

## Лабораторная работа № 13

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

---

Ариоке Габриэль Одафе

01 Май 2023

Российский университет дружбы народов, Москва, Россия

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.

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`.
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: `gdb ./calcul`
  - Для запуска программы внутри отладчика введите команду `run: run`
  - Для постраничного (по 9 строк) просмотра исходного код используйте команду `list: 1 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. а команда `backtrace` покажет весь стек вызываемых функций от начала программы до текущего места.
  - 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
7. С помощью утилиты `splint` попробуйте проанализировать коды файлов `calculate.c` и `main.c`.

Процесс разработки программного обеспечения обычно разделяется на следующие этапы: - планирование, включающее сбор и анализ требований к функционалу и другим характеристикам разрабатываемого приложения;

- проектирование, включающее в себя разработку базовых алгоритмов и спецификаций, определение языка программирования;
- непосредственная разработка приложения;

- кодирование — по сути создание исходного текста программы (возможно в нескольких вариантах);
- анализ разработанного кода;
- сборка, компиляция и разработка исполняемого модуля;
- тестирование и отладка, сохранение произведённых изменений;
- документирование.

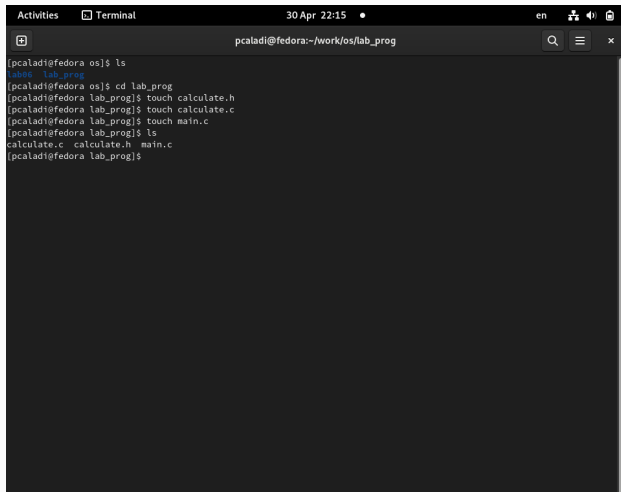
Для создания исходного текста программы разработчик может воспользоваться любым удобным для него редактором текста: vi, vim, mceditor, emacs, geany и др. После завершения написания исходного кода программы (возможно состоящей из нескольких файлов), необходимо её скомпилировать и получить исполняемый модуль.



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.

# Выполнение лабораторной работы

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

A screenshot of a Linux terminal window. The title bar shows 'Activities', 'Terminal', and the date '30 Apr 22:15'. The terminal content shows a user named 'pcaladi' at a 'fedora' machine. The user runs 'ls' and sees 'lab06' and 'lab\_prog'. Then they run 'cd lab\_prog'. Next, they run 'touch calculate.h', 'touch calculate.c', and 'touch main.c'. Finally, they run 'ls' again and see 'calculate.c', 'calculate.h', and 'main.c'.

```
[pcaladi@fedora os]$ ls
lab06  lab_prog
[pcaladi@fedora os]$ cd lab_prog
[pcaladi@fedora lab_prog]$ touch calculate.h
[pcaladi@fedora lab_prog]$ touch calculate.c
[pcaladi@fedora lab_prog]$ touch main.c
[pcaladi@fedora lab_prog]$ ls
calculate.c  calculate.h  main.c
[pcaladi@fedora lab_prog]$
```

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. 2, 3, 4)

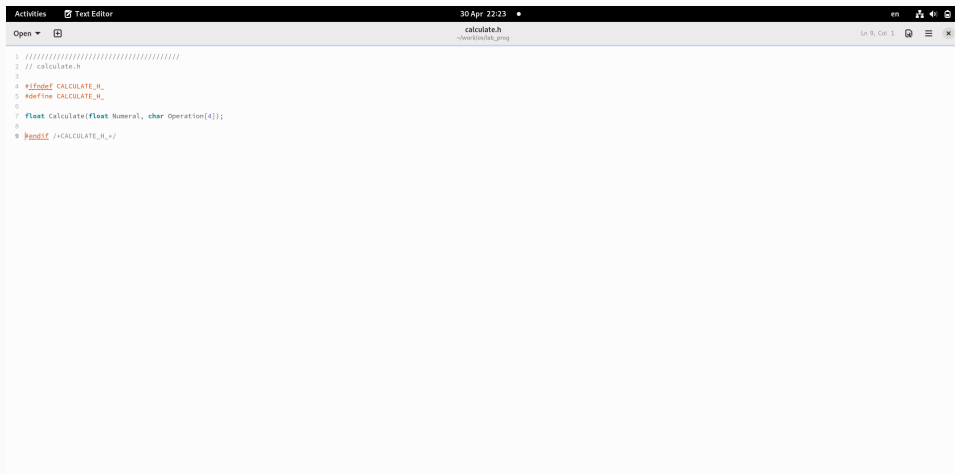
# Выполнение лабораторной работы



```
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("Добавление: ");
16         scanf("%f", &SecondNumeral);
17         return(Numeral + SecondNumeral);
18     }
19     else if(strncmp(Operation, "-", 1) == 0)
20     {
21         printf("Вычитание: ");
22         scanf("%f", &SecondNumeral);
23         return(Numeral - SecondNumeral);
24     }
25     else if(strncmp(Operation, "*", 1) == 0)
26     {
27         printf("Умножение: ");
28         scanf("%f", &SecondNumeral);
29         return(Numeral * SecondNumeral);
30     }
31     else if(strncmp(Operation, "/", 1) == 0)
32     {
33         printf("Деление: ");
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     }
43     else if(strncmp(Operation, "%", 1) == 0)
```

Figure 2: Текст программы

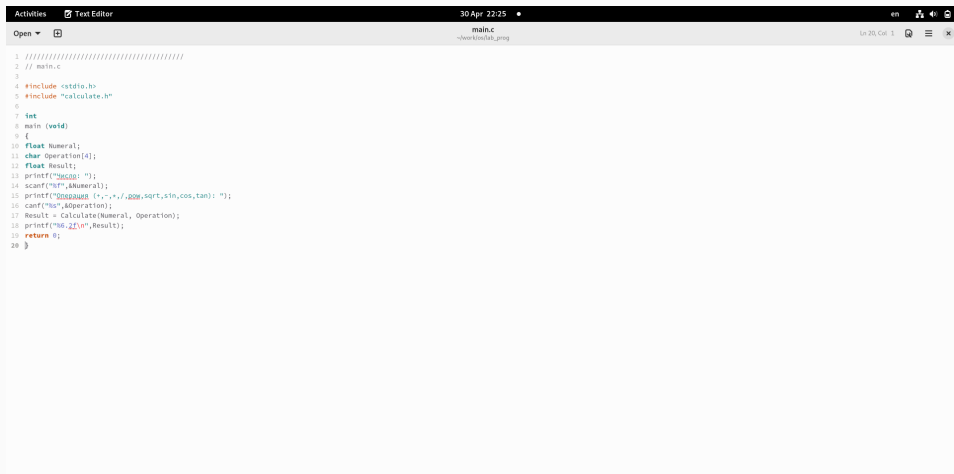
# Выполнение лабораторной работы



```
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_*/
```

Figure 3: Текст программы

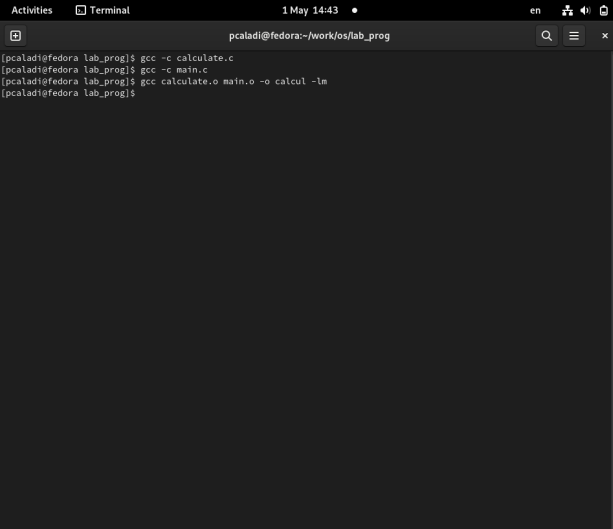
# Выполнение лабораторной работы



```
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("Введите: ");
14     scanf("%f", &Numeral);
15     printf("Операция (+, -, *, /, pow, sqrt, sin, cos, tan): ");
16     scanf("%s", &Operation);
17     Result = Calculate(Numeral, Operation);
18     printf("Вывод: %f\n", Result);
19     return 0;
20 }
```

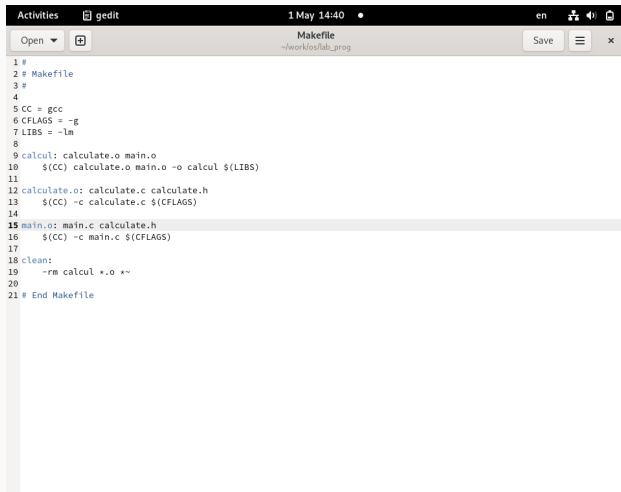
Figure 4: Текст программы

## 3. Выполните компиляцию программы посредством gcc. (рис. 5)



```
Activities Terminal 1 May 14:43 en
pcaladi@fedora:~/work/os/lab_prog
[pcaladi@fedora lab_prog]$ gcc -c calculate.c
[pcaladi@fedora lab_prog]$ gcc -c main.c
[pcaladi@fedora lab_prog]$ gcc calculate.o main.o -o calcul -lm
[pcaladi@fedora lab_prog]$
```

4. При необходимости исправьте синтаксические ошибки.
5. Создайте Makefile. (рис. 6)



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

# Выполнение лабораторной работы

```
Activities Terminal 1 May 15:12 en
pcaladi@fedora:~/work/os/lab_prog — gdb ./calcul

[pcaladi@fedora lab_prog]$ gdb ./calcul
GNU gdb (GDB) Fedora 12.1-2.fc36
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-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.
(No debugging symbols found in ./calcul)
(gdb) run
Starting program: /home/pcaladi/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): +
Второе слагаемое: 6
11.00
[Inferior 1 (process 3978) exited normally]
(gdb) 
```

- Для постраничного (по 9 строк) просмотра исходного код используйте команду `list`:
- Для просмотра строк с 12 по 15 основного файла используйте `list` с параметрами: `list 12,15` (рис. 8)

# Выполнение лабораторной работы

```
Activities Terminal 1 May 15:14 en
pcaladi@fedora:~/work/os/lab_prog — gdb ./calcul

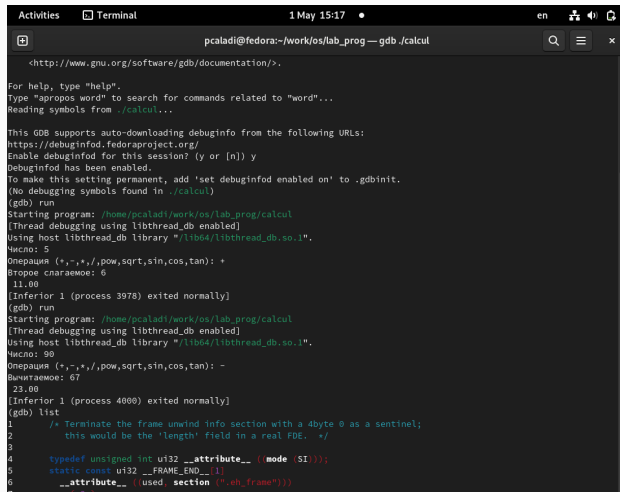
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

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.
(No debugging symbols found in ./calcul)
(gdb) run
Starting program: /home/pcaladi/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): +
Второе слагаемое: 6
11.00
[Inferior 1 (process 3978) exited normally]
(gdb) run
Starting program: /home/pcaladi/work/os/lab_prog/calcul
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 90
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): -
Вычитаемое: 67
23.00
[Inferior 1 (process 4000) exited normally]
(gdb) list
1      /* Terminate the frame unwind info section with a 4byte 0 as a sentinel;
2         this would be the 'length' field in a real FDE.  */
3
4      typedef unsigned int ui32 __attribute__((mode(SI)));
5      static const ui32 __FRAME_END__ = {
6          __attribute__((used, section(".eh_frame")))
7          = { 0 };
(gdb) list
Line number 8 out of range; sofini.c has 7 lines.
(gdb) list 12,15
Line number 12 out of range; sofini.c has 7 lines.
(gdb)
```

# Выполнение лабораторной работы

- Для просмотра определённых строк не основного файла используйте list с параметрами: list calculate.c:20,29 (рис. 9)



```
Activities Terminal 1 May 15:17 en
pcaladi@fedora:~/work/os/lab_prog — gdb ./calcul

<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.
(No debugging symbols found in ./calcul)
(gdb) run
Starting program: /home/pcaladi/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): +
Второе слагаемое: 6
11.18
[Inferior 1 (process 3978) exited normally]
(gdb) run
Starting program: /home/pcaladi/work/os/lab_prog/calcul
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 90
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): -
Вычитаемое: 67
23.00
[Inferior 1 (process 4000) exited normally]
(gdb) list
1      /* Terminate the frame unwind info section with a 4byte 0 as a sentinel;
2         this would be the 'length' field in a real FDE.  +/
3
4      typedef unsigned int ui32 __attribute__ ((mode (SI)));
5      static const ui32 __FRAME_END__[]
6      __attribute__ ((used, section (".eh_frame")))
```

- 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.
- Отладчик выдаст следующую информацию: `#0 Calculate (Numeral=5, Operation=0x7fffffffdd280 "-") at calculate.c:21 #1 0x0000000000400b2b in main () at main.c:17` а команда `backtrace` покажет весь стек вызываемых функций от начала программы до текущего места. (рис. 10)

# Выполнение лабораторной работы

```
Activities Terminal 1 May 15:19 • en
pcaladi@fedora:~/work/os/lab_prog — gdb ./calcul

2      this would be the 'length' field in a real FDE. */
3
4      typedef unsigned int u132 __attribute__((mode(SI)));
5      static const u132 __FRAME_END__[1]
6      __attribute__((used, section(".eh_frame")))
7      = { 0 };
(gdb) list
Line number 8 out of range; sofini.c has 7 lines.
(gdb) list 12,15
Line number 12 out of range; sofini.c has 7 lines.
(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
No line 21 in the current file.
Make breakpoint pending on future shared library load? (y or [n]) y
Breakpoint 1 (21) pending.
(gdb) run
Starting program: /home/pcaladi/work/os/lab_prog/calcul

Breakpoint 1, _dl_call_libc_early_init (libc_map=0x7ffff7fad500, initial=initial@entry=true) at dl-call-libc-early-init.c:27
27      (
(gdb) 4
Undefined command: "4". Try "help".
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/pcaladi/work/os/lab_prog/calcul

Breakpoint 1, _dl_call_libc_early_init (libc_map=0x7ffff7fad500, initial=initial@entry=true) at dl-call-libc-early-init.c:27
27      (
(gdb) backtrace
#0  _dl_call_libc_early_init (libc_map=0x7ffff7fad500, initial=initial@entry=true) at dl-call-libc-early-init.c:27
#1  0x00007ffff7fe7c13 in dl_main (phdr=<optimized out>, phnum=<optimized out>, user_entry=<optimized out>, auxv=<optimized out>) at rtld.c:2560
#2  0x00007ffff7fe3373 in _dl_sysdep_start (start_argptr=start_argptr@entry=0x7ffffffffffe060, dl_main=dl_main@entry=0x7ffff7fe5200 <dl_main>) at ../sysdeps/unix/sysv/linux/dl-sysdep.c:140
#3  0x00007ffff7fe4f9a in _dl_start_final (arg=0x7ffffffffffe060) at rtld.c:507
#4  _dl_start (arg=0x7ffffffffffe060) at rtld.c:596
#5  0x00007ffff7fe3d88 in _start () from /lib64/ld-linux-x86-64.so.2
#6  0x0000000000000001 in ?? ()
#7  0x00007ffff7fe3711 in ?? ()
#8  0x0000000000000000 in ?? ()
(gdb)
```

- 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 (рис. 11)



# Выполнение лабораторной работы

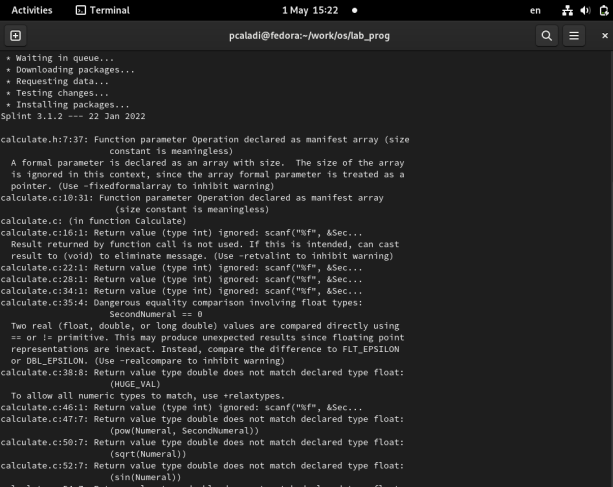
```
Activities Terminal 1 May 15:18 en
pcaladi@fedora:~/work/os/lab_prog — gdb ./calcul

[Inferior 1 (process 3978) exited normally]
(gdb) run
Starting program: /home/pcaladi/work/os/lab_prog/calcul
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 90
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): -
Вычитаемое: 67
23.00
[Inferior 1 (process 4000) exited normally]
(gdb) list
1      /* Terminate the frame unwind info section with a 4byte 0 as a sentinel;
2      this would be the 'length' field in a real FDE. */
3
4      typedef unsigned int ui32 __attribute__((mode(SI)));
5      static const ui32 __FRAME_END__[1]
6      __attribute__((used, section(".eh_frame")))
7      = { 0 };
(gdb) list
Line number 8 out of range; sofini.c has 7 lines.
(gdb) list 12,15
Line number 12 out of range; sofini.c has 7 lines.
(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
No line 21 in the current file.
Make breakpoint pending on future shared library load? (y or [n]) y
Breakpoint 1 (21) pending.
(gdb) run
Starting program: /home/pcaladi/work/os/lab_prog/calcul

Breakpoint 1, _dl_call_libc_early_init (libc_map=0x7ffff7fad500, initial=initial@entry=true) at dl-call-libc-early-in
it.c:27
27      {
(gdb) 4
Undefined command: "4". Try "help".
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/pcaladi/work/os/lab_prog/calcul

Breakpoint 1, _dl_call_libc_early_init (libc_map=0x7ffff7fad500, initial=initial@entry=true) at dl-call-libc-early-in
it.c:27
27      {
(gdb) 
```

7. С помощью утилиты splint попробуйте проанализировать коды файлов calculate.c и main.c. (рис. 12, 13)



```
Activities Terminal 1 May 15:22 en
pcaladi@fedora:~/work/os/lab_prog

* Waiting in queue...
* Downloading packages...
* Requesting data...
* Testing changes...
* Installing packages...
Splint 3.1.2 --- 22 Jan 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:16:1: 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:22:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:28:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:34:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:35:4: 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:38:8: Return value type double does not match declared type float:
      (HUGE_VAL)
      To allow all numeric types to match, use +relaxtypes.
calculate.c:46:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:47:7: Return value type double does not match declared type float:
      (pow(Numeral, SecondNumeral))
calculate.c:50:7: Return value type double does not match declared type float:
      (sqrt(Numeral))
calculate.c:52:7: Return value type double does not match declared type float:
      (sin(Numeral))
```

# Выполнение лабораторной работы

```
Activities Terminal 1 May 15:22 en
pcaladi@fedora:~/work/os/lab_prog

* Waiting in queue...
* Downloading packages...
* Requesting data...
* Testing changes...
* Installing packages...
Splint 3.1.2 --- 22 Jan 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:16:1: 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:22:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:28:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:34:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:35:4: 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:38:8: Return value type double does not match declared type float:
        (HUGE_VAL)
    To allow all numeric types to match, use +relaxtypes.
calculate.c:46:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:47:7: Return value type double does not match declared type float:
        (pow(Numeral, SecondNumeral))
calculate.c:50:7: Return value type double does not match declared type float:
        (sqrt(Numeral))
calculate.c:52:7: Return value type double does not match declared type float:
        (sin(Numeral))
calculate.c:54:7: Return value type double does not match declared type float:
        (cos(Numeral))
calculate.c:56:7: Return value type double does not match declared type float:
        (tan(Numeral))
calculate.c:60:7: Return value type double does not match declared type float:
        (HUGE_VAL)

Finished checking --- 15 code warnings

[pcaladi@fedora lab_prog]$
```

В процессе выполнения лабораторной работы я 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.

1. Лабораторная работа № 13. Средства, применяемые при разработке программного обеспечения в ОС типа UNIX Linux [Электронный ресурс]. URL:<https://esystem.rudn.ru/>.

Спасибо за внимание!