

Лабораторная работа 13

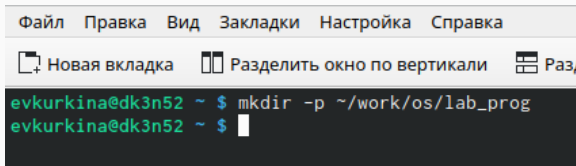
Куркина ЕВгения Вячеславовна

¹RUDN University, Moscow, Russian Federation

Лабораторная работа 13

Цель данной лабораторной работы — 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.

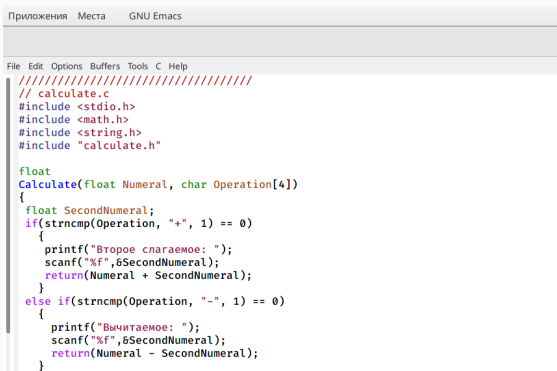
В домашнем каталоге создаю подкаталог `~/work/os/lab_prog` (рис. 1).

A screenshot of a terminal window with a light gray title bar containing menu items: 'Файл', 'Правка', 'Вид', 'Закладки', 'Настройка', and 'Справка'. Below the title bar is a toolbar with icons and labels: 'Новая вкладка', 'Разделить окно по вертикали', and 'Раз...'. The terminal area has a black background with green text. It shows the prompt 'evkurkina@dk3n52 ~ \$' followed by the command 'mkdir -p ~/work/os/lab_prog' and a second prompt 'evkurkina@dk3n52 ~ \$' with a cursor.

```
evkurkina@dk3n52 ~ $ mkdir -p ~/work/os/lab_prog
evkurkina@dk3n52 ~ $
```

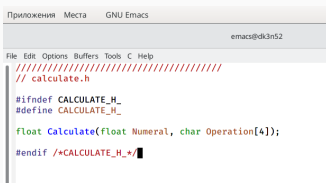
Figure 1: Создание подкаталога

Создала файлы calculate.h, calculate.c, main.c. Написала текст примитивнейшего калькулятора, который 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).



```
////////////////////////////////////  
// calculate.c  
#include <stdio.h>  
#include <math.h>  
#include <string.h>  
#include "calculate.h"  
  
float  
Calculate(float Numeral, char Operation[4])  
{  
    float SecondNumeral;  
    if(strncmp(Operation, "+", 1) == 0)  
    {  
        printf("Второе слагаемое: ");  
        scanf("%f", &SecondNumeral);  
        return(Numeral + SecondNumeral);  
    }  
    else if(strncmp(Operation, "-", 1) == 0)  
    {  
        printf("Вычитаемое: ");  
        scanf("%f", &SecondNumeral);  
        return(Numeral - SecondNumeral);  
    }  
}
```

Далее написала интерфейсный файл `calculate.h`, описывающий формат вызова функции калькулятора (рис. 3), а также текст основного файла `main.c`, реализующий интерфейс пользователя к калькулятору (рис. 4).



```
Приложения  Места  GNU Emacs
emacs@dk3n52

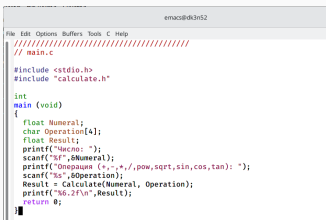
File Edit Options Buffers Tools C Help
////////////////////////////////////
// calculate.h

#ifndef CALCULATE_H_
#define CALCULATE_H_

float Calculate(float Numeral, char Operation[4]);

#endif /*CALCULATE_H_*/
```

Figure 3: Текст интерфейсного файла



```
emacs@dk3n52

File Edit Options Buffers Tools C Help
////////////////////////////////////
// main.c

#include <stdio.h>
#include "calculate.h"

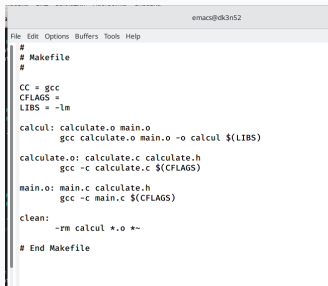
int
main (void)
{
    float Numeral;
    char Operation[4];
    float Result;
    printf("Число: ");
    scanf("%f",&Numeral);
    printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");
    scanf("%s",&Operation);
    Result = Calculate(Numeral, Operation);
    printf("%g.2f\n",Result);
    return 0;
}
```

Выполнила компиляцию программы посредством gcc (рис. 5).

```
evkurkina@edk3n52 ~/work/os/lab_prog $ gcc -c -g calculate.c
evkurkina@edk3n52 ~/work/os/lab_prog $ gcc -c -g main.c
evkurkina@edk3n52 ~/work/os/lab_prog $ gcc -g calculate.o main.o -o calcul -lm
evkurkina@edk3n52 ~/work/os/lab_prog $
```

Figure 5: Компиляция программы

Создала Makefile переписала в него данный текст (рис. 6), затем изменила его до рабочего состояния рис. 7).



```
emacs@dk3n52
File Edit Options Buffers Tools Help
#
# Makefile
#
CC = gcc
CFLAGS =
LIBS = -lm

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

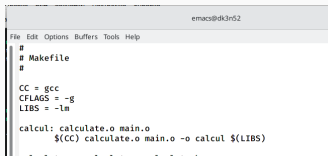
calculate.o: calculate.c calculate.h
gcc -c calculate.c $(CFLAGS)

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

clean:
rm calcul *.o *~

# End Makefile
```

Figure 6: Исходный текст



```
emacs@dk3n52
File Edit Options Buffers Tools Help
#
# Makefile
#
CC = gcc
CFLAGS = -g
LIBS = -lm

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


Данный файл необходим для автоматической компиляции файлов `calculate.c`. В переменную `GLASS` добавила опцию `-g`, утилита компиляции выбирается с помощью переменной `CC`. Далее я удалила файлы из каталога и выполнила компиляцию файлов (рис. 8)(рис. 9)

```
evkurkina@dk3n52 ~/work/os/lab_prog $ make clean
rm calcul *.o *~
evkurkina@dk3n52 ~/work/os/lab_prog $ make calculate.o
gcc -c calculate.c -g
evkurkina@dk3n52 ~/work/os/lab_prog $ make main.o
gcc -c main.c -g
```

Figure 8: удаление файлов и компиляция файлов

```
evkurkina@dk3n52 ~/work/os/lab_prog $ make calcul
gcc calculate.o main.o -o calcul -lm
```

Figure 9: Компиляция файлов

Запустила отладчик GDB, загрузив в него программу для отладки:gdb
./calcul(рис. 10).

```
evkurkina@dk3n52 ~/work/os/lab_prog $ gdb ./calcul
GNU gdb (Gentoo 11.2 vanilla) 11.2
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-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
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) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/e/v/evkurkina/work/os/lab_prog/calcul
Число: █
```

Figure 10: Запуск отладчика

Запустила программу внутри отладчика командой run (рис. 11).

```
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/e/v/evkurkina/work/os/lab_prog/calcul
Число: 6
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): *
Множитель: 5
30.00
[Inferior 1 (process 7604) exited normally]
(gdb) █
```

Figure 11: Команда run

Для постраничного (по 9 строк) просмотра исходного код использовала команду list(рис. 12).

```
(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("%.2f\n",Result);
19     return 0;
20 }
(gdb) █
```

Figure 12: Команда list

Для просмотра строк с 12 по 15 основного файла используйте list с параметрами: list 12,15(рис. 13). Для просмотра определённых строк не основного файла используйте list с параметрами: list calculate.c:20,29(рис. 14).

```
(gdb) list 12,15
12      float Result;
13      printf("Введите: ");
14      scanf("%f",&Numeral);
15      printf("Вычисляю (%f * %f) / pow(sqrt(x),sin(cos,tan)): ");
(gdb) █
```

Figure 13: Команда просмотра с 12 по 15 строк

```
(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("Умножитель: ");
27      scanf("%f",&SecondNumeral);
28      return(Numeral * SecondNumeral);
29  }
(gdb) █
```

Figure 14: Команда просмотра определенных строк

Установила точку отладки на 21 строке: `list calculate.c:20,27 break 21` (рис. 15).

```
(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 0x5555555525f: file calculate.c, line 21.
(gdb) █
```

Figure 15: Установка точки останова

Вывела информацию об имеющихся точках останова:info breakpoints
(рис. 16).

```
(gdb) info breakpoints
Num   Type       Disp Enb Address            What
1     breakpoint keep y   0x00005555555525f in calculate at calculate.c:21
(gdb) 
```

Figure 16: Информация о точках останова

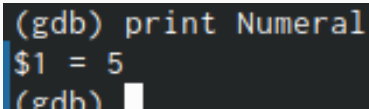
Запустила программу внутри отладчика и убедилась, что программа остановится в момент прохождения точки останова: run 5 - backtrace
Отладчик выдал следующую информацию: 1 #0 Calculate (Numeral=5, Operation=0x7fffffffcd280 "-") 2 at calculate.c:21 3 #1 0x0000000000400b2b in main () at main.c:17(рис. 17).

```
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/e/v/evkurkina/work/os/lab_prog/calcul
Число: 5
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): -

Breakpoint 1, Calculate (Numeral=5, Operation=0x7fffffffcd44 "-") at calculate.c:21
21      scanf("%f",&SecondNumeral);
(gdb) backtrace
#0 Calculate (Numeral=5, Operation=0x7fffffffcd44 "-") at calculate.c:21
#1 0x00005555555555a9 in main () at main.c:17
(gdb) █
```

Figure 17: Проверка точки останова

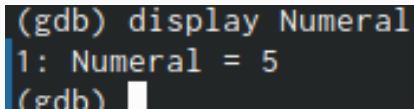
Просмотрела, чему равно на этом этапе значение переменной Numeral:
print Numeral (рис. 18).



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

Figure 18: Значение переменной

Сравнила с результатом вывода на экран после использования команды: `display Numeral` (рис. 19).



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

Figure 19: Сравнение результатов

Убрала точку останова `info breakpoints delete` (рис. 20).

```
(gdb) info breakpoints
Num      Type             Disp Enb Address            What
1        breakpoint      keep y   0x00005555555525f in calculate at calculate.c:21
          breakpoint already hit 1 time
(gdb) delete 1
(gdb) █
```

Figure 20: Удаление точки останова

С помощью утилиты splint проанализировала коды файлов calculate.c и main.c (рис. 21).(рис. 22).

```
svkurkin@mk3052 ~/work/os/lab_prog $ splint calculate.c
Splint 3.1.2 --- 13 Jan 2021

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:9:31: Function parameter Operation declared as manifest array (size
      constant is meaningless)
calculate.c: (in function Calculate)
calculate.c:15:5: 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:21:6: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:27:6: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:33:6: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:34:9: 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:37:12: Return value type double does not match declared type float:
      (HUGE_VAL)
To allow all numeric types to match, use *relaxtypes.
calculate.c:45:6: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:46:12: Return value type double does not match declared type float:
      (pow(Numeral, SecondNumeral))
calculate.c:49:9: Return value type double does not match declared type float:
      (sqrt(Numeral))
calculate.c:51:9: Return value type double does not match declared type float:
      (sin(Numeral))
calculate.c:53:9: Return value type double does not match declared type float:
      (cos(Numeral))
calculate.c:55:9: Return value type double does not match declared type float:
      (tan(Numeral))
calculate.c:59:12: Return value type double does not match declared type float:
      (HUGE_VAL)

Finished checking --- 15 code warnings
svkurkin@mk3052 ~/work/os/lab_prog $
```

Figure 21: Анализ кода файла calculate.c

```
svkurkin@mk3052 ~/work/os/lab_prog $ splint main.c
Splint 3.1.2 --- 13 Jan 2021

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)
```

Во время выполнения данной лабораторной работы, я приобрела простейшие навыки разработки, анализа, тестирования и отладки приложений ОС UNIX/LINUX на примере создания на языке программирования С калькулятора с простейшими функциями.

