

# Laboratory №14

---

Kulikov Maksim

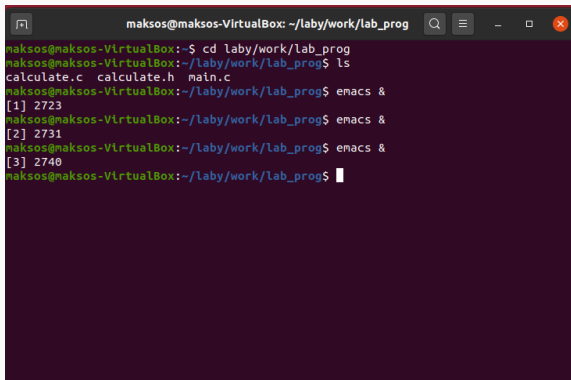
MAY-2021

RUDN University, Moscow, Russian Federation

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.

1. Ознакомиться с теоретическим материалом.
2. Выполнить работу.

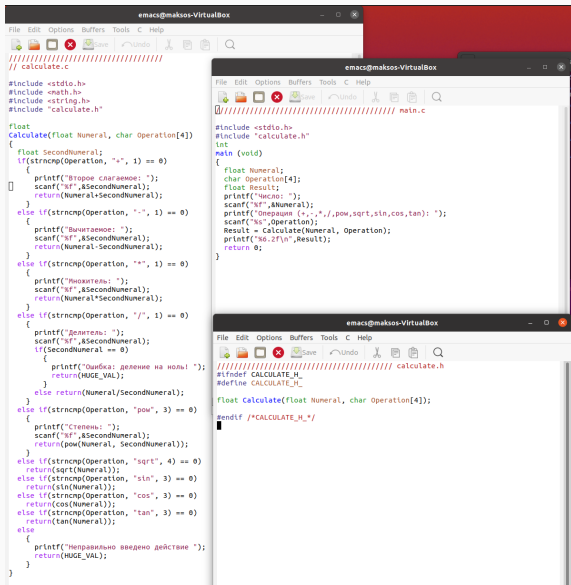
1. Создаю новый каталог и создаю 3 скрипта (рис. -fig. 1)



```
maksos@maksos-VirtualBox: ~/laby/work/lab_prog
maksos@maksos-VirtualBox:~$ cd laby/work/lab_prog
maksos@maksos-VirtualBox:~/laby/work/lab_prog$ ls
calculate.c  calculate.h  main.c
maksos@maksos-VirtualBox:~/laby/work/lab_prog$ emacs &
[1] 2723
maksos@maksos-VirtualBox:~/laby/work/lab_prog$ emacs &
[2] 2731
maksos@maksos-VirtualBox:~/laby/work/lab_prog$ emacs &
[3] 2740
maksos@maksos-VirtualBox:~/laby/work/lab_prog$
```

Рис. 1: Создание файлов

## 2. Три скрипта. (рис. -fig. 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);
    }
    else if(strncmp(Operation, "*", 1) == 0)
    {
        printf("Умножить: ");
        scanf("%f", &SecondNumeral);
        return(Numeral*SecondNumeral);
    }
    else if(strncmp(Operation, "/", 1) == 0)
    {
        printf("Делитель: ");
        scanf("%f", &SecondNumeral);
        if(SecondNumeral == 0)
        {
            printf("Ошибка: деление на ноль!");
            return(HUGE_VAL);
        }
        else return(Numeral/SecondNumeral);
    }
    else if(strncmp(Operation, "pow", 3) == 0)
    {
        printf("Степень: ");
        scanf("%f", &SecondNumeral);
        return(pow(Numeral, SecondNumeral));
    }
    else if(strncmp(Operation, "sqrt", 4) == 0)
    {
        return(sqrt(Numeral));
    }
    else if(strncmp(Operation, "sin", 3) == 0)
    {
        return(sin(Numeral));
    }
    else if(strncmp(Operation, "cos", 3) == 0)
    {
        return(cos(Numeral));
    }
    else if(strncmp(Operation, "tan", 3) == 0)
    {
        return(tan(Numeral));
    }
    else
    {
        printf("Неправильно введено действие ");
        return(HUGE_VAL);
    }
}

##### 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;
}

##### calculate.h

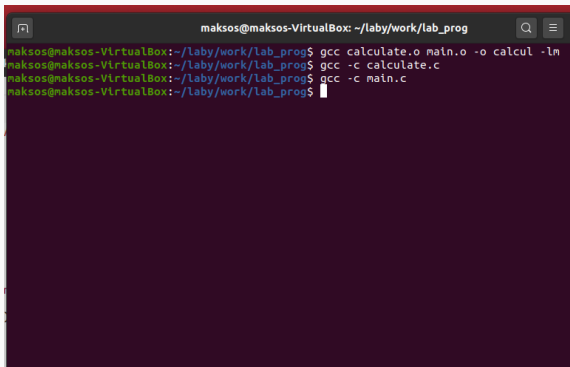
#ifndef CALCULATE_H
#define CALCULATE_H

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

#endif /*CALCULATE_H*/
```

Рис. 2: Скрипты

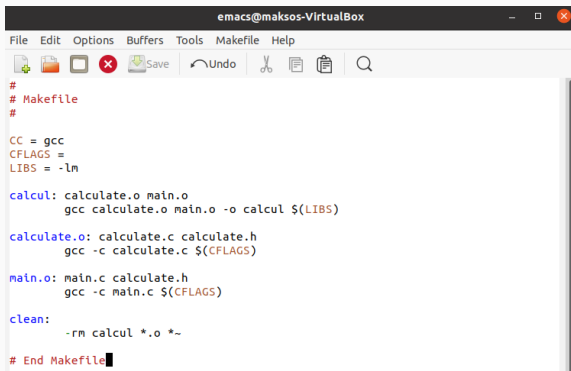
### 3. Выполняю компиляцию файлов. (рис. -fig. 3)



```
maksos@maksos-VirtualBox: ~/laby/work/lab_prog
maksos@maksos-VirtualBox:~/laby/work/lab_prog$ gcc calculate.o main.o -o calcul -lm
maksos@maksos-VirtualBox:~/laby/work/lab_prog$ gcc -c calculate.c
maksos@maksos-VirtualBox:~/laby/work/lab_prog$ gcc -c main.c
maksos@maksos-VirtualBox:~/laby/work/lab_prog$
```

Рис. 3: Компиляция

5. Создание Makefile. Данный файл необходим для автоматической компиляции файлов calculate.c(цельcalculate.o), main.c(цельmain.o), а также их объединения в один исполняемый файл calcul(цельcalcul). 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 (рис. -fig. 4)



```
#
# Makefile
#

CC = gcc
CFLAGS =
LIBS = -lm

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

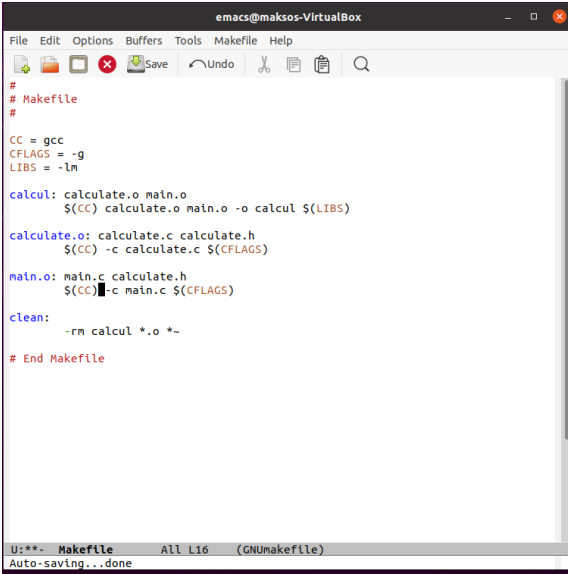
calculate.o: calculate.c calculate.h
gcc -c calculate.c $(CFLAGS)

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

clean:
-rm calcul *.o *~

# End Makefile
```

## 6. Вношу изменения в файл. (рис. -fig. 5)



```
#
# 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) -c main.c $(CFLAGS)

clean:
    -rm calcul *.o *~

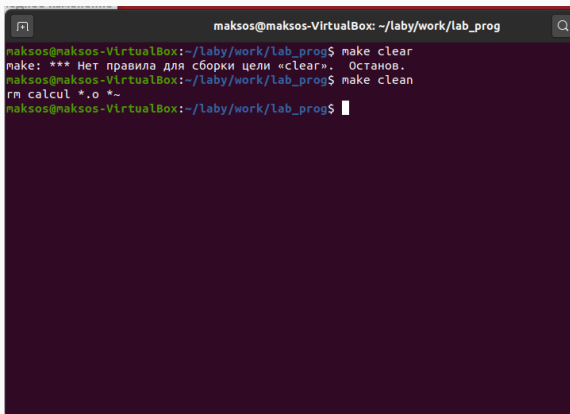
# End Makefile
```

U:\*\*- Makefile All L16 (GNUmakefile)  
Auto-saving...done

Рис. 5: Изменение содержимого



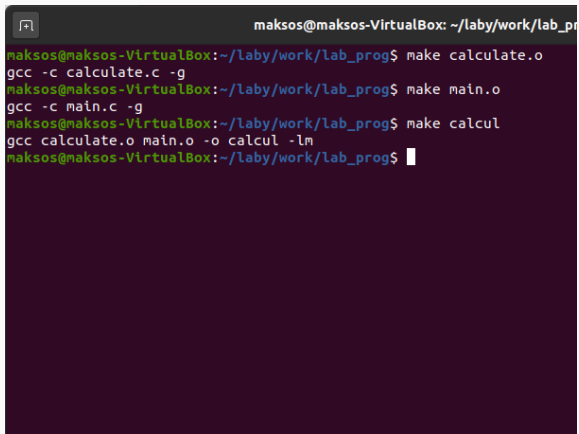
7. Выполняю make clean. Удаляю ненужные файлы. (рис. -fig. 6)

A terminal window with a dark purple background. The title bar at the top reads 'maksos@maksos-VirtualBox: ~/laby/work/lab\_prog'. The terminal shows the following sequence of commands and output:

```
maksos@maksos-VirtualBox:~/laby/work/lab_prog$ make clear
make: *** Нет правила для сборки цели «clear». Останов.
maksos@maksos-VirtualBox:~/laby/work/lab_prog$ make clean
rm calcul *.o *~
maksos@maksos-VirtualBox:~/laby/work/lab_prog$
```

Рис. 6: make clean

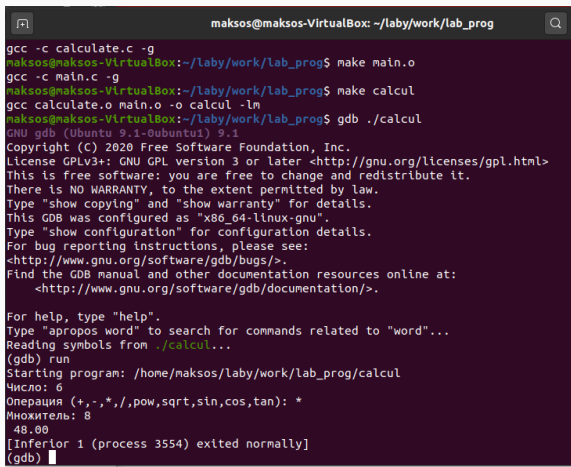
## 8. Конвертирую все файлы (рис. -fig. 7)

A terminal window with a dark background and light-colored text. The window title is 'maksos@maksos-VirtualBox: ~/laby/work/lab\_prog'. The terminal shows a series of commands and their outputs. The first command is 'make calculate.o', which runs 'gcc -c calculate.c -g'. The second command is 'make main.o', which runs 'gcc -c main.c -g'. The third command is 'make calcul', which runs 'gcc calculate.o main.o -o calcul -lm'. The prompt returns to the shell after each command.

```
maksos@maksos-VirtualBox: ~/laby/work/lab_prog$ make calculate.o
gcc -c calculate.c -g
maksos@maksos-VirtualBox:~/laby/work/lab_prog$ make main.o
gcc -c main.c -g
maksos@maksos-VirtualBox:~/laby/work/lab_prog$ make calcul
gcc calculate.o main.o -o calcul -lm
maksos@maksos-VirtualBox:~/laby/work/lab_prog$
```

Рис. 7: Конвертация

## 9. Тестирую программу (рис. -fig. 8)



```
maksos@maksos-VirtualBox: ~/laby/work/lab_prog
gcc -c calculate.c -g
maksos@maksos-VirtualBox:~/laby/work/lab_prog$ make main.o
gcc -c main.c -g
maksos@maksos-VirtualBox:~/laby/work/lab_prog$ make calcul
gcc calculate.o main.o -o calcul -lm
maksos@maksos-VirtualBox:~/laby/work/lab_prog$ gdb ./calcul
GNU gdb (Ubuntu 9.1-0ubuntu1) 9.1
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) run
Starting program: /home/maksos/laby/work/lab_prog/calcul
Число: 6
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): *
Множитель: 8
48.00
[Inferior 1 (process 3554) exited normally]
(gdb)
```

Рис. 8: Тест

## 10. Командой list вывожу содержимое файлов (рис. -fig. 6)

```
[inferior 1 (process 3334) exited normally]
(gdb) list
1  ////////////////////////////////////////////////// main.c
2
3  #include <stdio.h>
4  #include "calculate.h"
5  int
6  main (void)
7  {
8      float Numeral;
9      char Operation[4];
10     float Result;
(gdb) list 12,15
12     scanf("%f",&Numeral);
13     printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");
14     scanf("%s",Operation);
15     Result = Calculate(Numeral, Operation);
(gdb) list calculate.c 12,18
Function "calculate.c 12" not defined.
(gdb) list calculate.c:12,18
12     float SecondNumeral;
13     if(strncmp(Operation, "+", 1) == 0)
14     {
15         printf("Второе слагаемое: ");
16         scanf("%f",&SecondNumeral);
17         return(Numeral+SecondNumeral);
18     }
(gdb) █
```

Рис. 9: list

11. Ставлю breakpoint. Запускаю программу и проверяю работает ли он. Работает. Удаляю его (рис. -fig. 10)

```
(gdb) list calculate.c:20,27
20      {
21          printf("Вчитаемое: ");
22          scanf("%f",&SecondNumeral);
23          return(Numeral-SecondNumeral);
24      }
25      else if(strncmp(Operation, "*", 1) == 0)
26      {
27          printf("Множитель: ");
(gdb) break 21
Breakpoint 1 at 0x555555552dd: file calculate.c, line 21.
(gdb) info breakpoints
Num     Type             Disp Enb Address            What
1       breakpoint       keep y   0x0000555555552dd in Calculate at calculate.c:21
(gdb) run
Starting program: /home/maksos/laby/work/lab_prog/calcul
Число: 9
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): -
Breakpoint 1, Calculate (Numeral=9, Operation=0x7fffffffdf64 "-") at calculate.c:21
21      printf("Вчитаемое: ");
(gdb) print Numeral
$1 = 9
(gdb) display Numeral
1: Numeral = 9
(gdb) info breakpoints
Num     Type             Disp Enb Address            What
1       breakpoint       keep y   0x0000555555552dd in Calculate at calculate.c:21
        breakpoint already hit 1 time
(gdb) delete 1
(gdb)
```

Рис. 10: breakpoint

13. Устанавливаю splint . Запускаю его для calculate.c и main.c. 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. (рис. -fig. 11, рис. -fig. 12)

```
maksos@maksos-VirtualBox:~/laby/work/lab_prog$ splint calculate.c
Splint 3.1.2 --- 20 Feb 2018

calculate.h:5:37: 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:38:10: Return value type double does not match declared type float:
    (HUGE_VAL)
    To allow all numeric types to match, use +relaxtypes.
calculate.c:45:7: Return value (type int) ignored: scanf("%f", &Sec...
```

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

## Контрольные вопросы

1. Как получить более полную информацию о программах: gcc, make, gdb и др.? Дополнительную информацию о этих программах можно получить с помощью функций info и man.
2. Назовите и дайте краткую характеристику основным этапам разработки приложений в UNIX? Unix поддерживает следующие основные этапы разработки приложений: -создание исходного кода программы;
  - представляется в виде файла -сохранение различных вариантов исходного текста; -анализ исходного текста; Необходимо отслеживать изменения исходного кода, а также при работе более двух программистов над проектом программы нужно, чтобы они не делали изменений кода в одно время. -компиляция исходного текста и построение исполняемого модуля; -тестирование и отладка; -проверка кода на наличие ошибок -сохранение всех изменений. выполняемых при тестировании и отладке.