

Отчёт по лабораторной работе №13

Дисциплина: Операционные системы

Кристина Алексеевна Антипина

Содержание

Цель работы	1
Выполнение лабораторной работы	1
Вывод.....	10

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.

Выполнение лабораторной работы

1. В домашнем каталоге создаю подкаталог `~/work/os/lab_prog` с помощью команды `mkdir lab_prog`.
2. Создаю в каталоге файлы: `calculate.h`, `calculate.c`, `main.c`, используя команды `cd lab_prog` и `touch calculate.h calculate.c main.c` (рис. -@fig:001).

```
kaantipina@fedora:~/work/os/lab_prog
[kaantipina@fedora os]$ mkdir lab_prog
[kaantipina@fedora os]$ cd lab_prog/
[kaantipina@fedora lab_prog]$ touch calculate.h
[kaantipina@fedora lab_prog]$ touch calculate.c
[kaantipina@fedora lab_prog]$ touch main.c
[kaantipina@fedora lab_prog]$ ls
calculate.c calculate.h main.c
[kaantipina@fedora lab_prog]$
```

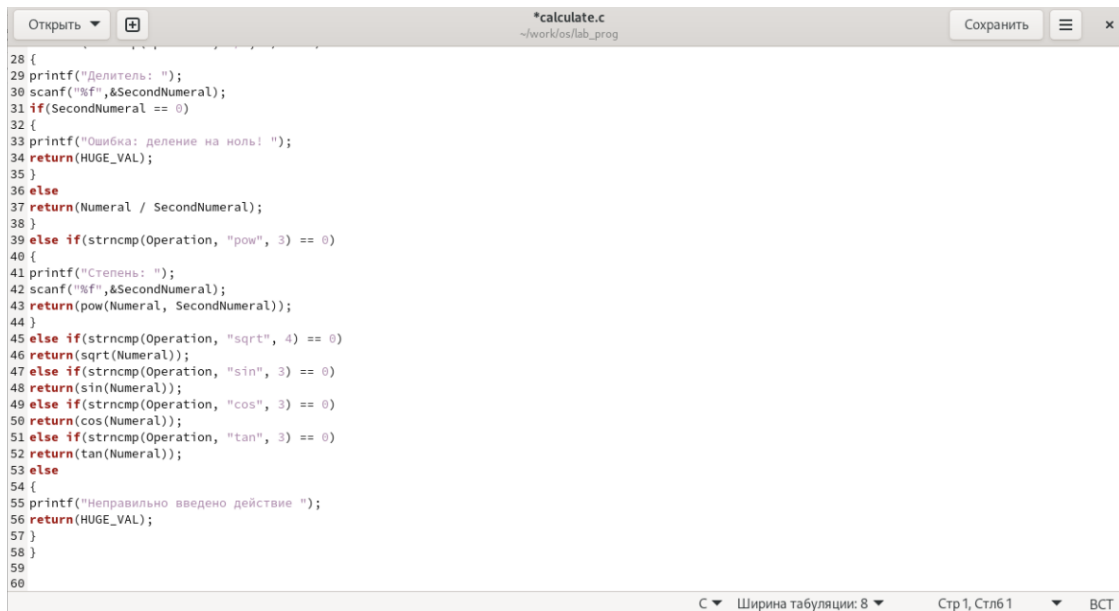
Создаю каталог и файлы в нём

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. Открыв редактор Emacs, приступаю к редактированию созданных файлов. Реализация функций калькулятора в файле `calculate.c` (рис. -@fig:002) (рис. -@fig:003).

```
*calculate.c
~/work/os/lab_prog

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


Реализация функций калькулятора в файле `calculate.c`



```
28 {
29 printf("Делитель: ");
30 scanf("%f",&SecondNumeral);
31 if(SecondNumeral == 0)
32 {
33 printf("Ошибка: деление на ноль! ");
34 return(HUGE_VAL);
35 }
36 else
37 return(Numeral / SecondNumeral);
38 }
39 else if(strncmp(Operation, "pow", 3) == 0)
40 {
41 printf("Степень: ");
42 scanf("%f",&SecondNumeral);
43 return(pow(Numeral, SecondNumeral));
44 }
45 else if(strncmp(Operation, "sqrt", 4) == 0)
46 return(sqrt(Numeral));
47 else if(strncmp(Operation, "sin", 3) == 0)
48 return(sin(Numeral));
49 else if(strncmp(Operation, "cos", 3) == 0)
50 return(cos(Numeral));
51 else if(strncmp(Operation, "tan", 3) == 0)
52 return(tan(Numeral));
53 else
54 {
55 printf("Неправильно введено действие ");
56 return(HUGE_VAL);
57 }
58 }
59
60
```

Реализация функций калькулятора в файле calculate.c

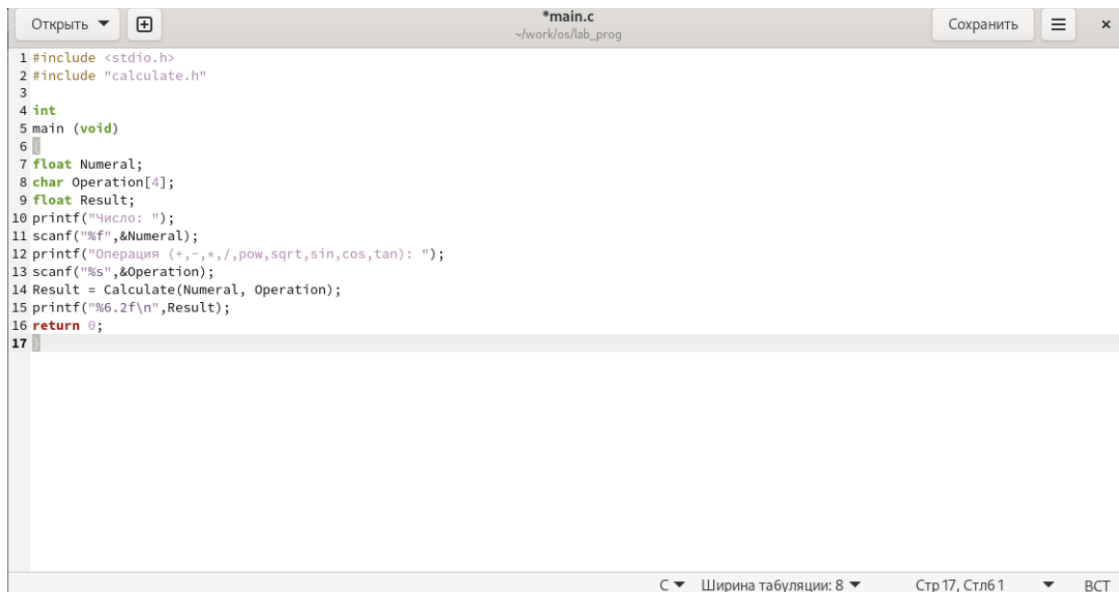
Интерфейсный файл calculate.h, описывающий формат вызова функции калькулятора (рис. -@fig:004).



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

Интерфейсный файл calculate.h

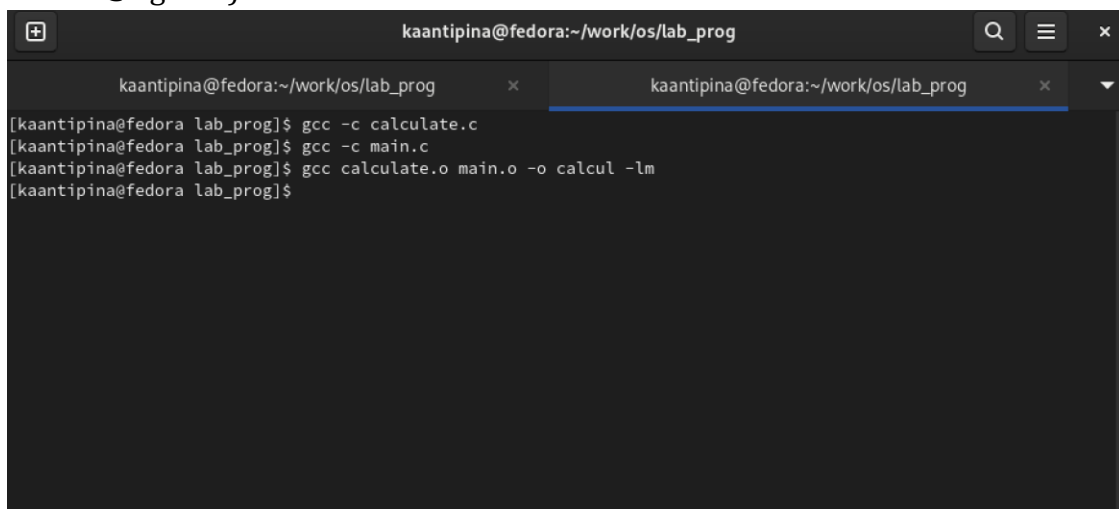
Основной файл main.c, реализующий интерфейс пользователя к калькулятору (рис. -@fig:005).



```
1 #include <stdio.h>
2 #include "calculate.h"
3
4 int
5 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("%6.2f\n",Result);
16    return 0;
17 }
```

Основной файл *main.c*

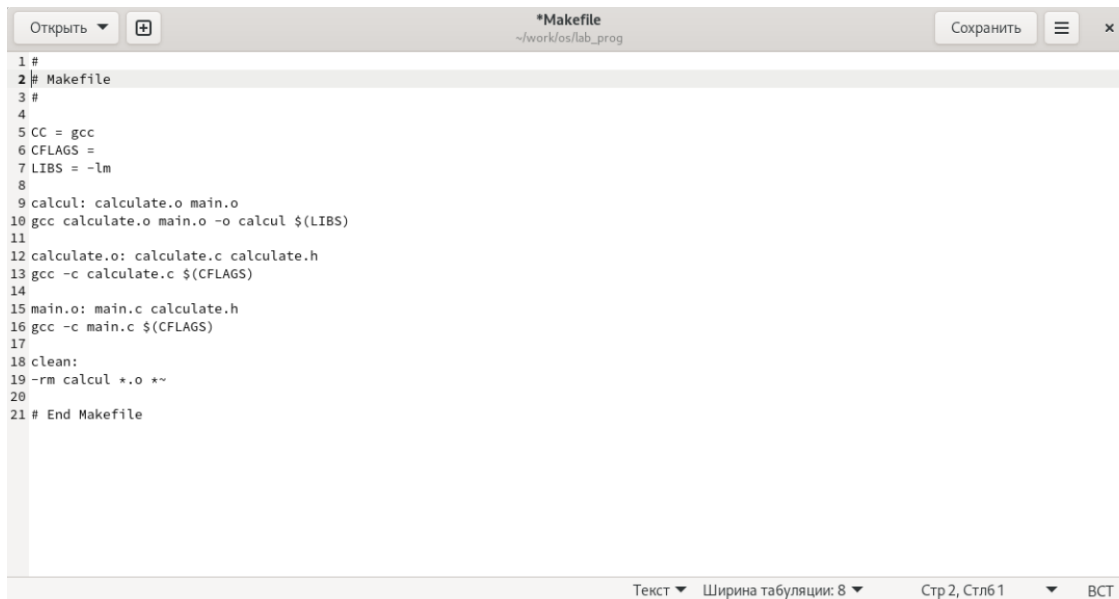
3. Выполню компиляцию программы посредством gcc, используя команды «gcc -с calculate.c», «gcc -с main.c» и «gcc calculate.o main.o -o calcul -lm» (рис. - @fig:006).



```
kaantipina@fedora:~/work/os/lab_prog
[kaantipina@fedora lab_prog]$ gcc -c calculate.c
[kaantipina@fedora lab_prog]$ gcc -c main.c
[kaantipina@fedora lab_prog]$ gcc calculate.o main.o -o calcul -lm
[kaantipina@fedora lab_prog]$
```

Выполняю компиляцию программы посредством gcc

4. В ходе компиляции программы никаких ошибок выявлено не было.
5. Создаю Makefile с необходимым содержанием (рис. - @fig:007). Данный файл необходим для автоматической компиляции файлов calculate.c (цель calculate.o), main.c (цель main.o), а также их объединения в один исполняемый файл calcul (цель calcul). Цель clean нужна для автоматического удаления файлов. Переменная CC отвечает за утилиту для компиляции. Переменная CFLAGS отвечает за опции в данной утилите. Переменная LIBS отвечает за опции для объединения объектных файлов в один исполняемый файл.

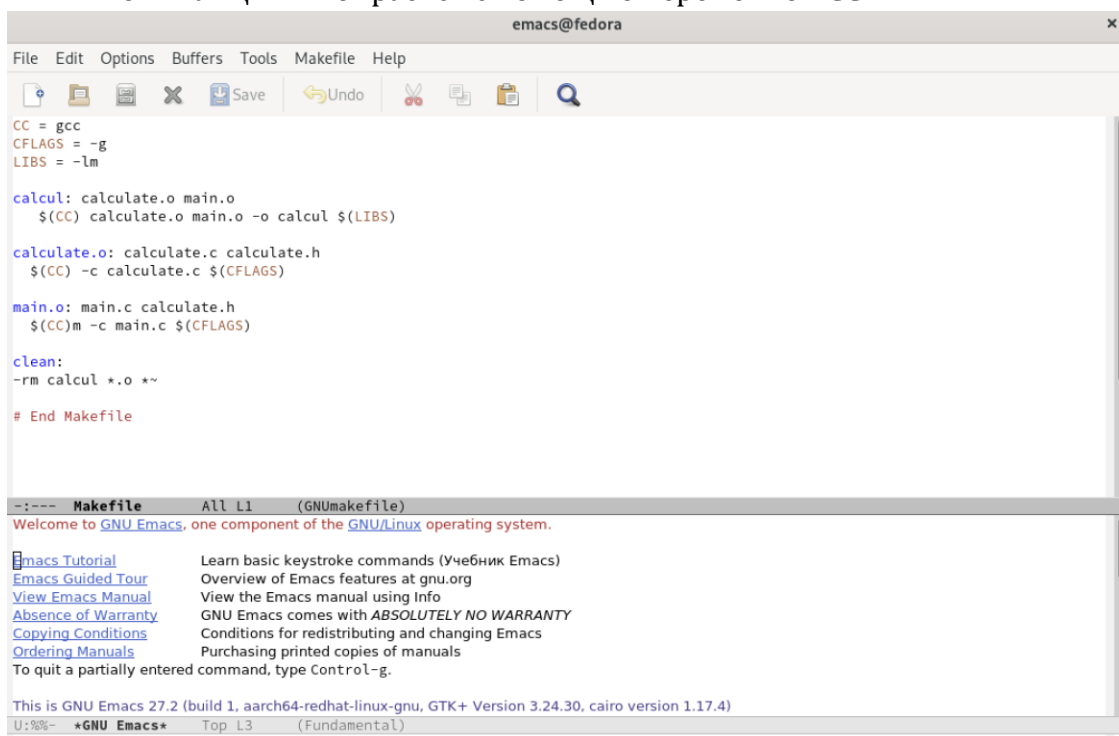


```
1 #
2 # Makefile
3 #
4
5 CC = gcc
6 CFLAGS =
7 LIBS = -lm
8
9 calcul: calculate.o main.o
10 gcc calculate.o main.o -o calcul $(LIBS)
11
12 calculate.o: calculate.c calculate.h
13 gcc -c calculate.c $(CFLAGS)
14
15 main.o: main.c calculate.h
16 gcc -c main.c $(CFLAGS)
17
18 clean:
19 -rm calcul *.o *~
20
21 # End Makefile
```

Текст ▾ Ширина табуляции: 8 ▾ Стр 2, Стлб 1 ▾ ВСТ

Создаю Makefile с необходимым содержанием

6. Далее исправляю Makefile (рис. -@fig:008). В переменную CFLAGS добавляю опцию -g, необходимую для компиляции объектных файлов и их использования в программе отладчика GDB. Сделаю так, что утилита компиляции выбирается с помощью переменной CC.



```
CC = gcc
CFLAGS = -g
LIBS = -lm

calcul: calculate.o main.o
$(CC) calculate.o main.o -o calcul $(LIBS)

calculate.o: calculate.c calculate.h
$(CC) -c calculate.c $(CFLAGS)

main.o: main.c calculate.h
$(CC)m -c main.c $(CFLAGS)

clean:
-rm calcul *.o *~

# End Makefile
```

emacsfedora

File Edit Options Buffers Tools Makefile Help

Save Undo Cut Copy Paste Find

-- Makefile All L1 (GNUmakefile)

Welcome to GNU Emacs, one component of the GNU/Linux operating system.

[Emacs Tutorial](#) Learn basic keystroke commands (Учебник Emacs)

[Emacs Guided Tour](#) Overview of Emacs features at gnu.org

[View Emacs Manual](#) View the Emacs manual using Info

[Absence of Warranty](#) GNU Emacs comes with ABSOLUTELY NO WARRANTY

[Copying Conditions](#) Conditions for redistributing and changing Emacs

[Ordering Manuals](#) Purchasing printed copies of manuals

To quit a partially entered command, type Control-g.

This is GNU Emacs 27.2 (build 1, aarch64-redhat-linux-gnu, GTK+ Version 3.24.30, cairo version 1.17.4)

U:%%- *GNU Emacs* Top L3 (Fundamental)

Далее исправлю Makefile

После этого я удалю исполняемые и объектные файлы из каталога с помощью команды «make clean». Выполню компиляцию файлов, используя команды «make calculate.o», «make main.o», «male calcul» (рис. -@fig:009).

```
[kaantipina@fedora lab_prog]$ make clean
rm calcul *.o *~
[kaantipina@fedora lab_prog]$ make calculate.o
gcc -c calculate.c -g
[kaantipina@fedora lab_prog]$ make main.o
gcc -c main.c -g
[kaantipina@fedora lab_prog]$ male calcul
bash: male: command not found...
[kaantipina@fedora lab_prog]$ make calcul
gcc calculate.o main.o -o calcul -lm
```

Используя команды make

Далее с помощью gdb выполню отладку программы calcul. Запускаю отладчик GDB, загрузив в него программу для отладки, используя команду: «gdb ./calcul» (рис. -@fig:010).

```
[kaantipina@fedora lab_prog]$ gdb ./calcul
GNU gdb (GDB) Fedora 10.2-9.fc35
Copyright (C) 2021 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 "aarch64-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...
(No debugging symbols found in ./calcul)
```

Запускаю отладчик GDB

Для запуска программы внутри отладчика ввожу команду «run» (рис. -@fig:011).

```
(gdb) run
Starting program: /home/kaantipina/work/os/lab_prog/calcul
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 4
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): -
Вычитаемое: 1
      3.00
[Inferior 1 (process 125530) exited normally]
```

Запуск программы внутри отладчика

Для постраничного (по 10 строк) просмотра исходного кода использую команду «list» (рис. -@fig:012).

```
(gdb) list
1      #include <stdio.h>
2      #include "calculate.h"
3
4      int
5      main (void)
6      {
7      float Numeral;
8      char Operation[4];
9      float Result;
10     printf("Число: ");
(gdb) █
```

Использую команду «list

Для просмотра строк с 12 по 15 основного файла использую команду «list 12,15» (рис. -@fig:013).

```
(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);
```

Просмотр строк с 12 по 15

Для просмотра определённых строк не основного файла использую команду «list calculate.c:20,29» (рис. -@fig:014).

```
(gdb) list calculate.c:20,29
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     {
```

Просмотр определённых строк не основного файла

Устанавливаю точку останова в файле calculate.c на строке номер 21, используя команды «list calculate.c:20,27» и «break 21» (рис. -@fig:015).

```
(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) break 21
Breakpoint 1 at 0x400938: file calculate.c, line 22.
(gdb)
```

Устанавливаю точку останова в файле calculate.c

Вывожу информацию об имеющихся в проекте точках останова с помощью команды «info breakpoints» (рис. -@fig:016).

```
(gdb) info breakpoints
Num   Type             Disp Enb Address                  What
1     breakpoint       keep y   0x0000000000400938 in Calculate at calculate.c:22
2     breakpoint       keep y   0x0000000000400938 in Calculate at calculate.c:22
3     breakpoint       keep y   0x0000000000400928 in Calculate at calculate.c:20
breakpoint already hit 1 time
```

Вывожу информацию об имеющихся в проекте точках останова

Запускаю программу внутри отладчика, программа остановилась в момент прохождения точки останова. Использую команды «run», «5», «-» и «backtrace» (рис. -@fig:017).

```
(gdb) run
Starting program: /home/kaantipina/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: backtrace

Breakpoint 3, Calculate (Numeral=5, Operation=0xffffffffef00 "-") at calculate.c:20
20         return(Numeral - SecondNumeral);
(gdb)
```

Запускаю программу внутри отладчика до точки останова

Посмотрю, чему равно на этом этапе значение переменной Numeral, введя команду «print Numeral». Сравню с результатом вывода на экран после использования команды «display Numeral». Значения совпадают. Убираю точку останова с помощью команд «info breakpoints» и «delete 3» (рис. -@fig:018).


```

(gdb) print Numeral
$1 = 5
(gdb) display Numeral
1: Numeral = 5
(gdb) info breakpoints
Num  Type           Disp Enb Address          What
1     breakpoint     keep y   0x0000000000400938 in Calculate at calculate.c:22
2     breakpoint     keep y   0x0000000000400938 in Calculate at calculate.c:22
3     breakpoint     keep y   0x0000000000400928 in Calculate at calculate.c:20
      breakpoint already hit 1 time
(gdb) delete 1
(gdb) splint calculate.c
Undefined command: "splint". Try "help".
(gdb) splint calculate.c
Undefined command: "splint". Try "help".
(gdb) splint
Undefined command: "splint". Try "help".
(gdb) delete 1
No breakpoint number 1.
(gdb) delete 2
(gdb) delete 3
(gdb)

```

Смотрю, чему равно Numeral, display Numeral и убираю точку останова

7. С помощью утилиты splint анализирую коды файлов calculate.c и main.c. Воспользуюсь командами «splint calculate.c» и «splint main.c» (рис. -@fig:019) (рис. -@fig:020).

С помощью утилиты splint выяснилось, что в файлах calculate.c и main.c присутствует функция чтения scanf, возвращающая целое число (тип int), но эти числа не используются и нигде не сохраняются. Утилита вывела предупреждение о том, что в файле calculate.c происходит сравнение вещественного числа с нулем. Также возвращаемые значения (тип double) в функциях pow, sqrt, sin, cos и tan записываются в переменную типа float, что свидетельствует о потере данных.

```
[kaantipina@fedora lab_prog]$ splint calculate.c
Splint 3.1.2 --- 23 Jul 2021

calculate.h:4:37: 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:7:31: 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:35:10: 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:47:11: Return value type double does not match declared type float:
(sqrt(Numeral))
calculate.c:49:11: Return value type double does not match declared type float:
(sin(Numeral))
calculate.c:51:11: Return value type double does not match declared type float:
```

Проанализирую код файла calculate.c

```
[kaantipina@fedora lab_prog]$ splint main.c
Splint 3.1.2 --- 23 Jul 2021

calculate.h:4:37: 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:11:1: 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:13:12: 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:13:9: Corresponding format code
main.c:13:1: Return value (type int) ignored: scanf("%s", &Op...

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

Проанализирую код файла main.c

Вывод

В ходе выполнения лабораторной работы № 13 я приобрела простейшие навыки разработки, анализа, тестирования и отладки приложений в ОС типа UNIX/Linux на примере создания на языке программирования С калькулятора с простейшими функциями.