

# **Отчёт по лабораторной работе №14**

**дисциплина: Операционные системы**

Латаева Гюзелия Андреевна

# Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	8
4	Выводы	18
5	Контрольные вопросы	19
	Список литературы	23

## Список иллюстраций

3.1	Рисунок 1 . . . . .	9
3.2	Рисунок 2 . . . . .	10
3.3	Рисунок 3 . . . . .	10
3.4	Рисунок 4 . . . . .	11
3.5	Рисунок 5 . . . . .	11
3.6	Рисунок 6 . . . . .	11
3.7	Рисунок 7 . . . . .	12
3.8	Рисунок 8 . . . . .	12
3.9	Рисунок 9 . . . . .	13
3.10	Рисунок 10 . . . . .	13
3.11	Рисунок 11 . . . . .	14
3.12	Рисунок 12 . . . . .	14
3.13	Рисунок 13 . . . . .	14
3.14	Рисунок 14 . . . . .	15
3.15	Рисунок 15 . . . . .	15
3.16	Рисунок 16 . . . . .	15
3.17	Рисунок 17 . . . . .	16
3.18	Рисунок 18 . . . . .	17
3.19	Рисунок 19 . . . . .	17

## 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

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 calculate.c
```

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

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

4. При необходимости исправьте синтаксические ошибки.
5. Создать `Makefile`.
6. С помощью `gdb` выполните отладку программы `calcul` (перед использованием `gdb` исправьте `Makefile`):

– Запустите отладчик GDB, загрузив в него программу для отладки: `gdb ./calcul`

– Для запуска программы внутри отладчика введите команду `run: run`

– Для постраничного (по 9 строк) просмотра исходного код используйте команду `list: list`

– Для просмотра строк с 12 по 15 основного файла используйте list с параметрами: list 12,15

– Для просмотра определённых строк не основного файла используйте list с параметрами: list calculate.c:20,29

– Установите точку останова в файле calculate.c на строке номер 21:

list calculate.c:20,27

break 21

– Выведите информацию об имеющихся в проекте точка останова: info breakpoints

– Запустите программу внутри отладчика и убедитесь, что программа остановится в момент прохождения точки останова: run

5

•

backtrace

– Посмотрите, чему равно на этом этапе значение переменной Numeral, введя: print Numeral (на экран должно быть выведено число 5).

– Сравните с результатом вывода на экран после использования команды: display Numeral

– Уберите точки останова:

info breakpoints

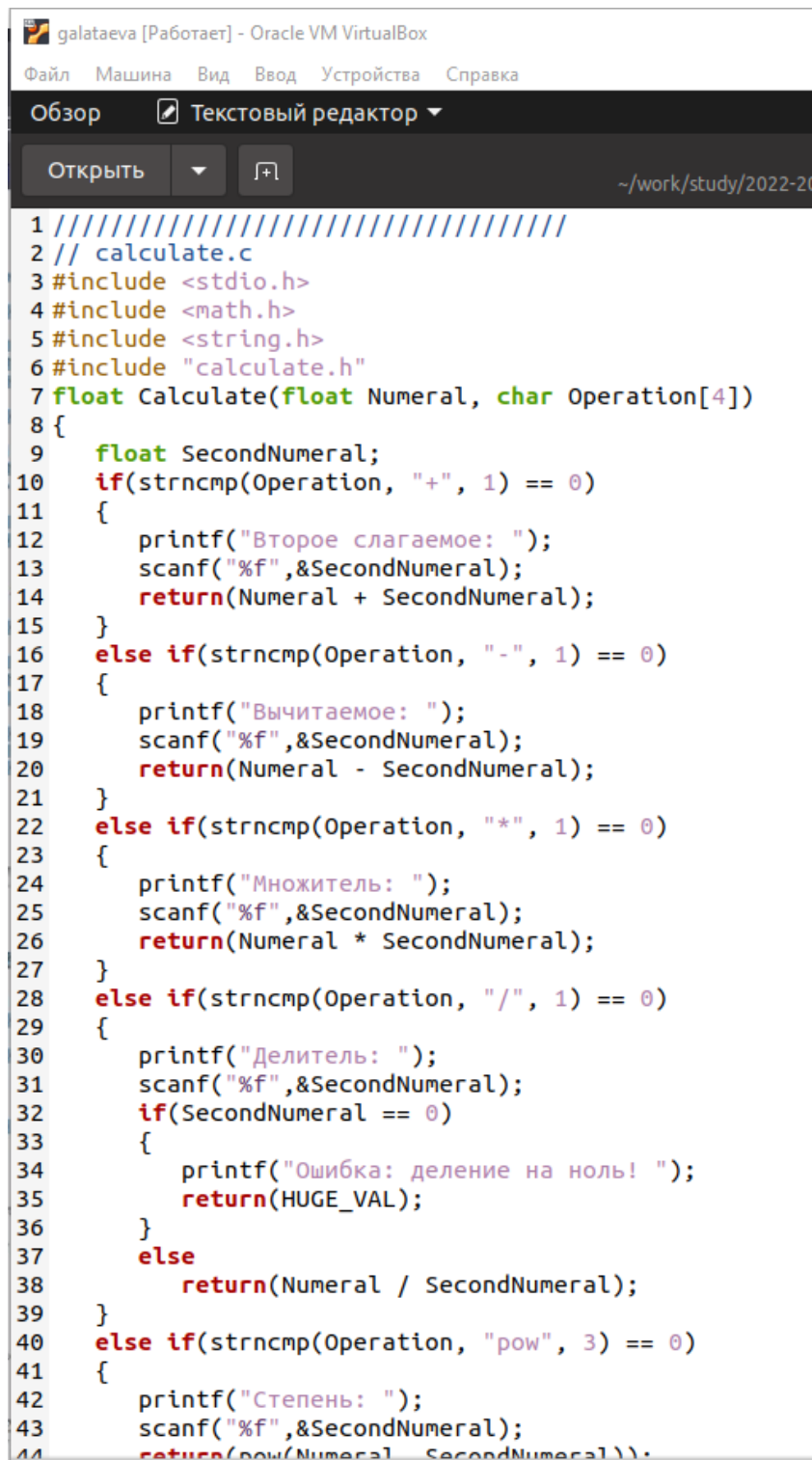
delete 1

7. С помощью утилиты splint попробуйте проанализировать коды файлов calculate.c и main.c

## 3 Выполнение лабораторной работы

1. В домашнем каталоге я создала подкаталог `~/work/os/lab_prog` и создала в нём файлы: `calculate.h`, `calculate.c`, `main.c` и заполнила их кодом из задания:(рис. 3.1), (рис. 3.2), (рис. 3.3)

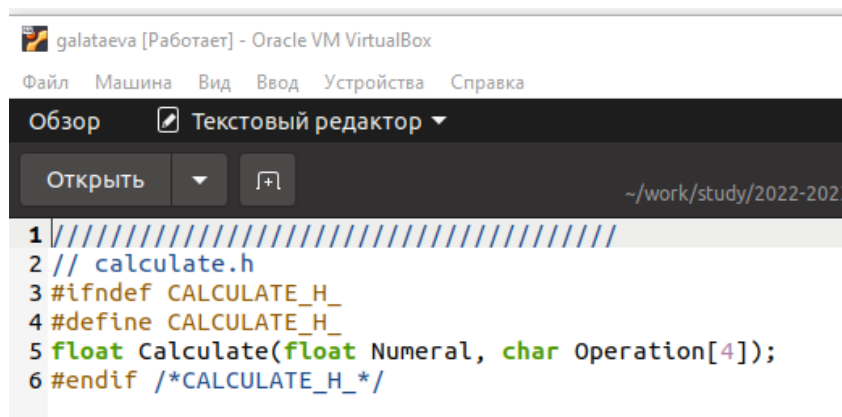




The image shows a screenshot of a text editor window titled "galataeva [Работает] - Oracle VM VirtualBox". The window has a menu bar with "Файл", "Машина", "Вид", "Ввод", "Устройства", and "Справка". Below the menu bar is a toolbar with "Обзор", a "Текстовый редактор" dropdown, and buttons for "Открыть", a dropdown arrow, and a file icon. The address bar shows the path "~/work/study/2022-20". The main text area contains C code for a calculator program, with line numbers 1 through 44 on the left. The code includes headers for stdio, math, and string, and defines a Calculate function that takes a float Numeral and a char Operation array. It uses if-else statements to handle addition, subtraction, multiplication, division, and power operations, with appropriate prompts and error handling for division by zero.

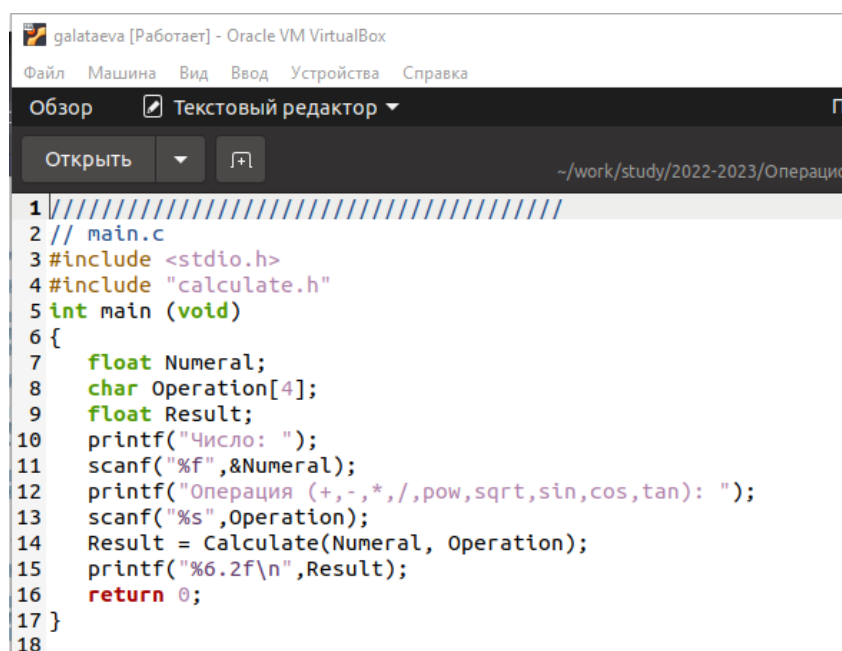
```
1 //////////////////////////////////////////////////
2 // calculate.c
3 #include <stdio.h>
4 #include <math.h>
5 #include <string.h>
6 #include "calculate.h"
7 float Calculate(float Numeral, char Operation[4])
8 {
9     float SecondNumeral;
10    if(strncmp(Operation, "+", 1) == 0)
11    {
12   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

Рис. 3.1: Рисунок 1



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

Рис. 3.2: Рисунок 2



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

Рис. 3.3: Рисунок 3

2. Выполнила компиляцию программы посредством gcc:

gcc -c main.c и gcc calculate.o main.o -o calcul -lm: (рис. 3.4)

```
galataeva@galataeva:~/lab_prog$ gcc -c main.c
galataeva@galataeva:~/lab_prog$ gcc calculate.o main.o -o calcul -lm
galataeva@galataeva:~/lab_prog$
```

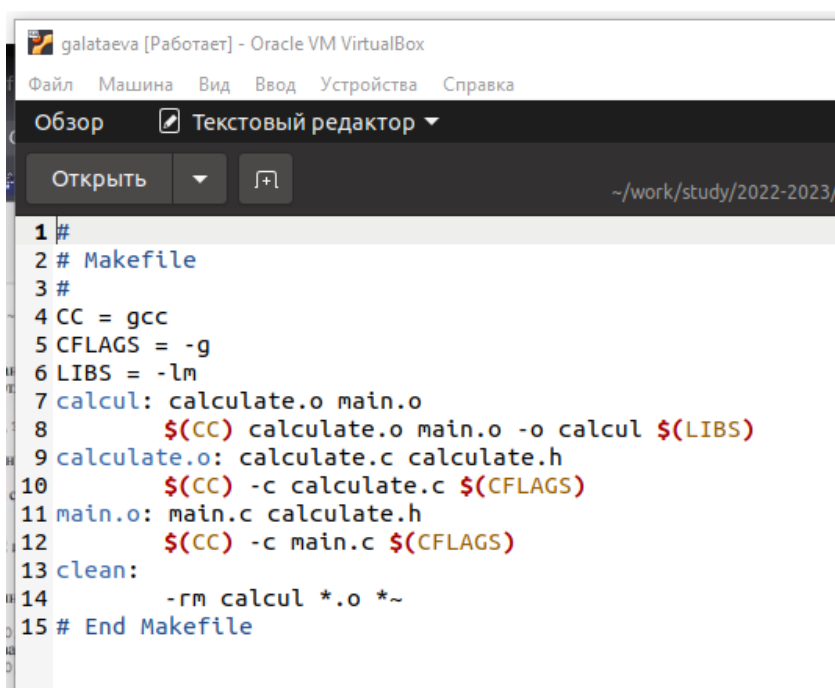
Рис. 3.4: Рисунок 4

gcc -c calculate.c: (рис. 3.5)

```
galataeva@galataeva:~/lab_prog$ gcc -c calculate.c
```

Рис. 3.5: Рисунок 5

3. Создала Makefile со следующим содержанием: (рис. 3.6)



```
1 #
2 # Makefile
3 #
4 CC = gcc
5 CFLAGS = -g
6 LIBS = -lm
7 calcul: calculate.o main.o
8     $(CC) calculate.o main.o -o calcul $(LIBS)
9 calculate.o: calculate.c calculate.h
10    $(CC) -c calculate.c $(CFLAGS)
11 main.o: main.c calculate.h
12    $(CC) -c main.c $(CFLAGS)
13 clean:
14     -rm calcul *.o *~
15 # End Makefile
```

Рис. 3.6: Рисунок 6

CC = gcc - создание переменной компилятора CC

CFLAGS = - создание переменной флагов компиляции CFLAGS

LIBS = -lm - создание переменной библиотек LIBS

calcul: calculate.o main.o. - описывается процесс создания исполняемого файла calcul, зависящего от файлов calculate.o и main.o

gcc calculate.o main.o -o calcul \$(LIBS) - сборка файла calcul

calculate.o: calculate.c calculate.h - описывается процесс создания исполняемого файла calculate.o, зависящего от файлов calculate.c и calculate.h

gcc -c calculate.c \$(CFLAGS): сборка файла calculate.o

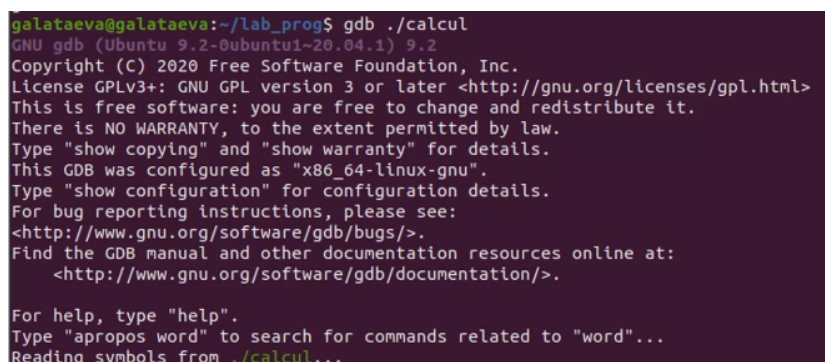
main.o: main.c calculate.h: описывается процесс создания исполняемого файла main.o, зависящего от файлов main.c и calculate.h

gcc -c main.c \$(CFLAGS): сборка файла main.o

clean: -rm calcul.o ~: очистка всех созданных файлов и объектов.

4. С помощью gdb выполнила отладку программы calcul:

– запустила отладчик GDB, загрузив в него программу для отладки: gdb ./calcul (рис. 3.7)

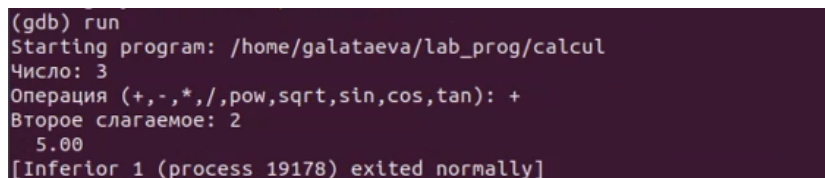


```
galataeva@galataeva:~/lab_prog$ gdb ./calcul
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://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...
```

Рис. 3.7: Рисунок 7

– для запуска программы внутри отладчика ввела команду run: (рис. 3.8)



```
(gdb) run
Starting program: /home/galataeva/lab_prog/calcul
Число: 3
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): +
Второе слагаемое: 2
5.00
[Inferior 1 (process 19178) exited normally]
```

Рис. 3.8: Рисунок 8

– использовала команду list для постраничного просмотра исходного кода:  
(рис. 3.9)

```
[Inferior 1 (process 19178) exited normally]
(gdb) list
1      ///////////////////////////////////////////////////
2      // main.c
3      #include <stdio.h>
4      #include "calculate.h"
5      int main (void)
6      {
7          float Numeral;
8          char Operation[4];
9          float Result;
10         printf("Число: ");
(gdb) list
11         scanf("%f",&Numeral);
12         printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");
13         scanf("%s",Operation);
14         Result = Calculate(Numeral, Operation);
15         printf("%6.2f\n",Result);
16         return 0;
17     }
18
(gdb) █
```

Рис. 3.9: Рисунок 9

– для просмотра строк с 12 по 15 основного файла использовала list 12,15: (рис. 3.10)

```
(gdb) list 12, 15
12         printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");
13         scanf("%s",Operation);
14         Result = Calculate(Numeral, Operation);
15         printf("%6.2f\n",Result);
```

Рис. 3.10: Рисунок 10

– для просмотра определённых строк не основного файла использовала list calculate.c:20,29: (рис. 3.11)

```

(gdb) list calculate.c:20,27
20     return(Numeral - SecondNumeral);
21     }
22     else if(strncmp(Operation, "*", 1) == 0)
23     {
24         printf("Множитель: ");
25         scanf("%f",&SecondNumeral);
26         return(Numeral * SecondNumeral);
27     }

```

Рис. 3.11: Рисунок 11

– установила точку останова в файле calculate.c на строке номер 21: (рис. 3.12)

```

(gdb) break 21
Breakpoint 1 at 0x55555555319: file calculate.c, line 22.
(gdb) list calculate.c:20,27
20     return(Numeral - SecondNumeral);
21     }
22     else if(strncmp(Operation, "*", 1) == 0)
23     {
24         printf("Множитель: ");
25         scanf("%f",&SecondNumeral);
26         return(Numeral * SecondNumeral);
27     }
(gdb) █

```

Рис. 3.12: Рисунок 12

– вывела информацию об имеющихся в проекте точках останова: (рис. 3.13)

```

(gdb) info breakpoints
Num    Type             Disp Enb Address                  What
1      breakpoint      keep y   0x000055555555319 in calculate
                                     at calculate.c:22
(gdb) █

```

Рис. 3.13: Рисунок 13

– запустила программу внутри отладчика. Программа остановилась в момент прохождения точки останова: (рис. 3.14)

```

2.00
[Inferior 1 (process 19231) exited normally]
(gdb) break 21
Note: breakpoint 1 also set at pc 0x55555555319.
Breakpoint 2 at 0x55555555319: file calculate.c, line 22.
(gdb) break 20
Breakpoint 3 at 0x55555555306: file calculate.c, line 20.
(gdb) info breakpoints
Num    Type             Disp Enb Address                  What
1      breakpoint      keep y   0x000055555555319 in Calculate
      at calculate.c:22
2      breakpoint      keep y   0x000055555555319 in Calculate
      at calculate.c:22
3      breakpoint      keep y   0x000055555555306 in Calculate
      at calculate.c:20
(gdb) run
Starting program: /home/galataeva/lab_prog/calcul
Число: 5
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): -
Вычитаемое: 3

Breakpoint 3, Calculate (Numeral=5, Operation=0x7fffffffdfb4 "-")
at calculate.c:20
20      return(Numeral - SecondNumeral);
(gdb)

```

Рис. 3.14: Рисунок 14

– значение переменной Numeral: (рис. 3.15)

```

(gdb) print Numeral
$1 = 5
(gdb)

```

Рис. 3.15: Рисунок 15

– сравнила с результатом вывода на экран после использования команды: (рис. 3.16)

```

(gdb) display Numeral
1: Numeral = 5

```

Рис. 3.16: Рисунок 16

– убрала точки останова: (рис. 3.17)

```

(gdb) info breakpoints
Num    Type             Disp Enb Address              What
1      breakpoint      keep y   0x0000555555555319 in Calculate
                                at calculate.c:22
2      breakpoint      keep y   0x0000555555555319 in Calculate
                                at calculate.c:22
3      breakpoint      keep y   0x0000555555555306 in Calculate
                                at calculate.c:20
breakpoint already hit 1 time
(gdb) delete 1
(gdb) info breakpoints
Num    Type             Disp Enb Address              What
2      breakpoint      keep y   0x0000555555555319 in Calculate
                                at calculate.c:22
3      breakpoint      keep y   0x0000555555555306 in Calculate
                                at calculate.c:20
breakpoint already hit 1 time
(gdb) delete 2
(gdb) info breakpoints
Num    Type             Disp Enb Address              What
3      breakpoint      keep y   0x0000555555555306 in Calculate
                                at calculate.c:20
breakpoint already hit 1 time
(gdb) delete 3
(gdb)

```

Рис. 3.17: Рисунок 17

5. Использовала утилиту splint чтобы проанализировать коды файлов calculate.c и main.c: (рис. 3.18) (рис. 3.19)



```

galataeva@galataeva:~/lab_prog$ splint calculate.c
Splint 3.1.2 --- 20 Feb 2018

calculate.h:5:37: 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:7:37: Function parameter Operation declared as manifest array (size
constant is meaningless)
calculate.c: (in function Calculate)
calculate.c:13:7: 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:19:7: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:25:7: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:31:7: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:32:10: 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:35:16: Return value type double does not match declared type float
(HUGE_VAL)
To allow all numeric types to match, use +relaxtypes.
calculate.c:43:7: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:44:13: Return value type double does not match declared type float
(pow(Numeral, SecondNumeral))
calculate.c:47:13: Return value type double does not match declared type float
(sqrt(Numeral))
calculate.c:49:13: Return value type double does not match declared type float
(sin(Numeral))
calculate.c:51:13: Return value type double does not match declared type float
(cos(Numeral))
calculate.c:53:13: Return value type double does not match declared type float
(tan(Numeral))
calculate.c:57:13: Return value type double does not match declared type float

```

Рис. 3.18: Рисунок 18

```

Finished checking --- 3 code warnings
galataeva@galataeva:~/lab_prog$ splint main.c
Splint 3.1.2 --- 20 Feb 2018

calculate.h:5:37: 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:11:4: 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:13:4: Return value (type int) ignored: scanf("%s", Oper...

Finished checking --- 3 code warnings
galataeva@galataeva:~/lab_prog$

```

Рис. 3.19: Рисунок 19

## 4 Выводы

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

## 5 Контрольные вопросы

1. Как получить информацию о возможностях программ gcc, make, gdb и др.?

Использовать команды `man` и `info` в терминале Linux.

2. Назовите и дайте краткую характеристику основным этапам разработки приложений в UNIX.

1-Написать код на языке программирования

2-Компиляция исходного кода в исполняемый файл

3-Отладка программы

4-Создание Makefile и использование утилиты make

5-Доработка, тестирование

6- Документирование

3. Что такое суффикс в контексте языка программирования? Приведите примеры использования.

Это часть имени файла, которая указывает на его тип или назначение. Например, `program.c` суффикс `“.c”` указывает, что это файл кода на языке программирования C.

4. Каково основное назначение компилятора языка C в UNIX?

Преобразование исходного кода программы, написанной на языке C, в машинный код, который может быть исполнен процессором компьютера.

5. Для чего предназначена утилита make?

Для автоматизации процесса сборки приложения в UNIX.

6. Приведите пример структуры Makefile. Дайте характеристику основным элементам этого файла.

См. пункт 3 выполнения отчета.

7. Назовите основное свойство, присущее всем программам отладки. Что необходимо сделать, чтобы его можно было использовать?

Возможность управлять выполнением программы, остановка ее на определенном месте, просмотр значений переменных и выполнения команд в контексте отладки.

8. Назовите и дайте основную характеристику основным командам отладчика gdb.

backtrace - вывод на экран пути к текущей точке останова

break - установить точку останова

clear - удалить все точки останова в функции continue продолжить выполнение программы

delete - удалить точку останова

display - добавить выражение в список выражений, значения которых отображаются при достижении точки останова программы

finish - выполнить программу до момента выхода из функции

info breakpoints - вывести на экран список используемых точек останова

info watchpoints - вывести на экран список используемых контрольных выражений

list - вывести на экран исходный код (в качестве параметра может быть указано название файла и через двоеточие номера начальной и конечной строк)

`next` - выполнить программу пошагово, но без выполнения вызываемых в программе функций

`print` - вывести значение указываемого в качестве параметра выражения

`run` - запуск программы на выполнение

`set` - установить новое значение переменной

`step` - пошаговое выполнение программы `watch` установить контрольное выражение, при изменении значения которого программа будет остановлена

9. Опишите по шагам схему отладки программы, которую Вы использовали при выполнении лабораторной работы.

1-Компиляция программы с опцией `-c`

2-создание Makefile

3-запуск отладчика `gdb`, указав имя скомпилированного исполняемого файла

4-запуск `run`

5-посмотр исходного кода

6-установка точки останова на нужных строках кода с помощью команды `break`

7-запуск программы

8-исправление кода программы и повтор отладки после нахождения ошибки

10. Прокомментируйте реакцию компилятора на синтаксические ошибки в программе при его первом запуске.

11. Назовите основные средства, повышающие понимание исходного кода программы.

Комментарии, документация, отладчики, редакторы кода, программы анализа кода, руководства.

12. Каковы основные задачи, решаемые программой `splint`?

Это инструмент статического анализа кода на языке C, который помогает обнаруживать ошибки в коде, связанные с безопасностью и потенциальными уязвимостями.

Основные задачи: поиск ошибок в коде, анализ потока данных, проверка безопасности, проверка соответствия стандартам, улучшение качества кода.

# Список литературы

1. ya.ru