

# Презентация лабораторной работы 13

**ТЕМА** «Средства, применяемые при разработке программного обеспечения в ОС типа UNIX/Linux»

## Выполнил/ла:

---

**Студент/ка группы** НПИбд-02-21

**Студенческий билет No** 1032205421

**Студент/ка:** Стелина Петрити

## 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.

## Последовательность выполнения работы

*1. В домашнем каталоге создайте подкаталог `~/work/os/lab_prog`*

```

[inna@fedora ~]$ mkdir work/os/lab_prog
[inna@fedora ~]$ cd
[inna@fedora ~]$ ls
10.tmp                hello.cpp
123.sh                home
123.tar               '###lab07.sh###'
1.tmp                 '##lab07.sh##'
2.tmp                 '##lab07.sh##~'
3.tmp                 '#lab07.sh#'
456.sh                lab07.sh
4.tmp                 lab10.sh
5.tmp                 lab10.sh~
6.tmp                 lab10.tar
7.tmp                 lab4.md
8.tmp                 lab4.pdf
9.tmp                 letters
abc1.sh               lock.file
abc1.tar               may
archive               memos
backup                misk
conf.txt              monthly
Desktop               Music
dir1                  pandoc-crossref-Linux.tar.xz
'dmesg | grep -i "What are we looking"'
Documents              parentdir
Downloads              Pictures
exc3.sh                play
exc4.sh                point3
exp1.tar               Public
expl.txt               reports
exp2.txt               ski.plases
                       Templates

```

### 1.1. 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.

```

[inna@fedora ~]$ cd work/os/lab_prog
[inna@fedora lab_prog]$ touch calculate.h calculate.c main.c
[inna@fedora lab_prog]$ ls
calculate.c  calculate.h  main.c
[inna@fedora lab_prog]$

```

### 2.1. Создание файлы- *calculate.h*, *calculate.c*, *main.c*.

```

[inna@fedora lab_prog]$ emacs calculate.c
[inna@fedora lab_prog]$ emacs calculate.h
[inna@fedora lab_prog]$ emacs main.c
[inna@fedora lab_prog]$

```

### 2.2. emacs: *calculate.h*, *calculate.c*, *main.c*.

```

//////////
// 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);
}
}

```

U:--- **calculate.c**    Top L10    (C/\*l Abbrev)

### 2.3. функция калькулятора calculate.h

```

//////////
// calculate.h
#ifndef CALCULATE_H_
#define CALCULATE_H_

float Calculate(float Numeral, char Operation[4]);
#endif /*CALCULATE_H_*/

```

### 2.4. Интерфейсный файл calculate.h

```

////////////////////
// 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;
}

```

U:\*\*\*- **main.c** All L11 (C/\*l Abbrev)

2.5. файл main

3. Выполните компиляцию программы посредством gcc:

```

[inna@fedora lab_prog]$ gcc -c calculate.c
[inna@fedora lab_prog]$ gcc -c main.c

[inna@fedora lab_prog]$ gcc calculate.o main.o -o calcul -lm
[inna@fedora lab_prog]$

```

3.1. Выполнение компиляцию программы

4. Создайте Makefile со следующим содержанием: В этом файле мы создаем переменные CC, CFLAGS, LIBS. Инициализируем и создаем блоки.

```

# 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

```

Makefile All L1 (GNUmakefile)

#### 4.1. Makefile

6. С помощью *gdb* выполните отладку программы *calcul* (перед использованием *gdb* исправьте *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 *~

```

#### 6.1. отладка программы *calcul*

- Запустим отладчик *GDB*, загрузив в него программу для отладки (рис. [-@fig:008]).

```
[inna@fedora lab_prog]$ gdb ./calcul
```

GNU gdb (GDB) Fedora 10.2-9.fc35

Copyright (C) 2021 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:

### 6.2. Запустим отладчик GDB

- Для просмотра строк с 12 по 15 основного файла используйте `list` с параметрами

```
(gdb) list 12,15
12      printf("Число: ");
13      scanf("%f",&Numeral);
14      printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");
15      scanf("%s",&Operation);
```

### 6.3. строк с 12 по 15

- Установите точку останова в файле `calculate.c` на строке номер 21

```
list calculate.c:20,27
    printf("Вычитаемое: ");
    scanf("%f",&SecondNumeral);
    return(Numeral - SecondNumeral);
}
else if(strncmp(Operation, "*", 1) == 0)
{
    printf("Множитель: ");
    scanf("%f",&SecondNumeral);
break 21
point 1 at 0x40121e: file calculate.c, line 21.
```

### 6.4. просмотра строк номер 21

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

и `main.c`: `splint calculate.c`

```
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:9:37: Function parameter Operation declared as manifest array (size
constant is meaningless)
```

### 7.1. `splint calculate.c`

```
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:13:3: Return value (type int) ignored: scanf("%f", &Num...
```

*7.2.splint main.c*

## Выводы

Во время этой лабораторной работы я работала с самыми простыми навыками. Эти простейшие навыки включают в себя навыки разработки, анализа, тестирования и отладки приложений. Я использовала в качестве примера создание калькулятора с простейшими функциями.