

- Для просмотра определённых строк не основного файла используйте list с параметрами (рис. 4.12):

```
14          scanf("%f",&Numeral);
15          printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");
(gdb) list calculate.c:20,29
20      {
21          printf("Вычитаемое: ");
22          scanf("%f",&SecondNumeral);
23          return(Numeral - SecondNumeral);
24      }
25      else if(strncmp(Operation, "+", 1) == 0)
26      {
27          printf("Множитель: ");
28          scanf("%f",&SecondNumeral);
29          return(Numeral * SecondNumeral);
(gdb)
```

Рис. 4.12: list calculate.c:20,29

- Установите точку остановки в файле calculate.c на строке номер 21 (рис. 4.13):

```

28         scanf("%i",&SecondNumeral);
29         return(Numeral * SecondNumeral);
(gdb) list calculate.c:20,27
20     {
21         printf("Вычитаемое: ");
22         scanf("%i",&SecondNumeral);
23         return(Numeral - SecondNumeral);
24     }
25     else if(strcmp(Operation, "+", 1) == 0)
26     {
27         printf("Множитель: ");
(gdb) break 21
Breakpoint 1 at 0x40120f: file calculate.c, line 21.
(gdb)

```

Рис. 4.13: Точка остановки

- Выведите информацию об имеющихся в проекте точках остановки (рис. 4.14):

```

(gdb) break 21
Breakpoint 1 at 0x40120f: file calculate.c, line 21.
(gdb) info breakpoints
Num  Type      Disp Enb Address             What
1    breakpoint keep y  0x000000000040120f in Calculate at calculate.c:21
(gdb)

```

Рис. 4.14: Точки остановки

- Запустите программу внутри отладчика и убедитесь, что программа останавливается в момент прохождения точки остановки (рис. 4.15):

```

(gdb) run
Starting program: /home/narogozhina/work/os/lab_prog/calcul
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 5
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): -
Breakpoint 1, Calculate (Numeral=5, Operation=0x7fffffffdea4 "-") at calculate.c:21
21     printf("Вычитаемое: ");
(gdb) backtrace
#0  Calculate (Numeral=5, Operation=0x7fffffffdea4 "-") at calculate.c:21
#1  0x00000000004014eb in main () at main.c:17
(gdb)

```

Рис. 4.15: run + backtrace

- Посмотрите, чему равно на этом этапе значение переменной Numeral (рис. 4.16):



```
#1  0x000000000004014eb in main () at m
(gdb) print Numeral
$1 = 5
(gdb)
```

Рис. 4.16: print Numeral

- Сравните с результатом вывода на экран после использования команды (рис. 4.17):

```
#1  0x000000000004014eb in main () at ma
(gdb) print Numeral
$1 = 5
(gdb) display Numeral
1: Numeral = 5
(gdb)
```

Рис. 4.17: display Numeral

Результаты вывода обеих команд одинаковы.

- Уберите точки остановки (рис. 4.18):

```
(gdb) display Numeral
1: Numeral = 5
(gdb) info breakpoints
Num   Type             Disp Enb Address            What
1     breakpoint       keep y   0x0000000000040120f in Calculate at calculate.c:21
      breakpoint already hit 1 time
(gdb) delete 1
(gdb)
```

Рис. 4.18: Удаление точек остановки

После я решила проверить удаление точек останова и запустила программу снова (рис. 4.19):

```
1 breakpoint keep y 0x00000000401201 in calculate
breakpoint already hit 1 time
(gdb) delete 1
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/narogozhina/work/os/lab_prog/calcul
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 5
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): -
Вычитаемое: 1
4.00
[Inferior 1 (process 3796) exited normally]
(gdb) █
```

Рис. 4.19: run

7. С помощью утилиты splint попробуйте проанализировать коды файлов calculate.c и main.c (рис. 4.20, 4.21):

```

[narogozhina@narogozhina lab_prog]$ splint calculate.c
Splint 3.1.2 --- 23 Jul 2022

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)
calculate.c:10:31: Function parameter Operation declared as manifest array
        (size constant is meaningless)
calculate.c: (in function Calculate)
calculate.c:16:4: Return value (type int) ignored: scanf("%f", &Sec...
    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)
calculate.c:22:4: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:28:4: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:34:4: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:35:7: 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:38:12: Return value type double does not match declared type float:
        (HUGE_VAL)
    To allow all numeric types to match, use +relaxtypes.
calculate.c:46:4: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:47:10: Return value type double does not match declared type float:
        (pow(Numeral, SecondNumeral))
calculate.c:50:9: Return value type double does not match declared type float:
        (sqrt(Numeral))
calculate.c:52:9: Return value type double does not match declared type float:
        (sin(Numeral))
calculate.c:54:9: Return value type double does not match declared type float:
        (cos(Numeral))
calculate.c:56:9: Return value type double does not match declared type float:
        (tan(Numeral))
calculate.c:60:10: Return value type double does not match declared type float:
        (HUGE_VAL)

Finished checking --- 15 code warnings
[narogozhina@narogozhina lab_prog]$

```

Рис. 4.20: splint calculate.c

```
Finished checking --- 15 code warnings
[narogozhina@narogozhina lab_prog]$ splint main.c
Splint 3.1.2 --- 23 Jul 2022

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:14:2: 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:16:13: 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:16:10: Corresponding format code
main.c:16:2: Return value (type int) ignored: scanf("%s", &Ope...

Finished checking --- 4 code warnings
[narogozhina@narogozhina lab_prog]$
```

Рис. 4.21: splint main.c

## 5 Выводы

В ходе лабораторной работы мы приобрели простейшие навыки разработки, анализа, тестирования и отладки приложений в ОС типа UNIX/Linux на примере создания на языке программирования С калькулятора с простейшими функциями.

# Список литературы

1. Руководство по выполнению лабораторной работы №13
2. Руководство пользования gdb
3. man splint