

# **Лабораторная работа No 13.**

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

Паласиос Фелипе

# Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	8
4	Выводы	17

## Список иллюстраций

3.1	lab_prog . . . . .	8
3.2	Файлы . . . . .	9
3.3	calculate.c . . . . .	9
3.4	calculate.h . . . . .	10
3.5	main . . . . .	10
3.6	Компиляция . . . . .	11
3.7	Makefile . . . . .	11
3.8	Calcul . . . . .	12
3.9	Run . . . . .	12
3.10	list . . . . .	13
3.11	list строк 12-15 . . . . .	13
3.12	list 20,29 . . . . .	14
3.13	list 20,27, break 21 . . . . .	14
3.14	info breakpoints . . . . .	14
3.15	backtrace . . . . .	14
3.16	print Numeral . . . . .	15
3.17	display Numera . . . . .	15
3.18	info breakpoints delete 1 . . . . .	15
3.19	splint calculate.c . . . . .	15
3.20	splint main.c . . . . .	16

## **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. Реализация функций калькулятора в файле `calculate.h`:

Интерфейсный файл `calculate.h`, описывающий формат вызова функции-калькулятора:

Основной файл `main.c`, реализующий интерфейс пользователя к калькулятору:

3. Выполните компиляцию программы посредством `gcc`:
4. При необходимости исправьте синтаксические ошибки.
5. Создайте `Makefile` со следующим содержанием:
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 кода 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 па- раметрами:

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

– Выведите информацию об имеющихся в проекте точка останова:

– Запустите программу внутри отладчика и убедитесь, что программа остано- вится в момент прохождения точки останова:

– Отладчик выдаст следующую информацию: а команда backtrace покажет весь стек вызываемых функций от начала програм- мы до текущего места.

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

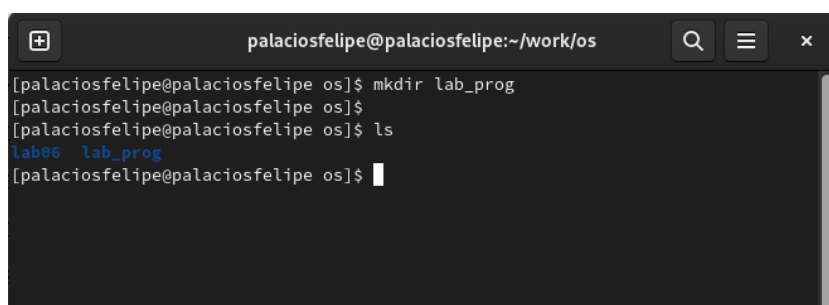
– Сравните с результатом вывода на экран после использования команды:

– Уберите точки останова

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

### 3 Выполнение лабораторной работы

1. В домашнем каталоге создайте подкаталог ~/work/os/lab\_prog. (рис. 3.1)

A terminal window with a dark background. The title bar shows the user 'palaciosfelipe' at host 'palaciosfelipe' in the directory '~/work/os'. The terminal contains the following commands and output:

```
[palaciosfelipe@palaciosfelipe os]$ mkdir lab_prog
[palaciosfelipe@palaciosfelipe os]$
[palaciosfelipe@palaciosfelipe os]$ ls
lab06  lab_prog
[palaciosfelipe@palaciosfelipe os]$
```

Рис. 3.1: lab\_prog

2. Создайте в нём файлы: calculate.h, calculate.c, main.c. (рис. 3.2)



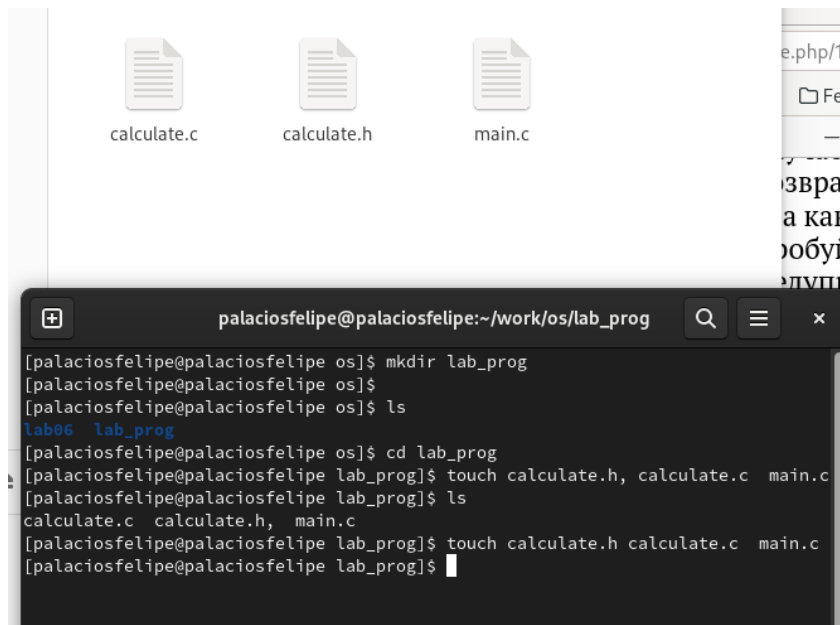


Рис. 3.2: Файлы

Реализация функций калькулятора в файле `calculate.c`(рис. 3.3)

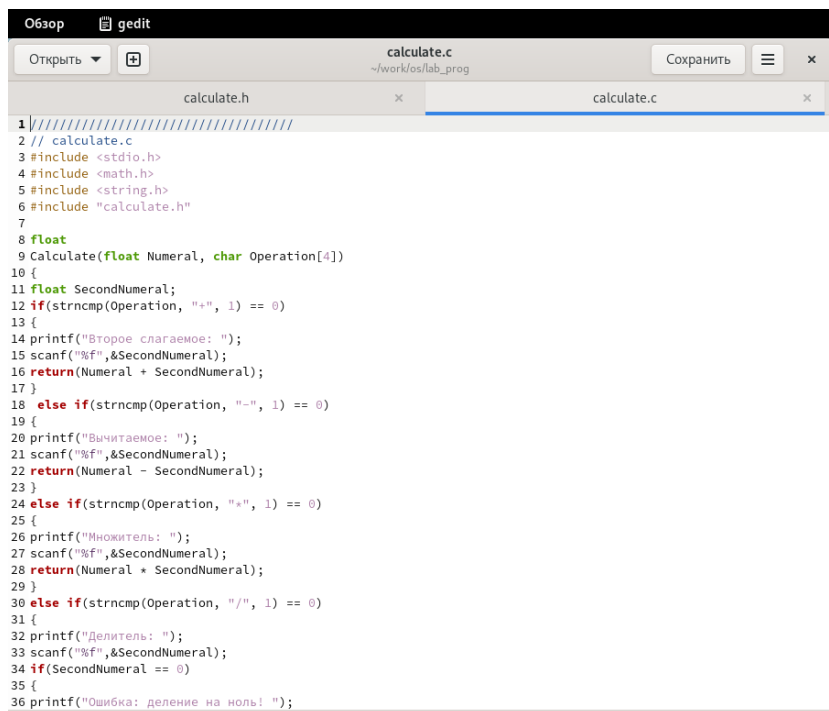
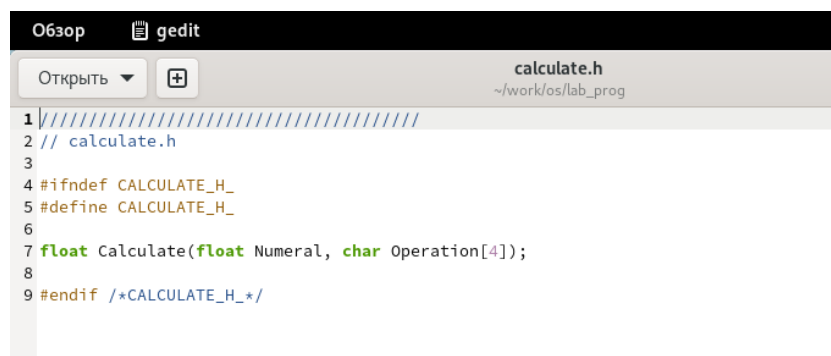


Рис. 3.3: `calculate.c`

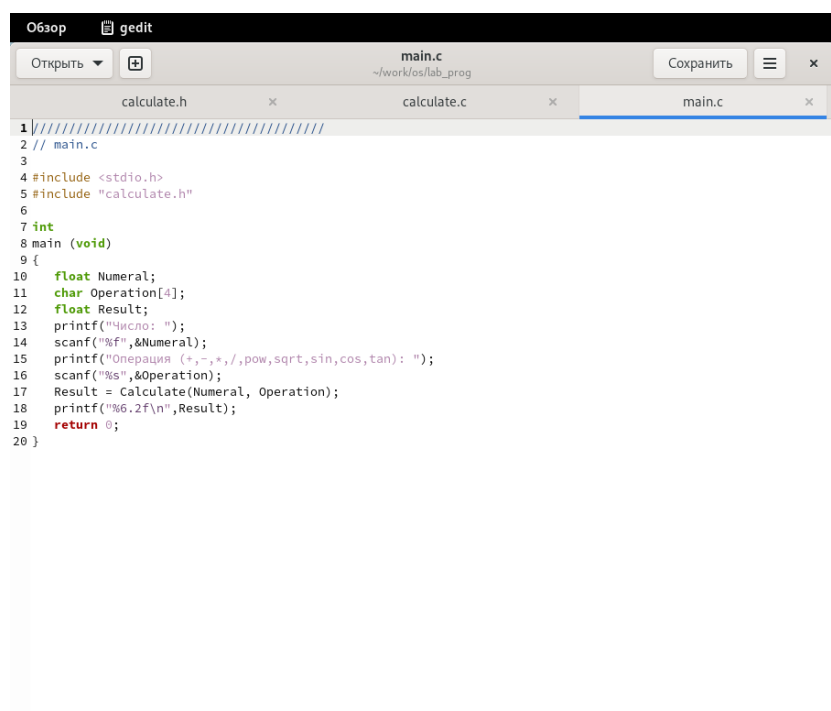
Интерфейсный файл calculate.h, описывающий формат вызова функции- калькулятора:рис. 3.4)



```
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. 3.4: calculate.h

Основной файл main.c, реализующий интерфейс пользователя к калькулятору:рис. 3.5)



```
Обзор gedit
main.c
~work/os/lab_prog
Сохранить

calculate.h x calculate.c x main.c x

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 }
```

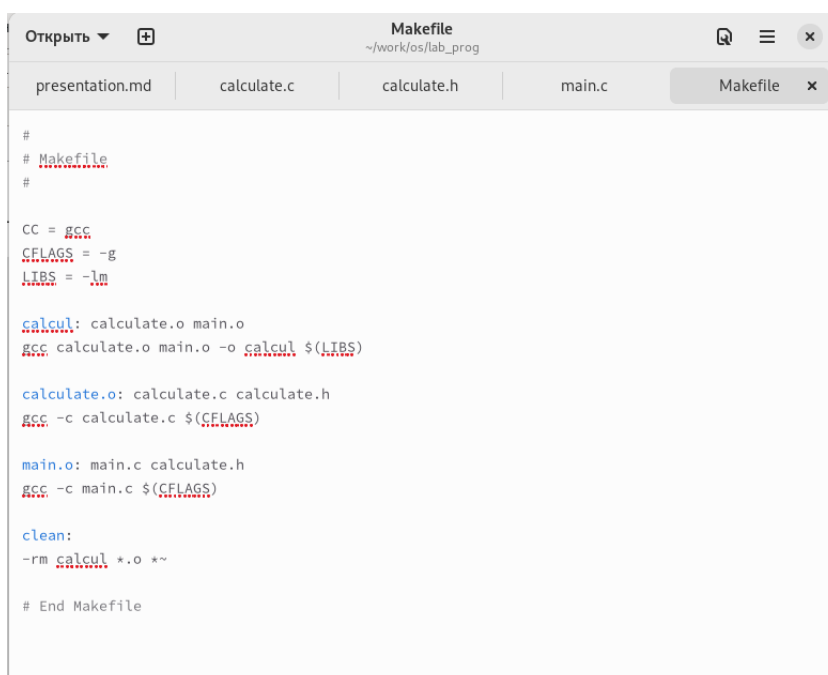
Рис. 3.5: main

3. Выполните компиляцию программы посредством gcc: (рис. 3.6)

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

Рис. 3.6: Компиляция

4. При необходимости исправьте синтаксические ошибки.
5. Создайте Makefile со следующим содержанием: (рис. 3.7)



```
#
# Makefile
#

CC = gcc
CFLAGS = -g
LIBS = -lm

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

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

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

clean:
-rm calcul *.o *~

# End Makefile
```

Рис. 3.7: Makefile

6. С помощью gdb выполните отладку программы calcul (перед использованием gdb исправьте Makefile) (рис. 3.8, 3.9, 3.10, 3.11, 3.12, 3.13, 3.14, 3.15, 3.16, 3.17, 3.18)

```
palaciosfelipe@palaciosfelipe:~/work/os/lab_prog — gdb ./calcul
[palaciosfelipe@palaciosfelipe lab_prog]$ gdb ./calcul
GNU gdb (GDB) Fedora Linux 13.1-1.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...

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.
Downloading separate debug info for /home/palaciosfelipe/work/os/lab_prog/calcul
(No debugging symbols found in ./calcul)
(gdb) run
Starting program: /home/palaciosfelipe/work/os/lab_prog/calcul
```

Рис. 3.8: Calcul

```
palaciosfelipe@palaciosfelipe:~/work/os/lab_prog — gdb ./calcul
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Downloading separate debug info for /home/palaciosfelipe/work/os/lab_prog/calcul
(No debugging symbols found in ./calcul)
(gdb) run
Starting program: /home/palaciosfelipe/work/os/lab_prog/calcul
Downloading separate debug info for system-supplied DS0 at 0x7ffff7fc6000
Downloading separate debug info for /lib64/libm.so.6
Downloading separate debug info for /lib64/libc.so.6
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 20
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): -
Вычитаемое: 10
10.00
[Inferior 1 (process 7357) exited normally]
(gdb) 50
Undefined command: "50". Try "help".
(gdb) run
Starting program: /home/palaciosfelipe/work/os/lab_prog/calcul
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 50
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): *
Множитель: 20
1000.00
[Inferior 1 (process 7403) exited normally]
(gdb)
```

Рис. 3.9: Run

```
palaciosfeline@palaciosfeline:~/work/os/lab_prog — gdb ./calcul
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): *
Множитель: 10
100.00
[Inferior 1 (process 7762) 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) list
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 }
(gdb)
```

Рис. 3.10: list

```
palaciosfeline@palaciosfeline:~/work/os/lab_prog — gdb ./calcul
1  //////////////////////////////////////////////////
2  // main.c
3
4  #include <stdio.h>
5  #include "calculate.h"
6
7  int
8  main (void)
9  {
10     float Numeral;
(gdb) list
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 }
(gdb) list 12,15
12     float Result;
13     printf("Число: ");
14     scanf("%f",&Numeral);
15     printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");
(gdb)
```

Рис. 3.11: list строк 12-15

```

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

```

Рис. 3.12: list 20,29

```

(gdb) list calculate.c:20,27
20     printf("Вычитаемое: ");
21     scanf("%f",&SecondNumeral);
22     return(Numeral - SecondNumeral);
23 }
24 else if(strncmp(Operation, "*", 1) == 0)
25 {
26     printf("Множитель: ");
27     scanf("%f",&SecondNumeral);
(gdb) break 21
Breakpoint 1 at 0x40121e: file calculate.c, line 21.
(gdb)

```

Рис. 3.13: list 20,27, break 21

```

(gdb) info breakpoints
Num   Type       Disp Enb Address          What
1     breakpoint keep y  0x000000000040121e in Calculate
                                at calculate.c:21
(gdb)

```

Рис. 3.14: info breakpoints

```

(gdb) run
Starting program: /home/palaciosfelipe/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=0x7fffffffdf34 "-")
at calculate.c:21
21     scanf("%f",&SecondNumeral);
(gdb) backtrace
#0 Calculate (Numeral=5, Operation=0x7fffffffdf34 "-") at calculate.c:21
#1 0x00000000004014eb in main () at main.c:17
(gdb)

```

Рис. 3.15: backtrace

```
#1 0x000000000040142b in main () at main.c:17
(gdb) print Numeral
$1 = 5
(gdb)
```

Рис. 3.16: print Numeral

```
(gdb) print Numeral
$1 = 5
(gdb) display Numeral
1: Numeral = 5
(gdb)
```

Рис. 3.17: display Numera

```
(gdb) info breakpoints
Num   Type             Disp Enb Address                  What
1      breakpoint      keep y   0x000000000040121e in Calculate
                                           at calculate.c:21
breakpoint already hit 1 time
(gdb) delete 1
(gdb)
```

Рис. 3.18: info breakpoints delete 1

7. С помощью утилиты splint попробуйте проанализировать коды файлов calculate.c и main.c (рис. 3.19, 3.20)

```
[palaciosfelipe@palaciosfelipe lab_prog]$ splint calculate.c
\bash: splint: команда не найдена...
Установить пакет «splint», предоставляющий команду «splint»? [N/y] y

* Ожидание в очереди...
Следующие пакеты должны быть установлены:
splint-3.1.2-29.fc37.x86_64  An implementation of the lint program
Продолжить с этими изменениями? [N/y] y

* Ожидание в очереди...
* Ожидание аутентификации...
* Ожидание в очереди...
* Загрузка пакетов...
* Запрос данных...
* Проверка изменений...
```

Рис. 3.19: splint calculate.c

```

[palaciosfelipe@palaciosfelipe 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:14:4: 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:16:15: 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:16:12: Corresponding format code
main.c:16:4: Return value (type int) ignored: scanf("%s", &Ope...

Finished checking --- 4 code warnings
[palaciosfelipe@palaciosfelipe lab_prog]$

```

Рис. 3.20: splint main.c



## 4 Выводы

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