

---

## Front matter

title: "Отчёт по лабораторной работе №13" subtitle: "Средства, применяемые при разработке программного обеспечения в ОС типа UNIX/Linux" author: "Абу Сувейлим Мухаммед Мунифович"

## Generic options

lang: ru-RU toc-title: "Содержание"

## Bibliography

bibliography: bib/cite.bib csl: pandoc/csl/gost-r-7-0-5-2008-numeric.csl

## Pdf output format

toc: true # Table of contents toc-depth: 2 lof: true # List of figures lot: true # List of tables fontsize: 12pt  
linestretch: 1.5 papersize: a4 documentclass: scrreprt

## l18n polyglossia

polyglossia-lang: name: russian options: - spelling=modern - babelshorthands=true polyglossia-otherlangs:  
name: english

## l18n babel

babel-lang: russian babel-otherlangs: english

## Fonts

mainfont: PT Serif romanfont: PT Serif sansfont: PT Sans monofont: PT Mono mainfontoptions: Ligatures=TeX  
romanfontoptions: Ligatures=TeX sansfontoptions: Ligatures=TeX,Scale=MatchLowercase monofontoptions:  
Scale=MatchLowercase,Scale=0.9

## Biblatex

biblatex: true biblio-style: "gost-numeric" biblatexoptions:

- parenttracker=true
- backend=biber
- hyperref=auto
- language=auto
- autolang=other\*
- citestyle=gost-numeric

## Pandoc-crossref LaTeX customization

figureTitle: "Рис." tableTitle: "Таблица" listingTitle: "Листинг" lofTitle: "Список иллюстраций" lotTitle:  
"Список таблиц" lolTitle: "Листинги"

## Misc options

indent: true header-includes:

- `\usepackage{indentfirst}`
- `\usepackage{float} # keep figures where there are in the text`
- `\floatplacement{figure}{H} # keep figures where there are in the text`

---

## Цель 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.

---

## Задание

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. Реализация функций калькулятора в файле `calculate.h`:

```
////////////////////////////////////
// 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("Множитель: ");
    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);
}
}

```

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

```

////////////////////////////////////
// calculate.h

#ifndef CALCULATE_H_
#define CALCULATE_H_

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

#endif /*CALCULATE_H_*/

```

Основной файл main.c, реализующий интерфейс пользователя к калькулятору:

```
////////////////////////////////////  
// 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("%.2f\n",Result);  
    return 0;  
}
```

3. Выполните компиляцию программы посредством gcc:

```
gcc -c calculate.c  
gcc -c main.c  
gcc calculate.o main.o -o calcul -lm
```

4. При необходимости исправьте синтаксические ошибки.

5. Создайте Makefile со следующим содержанием:

```
#  
# 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
```

Поясните в отчёте его содержание. 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 кода используйте команду list:

```
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 точках останова:

```
info breakpoints
```

- Запустите программу внутри отладчика и убедитесь, что программа остановится в момент прохождения точки останова:

```
run
5
-
backtrace
```

- Отладчик выдаст следующую информацию:

```
#0 Calculate (Numeral=5, Operation=0x7fffffff280 "-")
at calculate.c:21
#1 0x00000000400b2b in main () at main.c:17
```

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:


```
info breakpoints
delete 1
```

7. С помощью утилиты splint попробуйте проанализировать коды файлов calculate.c и main.c.


## Выполнение задания


- В домашнем каталоге я создал подкаталог ~/work/os/lab\_prog.
- В нём файлы я также создал: 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.


 "Рис.1: lab\_prog"

 "Рис.2: calculate.h calculate.c main.c"

Далее, я скопировал в файле calculate.h calculate.c и main.c в программе, соответственно.

 "Рис.3: calculate.c 1"

 "Рис.4: calculate.c 2"

 "Рис.5: calculate.h"

Я выполнил компиляцию программы посредством gcc:  "Рис.6: gcc"

## Выводы

---

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