

# РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

## Факультет физико-математических и естественных наук

### Кафедра прикладной информатики и теории вероятностей

#### ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 14 «Средства, применяемые при разработке программного обеспечения в ОС типа UNIX/Linux»

дисциплина: *Операционные системы*

Студент: Саргсян Арам Грачьевич

Группа: НПИбд 02-20

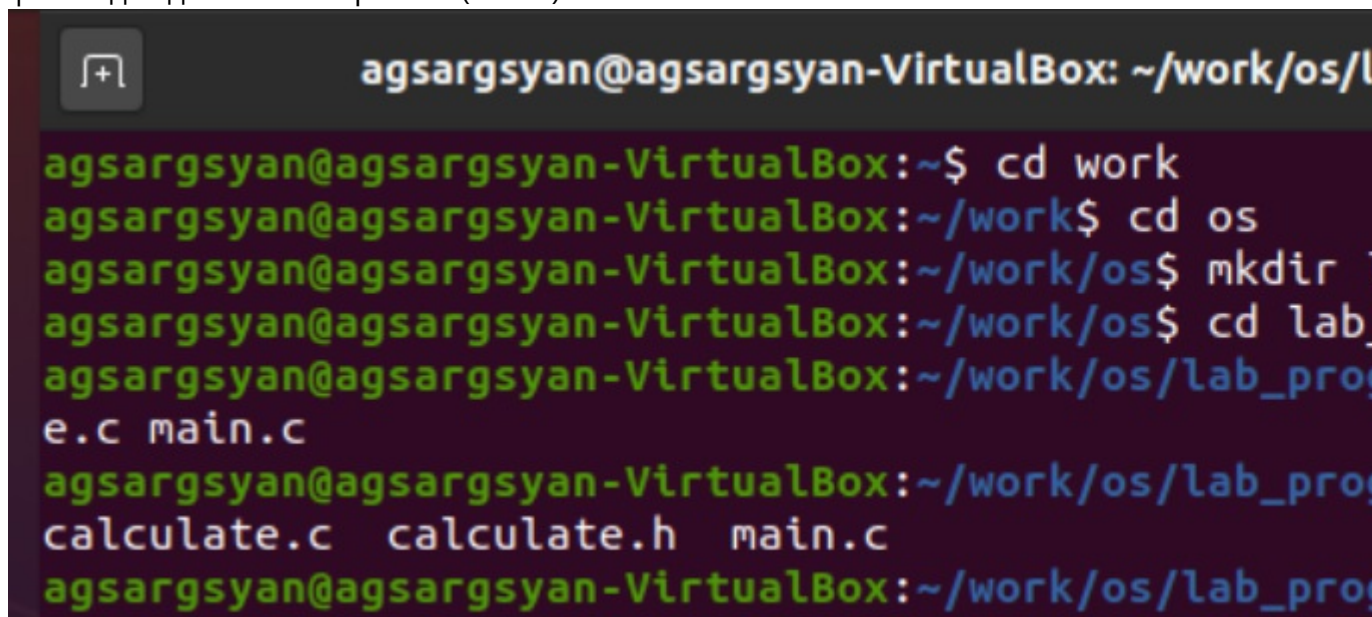
МОСКВА 2021 г.

---

**Цель работы:** Приобрести простейшие навыки разработки, анализа, тестирования и отладки приложений в ОС типа UNIX/Linux на примере создания на языке С калькулятора с простейшими функциями.

#### Ход работы

1. В домашнем каталоге создали подкаталог `~/work/os/lab_prog`. Создал в нем необходимые файлы для дальнейшей работы(Рис. 1)



```
agsargsyan@agsargsyan-VirtualBox: ~/work/os/l
agsargsyan@agsargsyan-VirtualBox:~$ cd work
agsargsyan@agsargsyan-VirtualBox:~/work$ cd os
agsargsyan@agsargsyan-VirtualBox:~/work/os$ mkdir
agsargsyan@agsargsyan-VirtualBox:~/work/os$ cd lab_
agsargsyan@agsargsyan-VirtualBox:~/work/os/lab_prog
e.c main.c
agsargsyan@agsargsyan-VirtualBox:~/work/os/lab_prog
calculate.c calculate.h main.c
agsargsyan@agsargsyan-VirtualBox:~/work/os/lab_prog
```

рис. 1

2. Реализовал в этих файлах программы из текста.(Рис. 2-4)

```

//////////////////////////////////// calculate.c
#include<stdio.h>
#include<math.h>
#include<string.h>
#include"calculate.h"
float
calculate(floatNumeral,charOperation[4])
{
    floatSecondNumeral;
    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);
    }
}

```

рис. 2

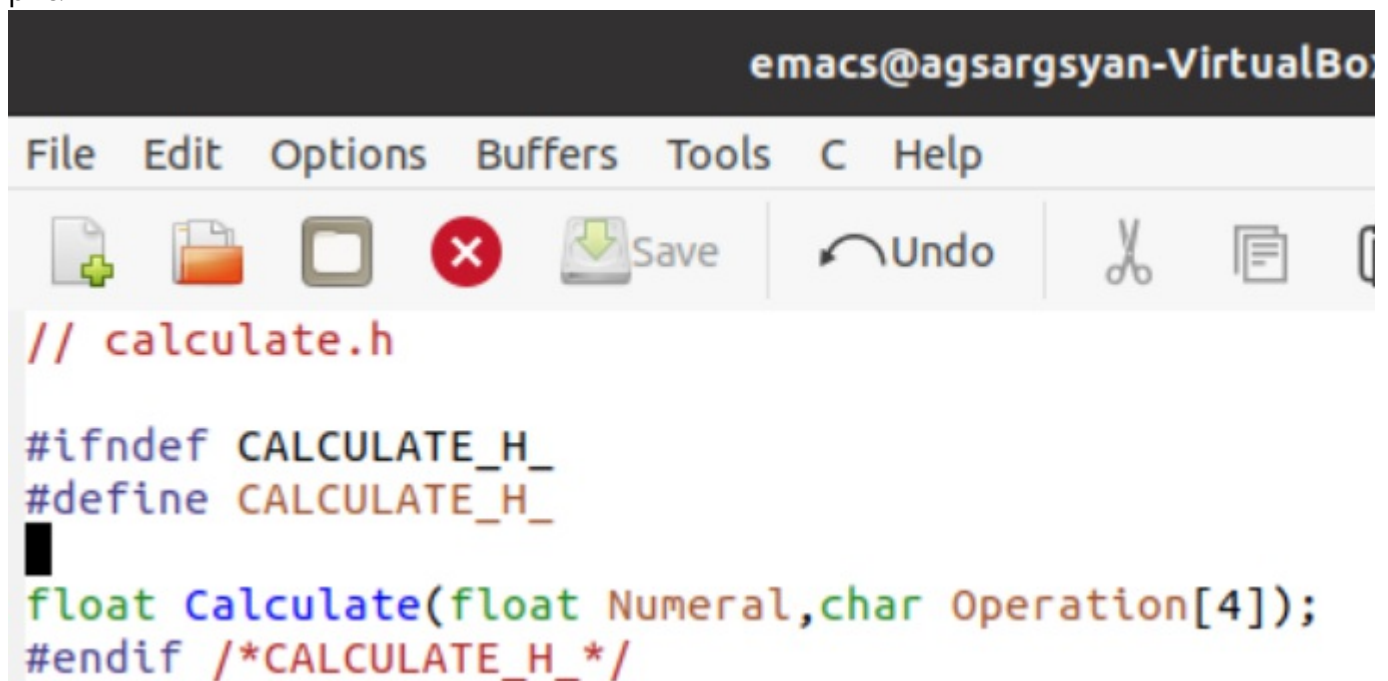


рис. 3

```
// main.c
#include<stdio.h>
#include"calculate.h"
int main(void)
{
    float Numeral;
    charOperation[4];
    floatResult;
    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. 4

3. Выполним компиляцию программы посредством gcc. После исправим ошибки. (Рис.5-7)

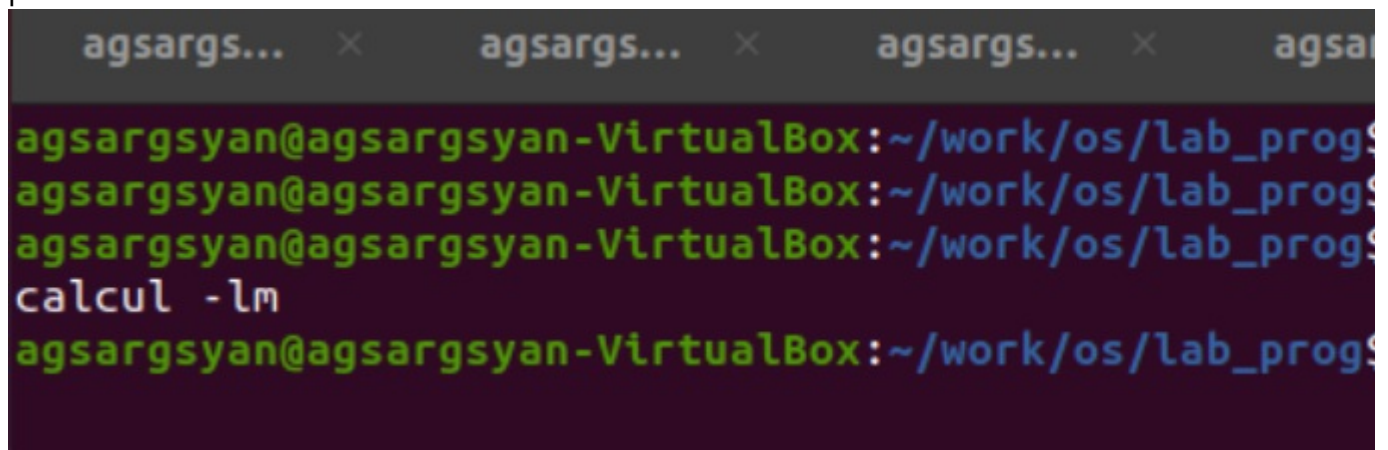
```
agsargsyan@agsargsyan-VirtualBox:~/work/os/lab_prog$
agsargsyan@agsargsyan-VirtualBox:~/work/os/lab_prog$
main.c: In function 'main':
main.c:13:11: warning: format '%s' expects argument
ent 2 has type 'char (*)[4]' [-Wformat=]
    13 |     scanf("%s",&Operation);
        |           ~^  ~~~~~
        |           |  |
        |           |  char (*)[4]
        |           char *
```

рис. 5



```
// 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. 6



```
agsargsyan@agsargsyan-VirtualBox:~/work/os/lab_prog$
agsargsyan@agsargsyan-VirtualBox:~/work/os/lab_prog$
agsargsyan@agsargsyan-VirtualBox:~/work/os/lab_prog$
calcul -lm
agsargsyan@agsargsyan-VirtualBox:~/work/os/lab_prog$
```

рис. 7

4. Создадим Makefile. (Рис. 8)

```
#
# 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)

clean:
        -rm calcul *.o *~

# End Makefile
```

рис. 8

5. С помощью gdb выполним отладку программы calcul. Запустим программу внутри отладки и выполним несколько действий(Рис. 9-13)

```
agsargsyan@agsargsyan-VirtualBox:~/work/os/lab_prog
agsargsyan@agsargsyan-VirtualBox:~/work/os/lab_prog
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://
This is free software: you are free to change and r
There is NO WARRANTY, to the extent permitted by la
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 resourc
<http://www.gnu.org/software/gdb/documentation/

For help, type "help".
Type "apropos word" to search for commands related
Reading symbols from ./calcul...
(No debugging symbols found in ./calcul)
(gdb)
```

рис. 9

```
(gdb) run
Starting program: /home/agsargsyan/work/os/lab_prog/
Число: 5
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): +
Второе слагаемое: 8
13.00
[Inferior 1 (process 2913) exited normally]
(gdb) █
```

рис. 10



```

(gdb) b 15
Note: breakpoint 1 also set at pc 0x7ffff7c9bf20.
Breakpoint 2 at 0x7ffff7c9bf20: file init-first.c,
(gdb) info b

```

Num	Type	Disp	Enb	Address
1	breakpoint	keep	y	0x00007ffff7c9bf20
2	breakpoint	keep	y	0x00007ffff7c9bf20

```

(gdb)

```

рис. 11

```

agsargsyan@agsargsyan-VirtualBox:~/work/os/lab_prog
Splint 3.1.2 --- 20 Feb 2018

calculate.h:6:36: Function parameter Operation decl
                    constant is meaningless)
    A formal parameter is declared as an array with s
    is ignored in this context, since the array forma
    pointer. (Use -fixedformalarray to inhibit warnin
calculate.c:7:31: Function parameter Operation decl
                    constant is meaningless)
calculate.c: (in function Calculate)
calculate.c:13:7: Return value (type int) ignored:
    Result returned by function call is not used. If
    result to (void) to eliminate message. (Use -retv
calculate.c:19:7: Return value (type int) ignored:
calculate.c:25:7: Return value (type int) ignored:
calculate.c:31:7: Return value (type int) ignored:
calculate.c:32:10: Dangerous equality comparison in
                    SecondNumeral == 0
    Two real (float, double, or long double) values a

```

рис. 12

```
agsargsyan@agsargsyan-VirtualBox:~/work/os/lab_prog:
Splint 3.1.2 --- 20 Feb 2018

calculate.h:6:36: Function parameter Operation declared as constant
                    constant is meaningless)
    A formal parameter is declared as an array with size, but size
    is ignored in this context, since the array format is a char
    pointer. (Use -fixedformalarray to inhibit warning)
main.c: (in function main)
main.c:11:3: Return value (type int) ignored: scanf
    Result returned by function call is not used. If you want to
    result to (void) to eliminate message. (Use -retvalue to
main.c:13:3: Return value (type int) ignored: scanf
Finished checking --- 3 code warnings
```

рис. 13

## ВЫВОД:

В ходе работы 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.

## КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Воспользоваться интернетом, воспользоваться командой man, info
2. Unix поддерживает следующие основные этапы разработки приложений:

-создание исходного кода программы;

-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;

-сохранение всех изменений, выполняемых при тестировании и отладке.



3. Использование суффикса ".c" для имени файла с программой на языке Си отражает удобное и полезное соглашение, принятое в ОС UNIX. Для любого имени входного файла суффикс определяет какая компиляция требуется. Суффиксы и префиксы указывают тип объекта. Одно из полезных свойств компилятора Си — его способность по суффиксам определять типы файлов. По суффиксу .c компилятор распознает, что файл abcd.c должен компилироваться, а по суффиксу .o, что файл abcd.o является объектным модулем и для получения исполняемой программы необходимо выполнить редактирование связей. Простейший пример командной строки для компиляции программы abcd.c и построения исполняемого модуля abcd имеет вид: gcc -o abcd abcd.c.
4. В компиляции всей программы в целом и получении исполняемого модуля.
5. Для упрощения и автоматизации работы пользователя с командной строкой
6. Текст, следующий за точкой с запятой, и все последующие строки, начинающиеся с литеры табуляции, являются командами ОС UNIX, которые необходимо выполнить для обновления целевого файла. Таким образом, спецификация взаимосвязей имеет формат:

target1 [ target2...]: [:] [dependment1...]

[(tab)commands]

[#commentary]

[(tab)commands]

[#commentary],

где # — специфицирует начало комментария; : — последовательность команд ОС UNIX должна содержаться в одной строке make-файла (файла описаний), есть возможность переноса команд (), но она считается как одна строка; :: — последовательность команд ОС UNIX может содержаться в нескольких последовательных строках файла описаний.

7. Все программы отладки позволяют отслеживать состояние программы на любом из этапов ее исполнения. Для того чтобы эту возможность использовать необходимо изучить документацию по использованию определенного отладчика. Понять общие принципы отладки.
8. – backtrace – выводит весь путь к текущей точке останова, то есть

названия всех функций, начиная от main(); иными словами, выводит

весь стек функций;

– break – устанавливает точку останова; параметром может быть

номер строки или название функции;

– clear – удаляет все точки останова на текущем уровне стека (то есть в текущей функции);

– continue – продолжает выполнение программы от текущей точки до конца;

– delete – удаляет точку останова или контрольное выражение;

– display – добавляет выражение в список выражений, значения которых отображаются каждый раз при остановке программы;

– finish – выполняет программу до выхода из текущей функции; отображает возвращаемое значение, если такое имеется;

– info breakpoints – выводит список всех имеющихся точек останова;

– info watchpoints – выводит список всех имеющихся контрольных выражений;

– list – выводит исходный код; в качестве параметра передаются название файла исходного кода, затем, через двоеточие, номер начальной и конечной строки;

– next – пошаговое выполнение программы, но, в отличие от команды step, не выполняет пошагово вызываемые функции;

– print – выводит значение какого-либо выражения (выражение передается в качестве параметра);

– run – запускает программу на выполнение;

– set – устанавливает новое значение переменной

– step – пошаговое выполнение программы;

– watch – устанавливает контрольное выражение, программа остановится, как только значение контрольного выражения изменится;

9. Сначала запустил отладчик для программы. Установил интересующую меня точку останова. Запустил программу ожидая, что программа остановится на точке останова. Узнал необходимые данные моей программы на текущем этапе ее исполнения путем ввода команд. Отобразил данные. Завершил программу, снял точки останова.

10. В моем случае не было синтаксических ошибок. Были ошибки семантические. Компилятор начал жаловаться на то, что программа по смыслу принимает указатель на char массив. В то время как я вводил не указатель, а прямое значение . Ошибка была исправлена путем удаления &.

11.

- cscope - исследование функций, содержащихся в программе;
- lint -- критическая проверка программ, написанных на языке Си.

12. Splint- инструмент для статической проверки С-программ на наличие уязвимостей и ошибок