

Лабораторная работа No 14

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

Кеан Путхеаро НПИбд-01-20

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	8
4	Вывод	13
5	Библиография	14

Список иллюстраций

3.1	каталог	8
3.2	файлы	8
3.3	calculate.h файл	8
3.4	calculate.c файл	9
3.5	main.c файл	9
3.6	компиляция gcc	10
3.7	Makefile	10
3.8	gdb ./calcul	11
3.9	gdb run	11
3.10	splint calculate.c	12
3.11	splint main.c	12

1 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.

2 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)); }
```

```

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

```

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

```

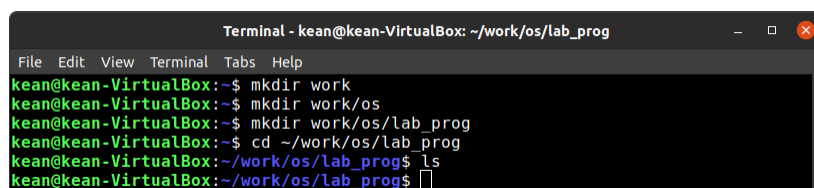
Поясните в отчёте его содержание.

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 код 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 точка 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=0x7ffffffd280 "-") at calculate.c:21 #1 0x0000000000400b2b 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`.

3 Выполнение лабораторной работы

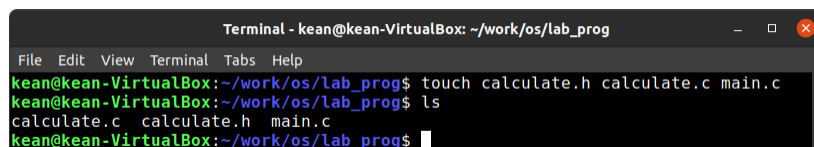
1. В домашнем каталоге создал подкаталог ~/work/os/lab_prog.



```
Terminal - kean@kean-VirtualBox: ~/work/os/lab_prog
File Edit View Terminal Tabs Help
kean@kean-VirtualBox:~$ mkdir work
kean@kean-VirtualBox:~$ mkdir work/os
kean@kean-VirtualBox:~$ mkdir work/os/lab_prog
kean@kean-VirtualBox:~$ cd ~/work/os/lab_prog
kean@kean-VirtualBox:~/work/os/lab_prog$ ls
kean@kean-VirtualBox:~/work/os/lab_prog$
```

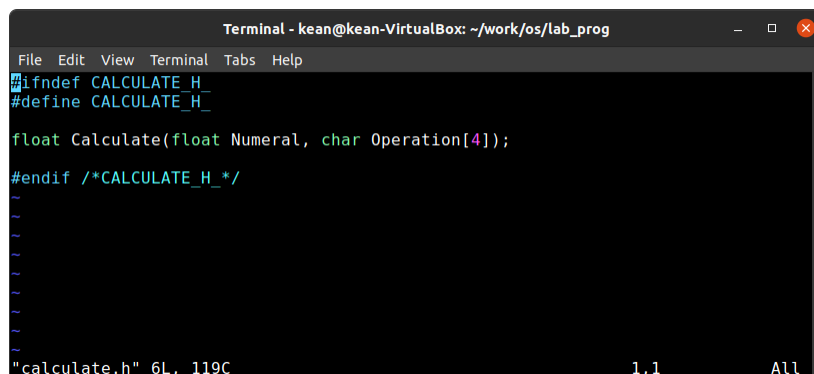
Рис. 3.1: каталог

2. Создал в нём файлы: calculate.h, calculate.c, main.c.



```
Terminal - kean@kean-VirtualBox: ~/work/os/lab_prog
File Edit View Terminal Tabs Help
kean@kean-VirtualBox:~/work/os/lab_prog$ touch calculate.h calculate.c main.c
kean@kean-VirtualBox:~/work/os/lab_prog$ ls
calculate.c calculate.h main.c
kean@kean-VirtualBox:~/work/os/lab_prog$
```

Рис. 3.2: файлы



```
Terminal - kean@kean-VirtualBox: ~/work/os/lab_prog
File Edit View Terminal Tabs Help
#ifndef CALCULATE_H
#define CALCULATE_H

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

#endif /*CALCULATE_H*/

"calculate.h" 6L, 119C 1,1 All
```

Рис. 3.3: calculate.h файл

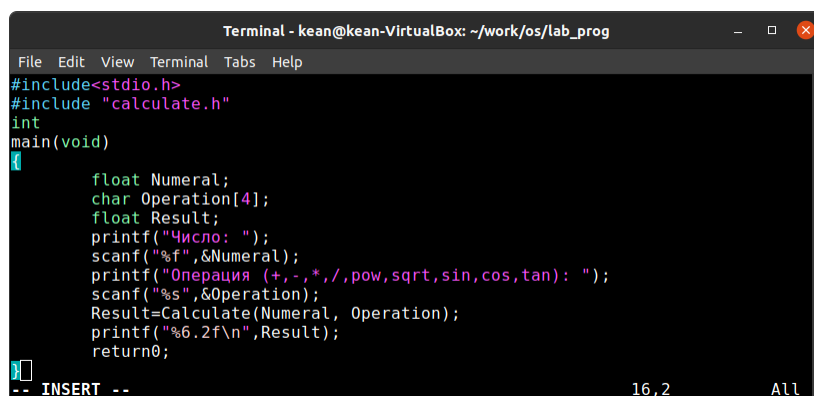


```
Terminal - kean@kean-VirtualBox: ~/work/os/lab_prog
File Edit View Terminal Tabs Help
#include<stdio.h>
#include<math.h>
#include<string.h>
#include"calculate.h"

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));
    }
}

1,8 Top
```

Рис. 3.4: calculate.c файл

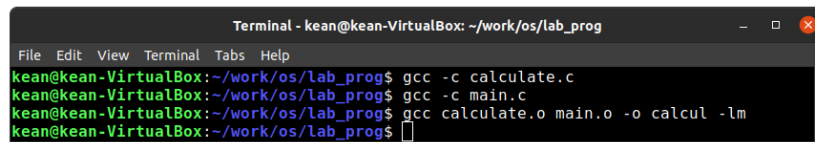


```
Terminal - kean@kean-VirtualBox: ~/work/os/lab_prog
File Edit View Terminal Tabs Help
#include<stdio.h>
#include "calculate.h"
int
main(void)
{
    float Numeral;
    char Operation[4];
    float Result;
    printf("Число: ");
    scanf("%f",&Numeral);
    printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");
    scanf("%s",&Operation);
    Result=Calculate(Numeral, Operation);
    printf("%.2f\n",Result);
    return 0;
}

-- INSERT -- 16,2 All
```

Рис. 3.5: main.c файл

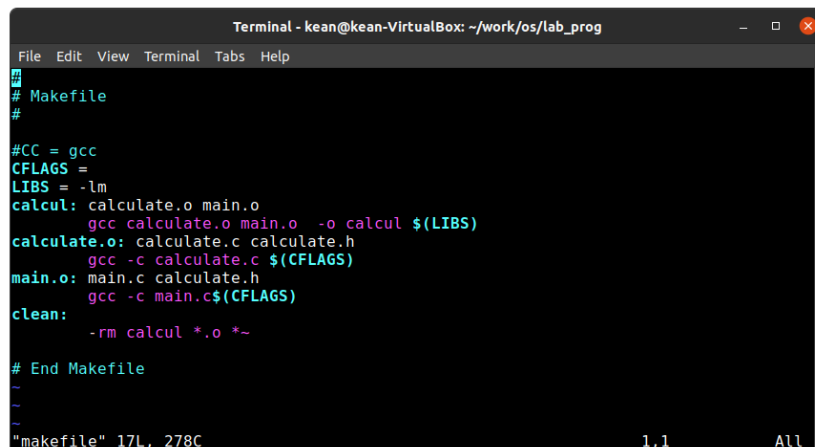
3. Выполнил компиляцию программы посредством gcc



```
Terminal - kean@kean-VirtualBox: ~/work/os/lab_prog
File Edit View Terminal Tabs Help
kean@kean-VirtualBox:~/work/os/lab_prog$ gcc -c calculate.c
kean@kean-VirtualBox:~/work/os/lab_prog$ gcc -c main.c
kean@kean-VirtualBox:~/work/os/lab_prog$ gcc calculate.o main.o -o calcul -lm
kean@kean-VirtualBox:~/work/os/lab_prog$
```

Рис. 3.6: компиляция gcc

4. Исправил синтаксические ошибки в файле main.c (удалил & перед operator в линии scanf("%s",&Operation);)
5. Создал Makefile со следующим содержанием

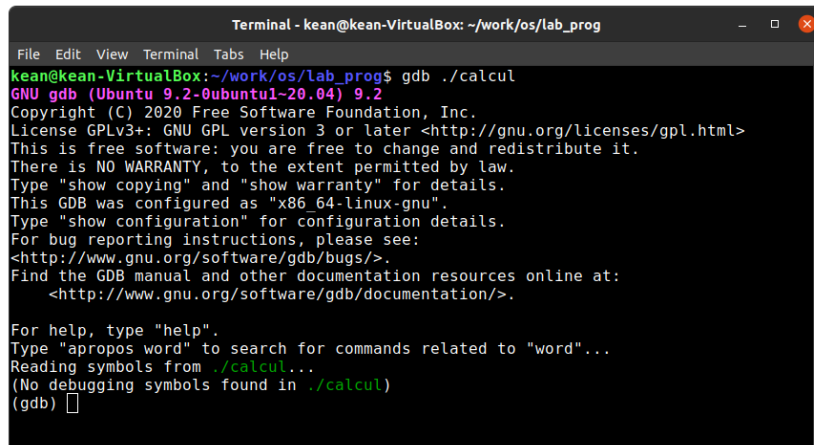


```
Terminal - kean@kean-VirtualBox: ~/work/os/lab_prog
File Edit View Terminal Tabs Help
#
# Makefile
#
#CC = gcc
CFLAGS =
LIBS = -lm
calcul: calculate.o main.o
    gcc calculate.o main.o -o calcul $(LIBS)
calculate.o: calculate.c calculate.h
    gcc -c calculate.c $(CFLAGS)
main.o: main.c calculate.h
    gcc -c main.c $(CFLAGS)
clean:
    -rm calcul *.o *~
# End Makefile
~
~
"makefile" 17L, 278C                               1,1      All
```

Рис. 3.7: Makefile

6. С помощью gdb выполнил отладку программы calcul (перед использованием gdb исправьте Makefile)

Запустите отладчик GDB, загрузив в него программу для отладки: `gdb ./calcul`

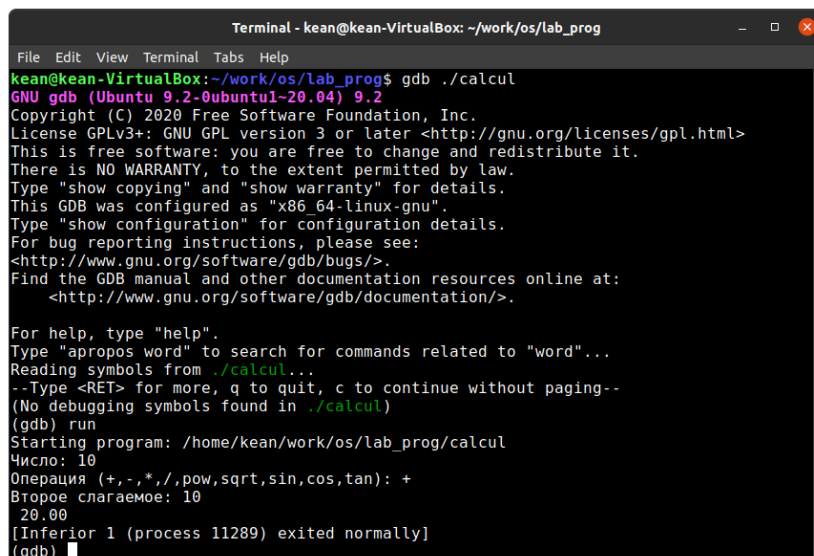


```
Terminal - kean@kean-VirtualBox: ~/work/os/lab_prog
File Edit View Terminal Tabs Help
kean@kean-VirtualBox:~/work/os/lab_prog$ gdb ./calcul
GNU gdb (Ubuntu 9.2-0ubuntu1-20.04) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
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...
(No debugging symbols found in ./calcul)
(gdb) 
```

Рис. 3.8: gdb ./calcul

Для запуска программы внутри отладчика ввел команду run: run



```
Terminal - kean@kean-VirtualBox: ~/work/os/lab_prog
File Edit View Terminal Tabs Help
kean@kean-VirtualBox:~/work/os/lab_prog$ gdb ./calcul
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://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...
--Type <RET> for more, q to quit, c to continue without paging--
(No debugging symbols found in ./calcul)
(gdb) run
Starting program: /home/kean/work/os/lab_prog/calcul
Число: 10
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): +
Второе слагаемое: 10
20.00
[Inferior 1 (process 11289) exited normally]
(gdb) 
```

Рис. 3.9: gdb run

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

```
Terminal - kean@kean-VirtualBox: ~/work/os/lab_prog
File Edit View Terminal Tabs Help
kean@kean-VirtualBox:~/work/os/lab_prog$ splint calculate.c
Splint 3.1.2 --- 20 Feb 2018

calculate.h:4:37: 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:6:36: Function parameter Operation declared as manifest array (size
constant is meaningless)
calculate.c: (in function Calculate)
calculate.c:12:3: 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:18:3: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:24:3: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:30:3: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:31:6: 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:34:10: Return value type double does not match declared type float:
(HUGE_VAL)
To allow all numeric types to match, use +relaxtypes.
calculate.c:42:3: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:43:9: Return value type double does not match declared type float:
(pow(Numeral, SecondNumeral))
calculate.c:46:9: Return value type double does not match declared type float:
```

Рис. 3.10: splint calculate.c

```
Terminal - kean@kean-VirtualBox: ~/work/os/lab_prog
File Edit View Terminal Tabs Help
kean@kean-VirtualBox:~/work/os/lab_prog$ splint main.c
Splint 3.1.2 --- 20 Feb 2018

calculate.h:4:37: 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)
main.c: (in function main)
main.c:10:2: Return value (type int) ignored: scanf("%f", &Num...
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)
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", &Op...

Finished checking --- 4 code warnings
kean@kean-VirtualBox:~/work/os/lab_prog$
```

Рис. 3.11: splint main.c

4 Вывод

В результате работы , я приобрёл простейшие навыки разработки, анализа, тестирования и отладки приложений в Линукс

5 Библиография

1. (Лабораторная работа №14) https://esystem.rudn.ru/pluginfile.php/1142386/mod_resource/content/1/lab_prog.pdf
2. (stackexchange) <https://vi.stackexchange.com/questions/10209/execute-current-buffer-as-bash-script-from-vim>
3. (BASH: функция getopt — используем опции в скриптах) https://esystem.rudn.ru/pluginfile.php/1142386/mod_resource/content/1/lab_shell_prog_2.pdf
4. (stackoverflow) <https://stackoverflow.com/questions/16483119/an-example-of-how-to-use-getopts-in-bash>