

Лабораторная работа No 13

Анастасия Павловна Баранова, НБИбд-01-21¹

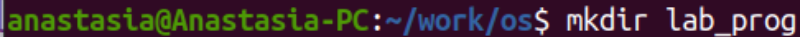
4 июня, Москва, 2022 г

¹Российский Университет Дружбы Народов

Целью данной лабораторной работы является приобрести простейшие навыки разработки, анализа, тестирования и отладки приложений в ОС типа UNIX/Linux на примере создания на языке программирования C калькулятора с простейшими функциями.

Выполнение лабораторной работы

В домашнем каталоге создам подкаталог ~/work/os/lab_prog.

A terminal window with a dark background. The prompt is 'anastasia@Anastasia-PC:~/work/os\$' in green and blue. The command 'mkdir lab_prog' is entered in white text.

```
anastasia@Anastasia-PC:~/work/os$ mkdir lab_prog
```

Figure 1: В домашнем каталоге создам подкаталог ~/work/os/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.

Создам в нём файлы: calculate.h, calculate.c, main.c.

```
anastasia@Anastasia-PC:~/work/os$ touch calculate.h
anastasia@Anastasia-PC:~/work/os$ touch calculate.c
anastasia@Anastasia-PC:~/work/os$ touch main.c
anastasia@Anastasia-PC:~/work/os$ ls
calculate.c  calculate.h  lab06  lab_prog  main.c
anastasia@Anastasia-PC:~/work/os$
```

Figure 2: Создам в нём файлы: calculate.h, calculate.c, main.c.

Реализация функций калькулятора в файле calculate.c

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

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     }
43     else if(strncmp(Operation, "pow", 3) == 0)
44     {
45         printf("Степень: ");
46         scanf("%f", &SecondNumeral);
47         return(pow(Numeral, SecondNumeral));
48     }
49     else if(strncmp(Operation, "sqrt", 4) == 0)
50         return(sqrt(Numeral));
51     else if(strncmp(Operation, "sin", 3) == 0)
52         return(sin(Numeral));
53     else if(strncmp(Operation, "cos", 3) == 0)
54         return(cos(Numeral));
55 }
```

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



The image shows a code editor window with a dark theme. The title bar at the top right says "calculate.h" and the path below it is "~/work/os/lab_prog". The editor contains the following C header file code:

```
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 4: Интерфейсный файл calculate.h.

Основной файл main.c, реализующий интерфейс пользователя к калькулятору



```
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 }
```

Figure 5: Основной файл main.c.

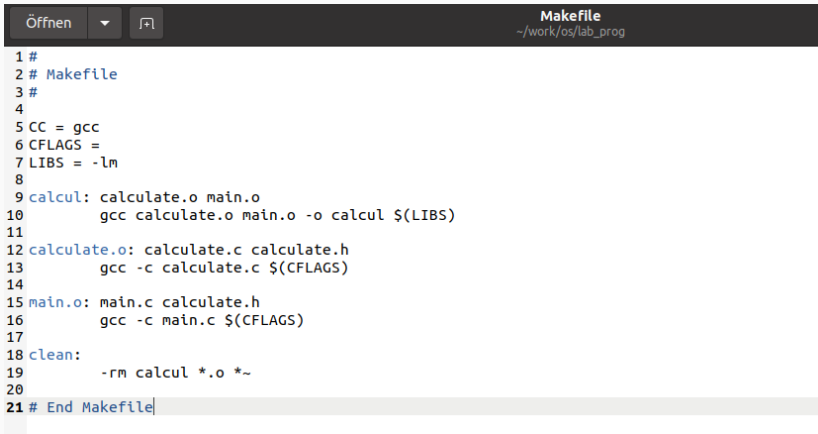
Выполню компиляцию программы посредством gcc

```
anastasia@Anastasia-PC:~/work/os/lab_prog$ gcc -c calculate.c
anastasia@Anastasia-PC:~/work/os/lab_prog$ gcc -c main.c
main.c: In function 'main':
main.c:16:11: warning: format '%s' expects argument of type 'char *', but argument 2 has type 'char (*)[4]' [-Wformat=]
   16 |     scanf("%s",&operation);
      |           ^~
      |           |
      |           | char (*)[4]
      |           char *
```

anastasia@Anastasia-PC:~/work/os/lab_prog\$ gcc calculate.o main.o -o calcul -lm

Figure 6: Выполню компиляцию программы посредством gcc.

Создам Makefile со следующим содержанием



```
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
```

Figure 7: Создам Makefile со следующим содержанием.

В содержании файла указаны флаги компиляции, тип компилятора и файлы, которые должен собрать сборщик.

С помощью gdb выполняю отладку программы calcul (перед использованием gdb исправлю Makefile):

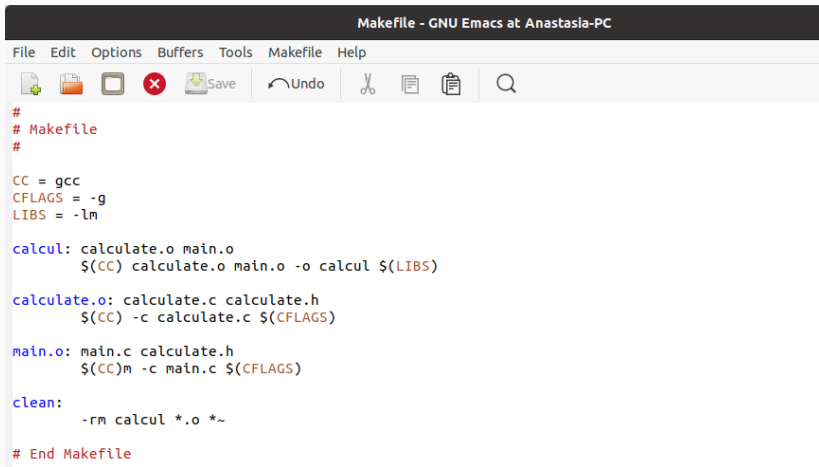
Запущу отладчик GDB, загрузив в него программу для отладки: `gdb ./calcul`

```
anastasia@Anastasia-PC:~/work/os/lab_prog$ gdb ./calcul
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 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...
(gdb) |
```

Figure 8: Запущу отладчик GDB.

Перед использованием gdb исправлю Makefile



```
#  
# Makefile  
#  
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
```

Figure 9: Перед использованием gdb исправлю Makefile.

Для запуска программы внутри отладчика введу команду run: run

```
(gdb) run
Starting program: /home/anastasia/work/os/lab_prog/calcul
Число: 3
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): +
Второе слагаемое: 6
9.00
[Inferior 1 (process 65651) exited normally]
(gdb) █
```

Figure 10: Для запуска программы внутри отладчика введу команду run.

Для постраничного (по 9 строк) просмотра исходного кода использую команду list: list

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

Figure 11: Для постраничного просмотра кода использую команду list.

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

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

```
(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) |
```

Figure 12: Использу list с параметрами: list calculate.c:20,29.

Устанавливаю точку останова в файле `calculate.c` на строке номер 21:

```
list calculate.c:20,27 break 21
```

Устанавливаю точку останова в файле calculate.c на строке номер 21

```
(gdb) list calculate.c:20,27
20      {
21          printf("Вычитаемое: ");
22          scanf("%f",&SecondNumeral);
23          return(Numeral - SecondNumeral);
24      }
25      else if(strncmp(Operation, "*", 1) == 0)
26      {
27          printf("Множитель: ");
(gdb) break 21
Haltepunkt 1 at 0x555555552dd: file calculate.c, line 21.
(gdb) █
```

Figure 13: Устанавливаю точку останова.

Выведу информацию об имеющихся в проекте точках останова: info breakpoints

```
(gdb) info breakpoints
Num      Type           Disp Enb Address                  What
1        breakpoint    keep y   0x0000555555552dd in calculate at calculate.c:21
(gdb) █
```

Figure 14: Выведу информацию об имеющихся точках останова.

Запущу программу внутри отладчика и проверю, что программа остановится в момент прохождения точки останова:

run 5 - backtrace

Запущу программу внутри отладчика и проверю, что программа остановится в момент прохождения точки останова

```
(gdb) run
Starting program: /home/anastasia/work/os/lab_prog/calcul
Число: 5
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): -

Breakpoint 1, Calculate (Numeral=5, Operation=0x7fffffffddc4 "-") at calculate.c:21
21      printf("Вычитаемое: ");
(gdb) backtrace
#0 Calculate (Numeral=5, Operation=0x7fffffffddc4 "-") at calculate.c:21
#1 0x00005555555555bd in main () at main.c:17
(gdb) █
```

Figure 15: Запущу программу внутри отладчика.

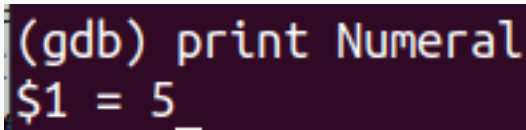
Отладчик выдал следующую информацию:

#0 Calculate (Numeral=5, Operation=0x7fffffff280 "-") at calculate.c:21 #1
0x000000000400b2b in main () at main.c:17 а команда backtrace показала
весь стек вызываемых функций от начала программы до текущего
места.

Посмотрю, чему равно на этом этапе значение переменной Numeral,
введя: `print Numeral`

На экран было выведено число 5.

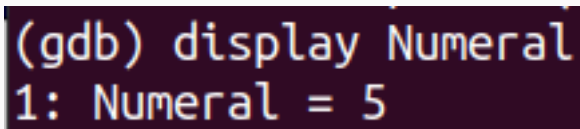
Посмотрю, чему равно на этом этапе значение переменной Numeral, введя: `print Numeral`



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

Figure 16: Посмотрю, чему равно на этом этапе значение переменной Numeral.

Сравню с результатом вывода на экран после использования команды:
`display Numeral`



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

Figure 17: Сравню с результатом вывода на экран после использования команды: `display Numeral`.

info breakpoints delete 1

```
(gdb) info breakpoints
Num      Type             Disp Enb Address                  What
1        breakpoint      keep y   0x000005555555552dd in Calculate at calculate.c:21
          breakpoint already hit 1 time
(gdb) delete 1
(gdb)
```

Figure 18: Уберу точки останова.

С помощью утилиты splint проанализирую коды файлов calculate.c и main.c

```
anastasia@Anastasia-PC:~/work/os/lab_prog$ splint calculate.c
Splint 3.1.2 --- 20 Feb 2018

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:16:7: 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:22:7: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:28:7: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:34:7: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:35:10: 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:38:17: Return value type double does not match declared type float:
    (HUGE_VAL)
    To allow all numeric types to match, use +relaxtypes.
calculate.c:46:6: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:47:12: Return value type double does not match declared type float:
    (pow(Numeral, SecondNumeral))
calculate.c:50:11: Return value type double does not match declared type float:
    (sqrt(Numeral))
calculate.c:52:11: Return value type double does not match declared type float:
    (sin(Numeral))
calculate.c:54:11: Return value type double does not match declared type float:
    (cos(Numeral))
calculate.c:56:11: Return value type double does not match declared type float:
    (tan(Numeral))
calculate.c:60:13: Return value type double does not match declared type float:
    (HUGE_VAL)

Finished checking --- 15 code warnings
anastasia@Anastasia-PC:~/work/os/lab_prog$
```

С помощью утилиты splint анализирую коды файлов calculate.c и main.c

```
anastasia@Anastasia-PC:~/work/os/lab_prog$ splint main.c
Splint 3.1.2 --- 20 Feb 2018

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:14:3: 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:16:14: 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:16:11: Corresponding format code
main.c:16:3: Return value (type int) ignored: scanf("%s", &Ope...

Finished checking --- 4 code warnings
anastasia@Anastasia-PC:~/work/os/lab_prog$
```

Figure 20: С помощью утилиты splint анализирую код файла main.c.

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