

# Презентация выполнения

Лабораторной работы №13

---

Атанесов Александр

5 Мая 2023

Российский университет дружбы народов, Москва, Россия

РУДН

## Информация

---

- Атанесов Александр Николаевич
- Студент первого курса НБИ-01-22
- Российский университет дружбы народов
- [<https://negoday7484.github.io/>]
- <https://github.com/NEGODAY7484>



## Вводная часть

---

## 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.

## Задание

---

1. Создать простейший калькулятор в UNIX;



## Выполнение лабораторной работы

---

1. Создаю папку lab\_prog. (рис. (fig:001?))

A terminal window with a black background and white text. The prompt is [aatanesov@fedora os]\$ and the command entered is mkdir lab\_prog.

```
[aatanesov@fedora os]$ mkdir lab_prog
```

Рис. 1: Использую команду mkdir

2. Создаю файлы calculate.c calculate.h main.c. (рис. (fig:002?))

```
[aatanesov@fedora lab_prog]$ touch calculate.h calculate.c main.c
```

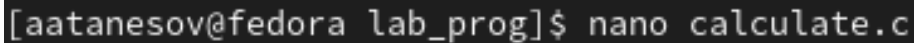
Рис. 2: Использую команду touch

3. Делаем эти файлы исполняемыми. (рис. (fig:003?))

```
[aatanesov@fedora lab_prog]$ chmod +x calculate.h calculate.c main.c
```

Рис. 3: Используя команду chmod

4. Открываю файл calculate.c через nano. (рис. (fig:004?))

A terminal window with a black background and white text. The prompt is [aatanesov@fedora lab\_prog]\$ and the command entered is nano calculate.c.

```
[aatanesov@fedora lab_prog]$ nano calculate.c
```

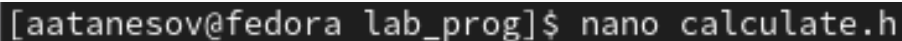
Рис. 4: Использую команду nano

## 5. Пишу код для будущего калькулятора. (рис. (fig:005?))

```
GNU nano 6.4 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("Мультипликатор: ");
        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));
}
```

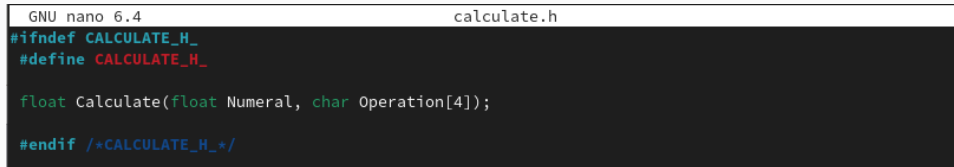
6. Открываю файл calculate.h через nano. (рис. (fig:006?))

A terminal window with a dark background and light gray text. The prompt is [aatanesov@fedora lab\_prog]\$ and the command entered is nano calculate.h.

```
[aatanesov@fedora lab_prog]$ nano calculate.h
```

Рис. 6: Использую команду nano

7. Пишу необходимый код для calculate.h . (рис. (fig:007?))



```
GNU nano 6.4                                calculate.h
#ifndef CALCULATE_H_
#define CALCULATE_H_

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

#endif /*CALCULATE_H_*/
```

Рис. 7: Использую редактор nano



8. Открываю файл main.c через nano. (рис. (fig:008?))

A terminal window with a black background and white text. The prompt is [aatanesov@fedora lab\_prog]\$ and the command entered is nano main.c.

```
[aatanesov@fedora lab_prog]$ nano main.c
```

Рис. 8: Использую команду nano

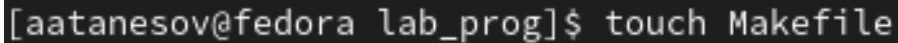
9. Пишу необходимый код, представленный в выполнение лабораторной работы №13 .  
(рис. (fig:009?))



```
GNU nano 6.4                                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("%6.2f\n",Result);
    return 0;
}
```

Рис. 9: Используя редактор nano

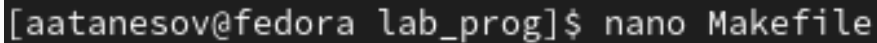
10. Создаю файл Makefile. (рис. (fig:010?))



```
[aatanesov@fedora lab_prog]$ touch Makefile
```

Рис. 10: Использую команду chmod

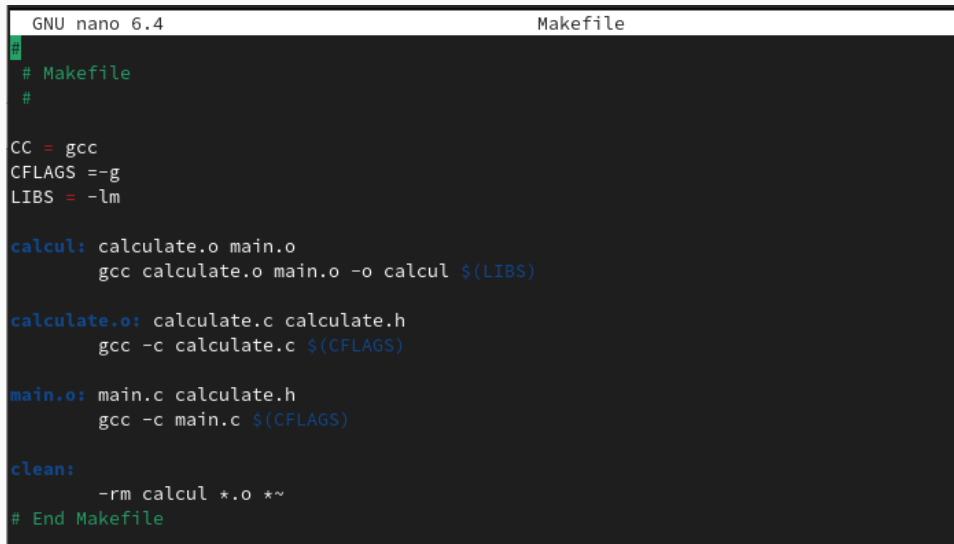
11. Открываю файл Makefile через nano. (рис. (fig:011?))

A terminal window with a dark background. The prompt is [aatanesov@fedora lab\_prog]\$ and the command nano Makefile has been entered.

```
[aatanesov@fedora lab_prog]$ nano Makefile
```

Рис. 11: Использую команду nano

12. Пишу необходимый код. (рис. (fig:012?))

A screenshot of the GNU nano 6.4 text editor interface. The title bar at the top shows "GNU nano 6.4" on the left and "Makefile" on the right. The editor area has a dark background with light green text. The content is a Makefile with the following lines: a comment "# Makefile", compiler settings "CC = gcc", "CFLAGS = -g", and "LIBS = -lm", three target rules for "calcul:", "calculate.o:", and "main.o:", a "clean:" rule with "-rm calcul \*.o \*~", and a final comment "# End Makefile".

```
GNU nano 6.4                                     Makefile
#
# 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. 12: Используя редактор nano

### 13. Запускаю файл калькулятор. (рис. (fig:013?))

```
[aatanesov@fedora lab_prog]$ gdb ./calcul
GNU gdb (GDB) Fedora Linux 13.1-3.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...
(gdb) run
Starting program: /home/aatanesov/work/os/lab_prog/calcul

This GDB supports auto-downloading debuginfo from the following URLs:
    <https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) n
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 5
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): -
Вычитаемое: 2
    3.00
[Inferior 1 (process 5460) exited normally]
Missing separate debuginfos, use: dnf debuginfo-install glibc-2.36-9.fc37.x86_64
```

14. Вывожу 10 строк кода . (рис. (fig:014?))

```
(gdb) list
1      #include <stdio.h>
2      #include "calculate.h"
3      int
4      main (void)
5      {
6      float Numeral;
7      char Operation[4];
8      float Result;
9      printf("Число: ");
10     scanf("%f",&Numeral);
```

(image

width=90%}

15. Вывожу с 12 по 15 строку кода нашей программы. (рис. (fig:015?))

```
(gdb) list 12,15
12      scanf("%s",&Operation);
13      Result = calculate(Numeral, Operation);
14      printf("%6.2f\n",Result);
15      return 0;
(gdb)
```

Рис. 14: Используя команду list 12,15



16. Вывожу с 20 по 29 строку кода программы. (рис. (fig:016?))

```
(gdb) list calculate.c:20,29
20     return(Numeral - SecondNumerал);
21 }
22 else if(strncmp(Operation, "*", 1) == 0)
23 {
24     printf("Множитель: ");
25     scanf("%f",&SecondNumerал);
26     return(Numerал * SecondNumerал);
27 }
28 else if(strncmp(Operation, "/", 1) == 0)
29 {
```

(image/18.png){#

width=90%}

17. Вывожу с 20 по 27 и ставлю точку остановки . (рис. (fig:017?))

```
(gdb) list calculate.c:20,27
20     return(Numeral - SecondNumeral);
21     }
22     else if(strncmp(Operation, "*", 1) == 0)
23     {
24     printf("Множитель: ");
25     scanf("%f",&SecondNumeral);
26     return(Numeral * SecondNumeral);
27     }
(gdb) break 21
Breakpoint 1 at 0x401247: file calculate.c, line 22.
(gdb) nfo breakpoints
Undefined command: "nfo". Try "help".
(gdb) info breakpoints
Num      Type           Disp Enb Address                  What
1        breakpoint     keep y   0x000000000000401247 in Calculate at calculate.c:22
(gdb)
```

Рис. 15: Использую команду break

## 18. Вывожу автоматический анализ кода программ main.c и calculate.c . (рис. (fig:018?))

```
[aatanesov@fedora lab_prog]$ splint calculate.c main.c
Splint 3.1.2 --- 23 Jul 2022

calculate.h:4:38: 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:7:32: Function parameter Operation declared as manifest array (size
constant is meaningless)
calculate.c: (in function Calculate)
calculate.c:13:2: 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:19:2: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:25:2: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:31:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:32:5: 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:35:8: Return value type double does not match declared type float:
(HUGE_VAL)
To allow all numeric types to match, use +relaxtypes.
calculate.c:43:2: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:44:8: Return value type double does not match declared type float:
(pow(Numeral, SecondNumeral))
calculate.c:47:8: Return value type double does not match declared type float:
(sqrt(Numeral))
calculate.c:49:8: Return value type double does not match declared type float:
(sin(Numeral))
calculate.c:51:8: Return value type double does not match declared type float:
(cos(Numeral))
calculate.c:53:8: Return value type double does not match declared type float:
(tan(Numeral))
calculate.c:57:8: Return value type double does not match declared type float:
(HUGE_VAL)
main.c: (in function main)
main.c:10:2: Return value (type int) ignored: scanf("%f", &Num...
main.c:12:13: 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:12:10: Corresponding format code
main.c:12:2: Return value (type int) ignored: scanf("%s", &Ope...

Finished checking --- 18 code warnings
```

## Выводы

---

- Я научился создавать простые приложения и открывать их через терминал .

## Пояснение содержимого файла Makefile

---

## Пояснение содержимого файла Makefile

- Данный Makefile описывает процесс сборки программы “calcul” из исходных файлов “calculate.c” и “main.c”, а также заголовочного файла “calculate.h”.

Переменные CC, CFLAGS и LIBS содержат информацию о компиляторе, флагах компиляции и необходимых для линковки библиотеках соответственно.

Цель “calcul” (строка 9) зависит от объектных файлов “calculate.o” и “main.o”, и собирается командой “gcc calculate.o main.o -o calcul \$(LIBS)”.

Цели “calculate.o” и “main.o” (строки 12-16) компилируют соответствующие исходные файлы в объектные файлы.

Цель “clean” (строка 18) удаляет собранные объектные файлы и исполняемый файл.

Комментарии на каждой строке поясняют назначение каждой переменной или команды.

## Ответы на контрольные вопросы

---



## Ответы на контрольные вопросы

- 1. Для получения информации о возможностях программ gcc, make, gdb и др. можно обратиться к их официальной документации, доступной в сети Интернет, а также использовать команду `man` в терминале UNIX.
- 2. Основными этапами разработки приложений в UNIX являются: проектирование, написание исходного кода, компиляция, отладка, тестирование, установка и настройка приложения.
- 3. Суффикс в контексте языка программирования - это часть названия файла, указывающая на его тип и формат. Например, файл со суффиксом `“.c”` обозначает исходный код на языке C, а файл со суффиксом `“.o”` - скомпилированный объектный файл.
- 4. Основное назначение компилятора языка C в UNIX - это компиляция исходного кода на этом языке в машинный код, который может быть выполнен на компьютере.
- 5. Утилита `make` предназначена для автоматизации процесса сборки приложения из исходного кода и скомпилированных объектных файлов.
- 6. Пример структуры Makefile: