

# РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

## ОТЧЕТ

### ПО ЛАБОРАТОРНОЙ РАБОТЕ № 14

дисциплина: Операционные системы

Студент: Мухтарова Камила Айратовна.

Группа: НПИбд-02-20

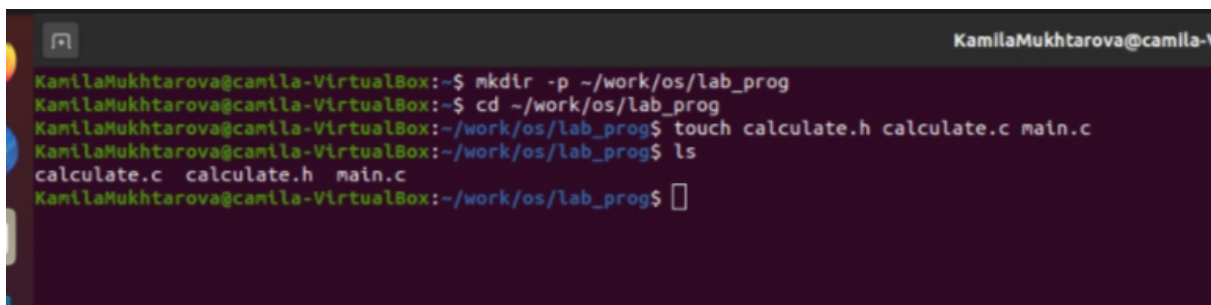
МОСКВА

2021 год

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.

Ход работы:

- В домашнем каталоге создадим подкаталог `~/work/os/lab_prog..` Создадим 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.



```
KamilaMukhtarova@camila-VirtualBox:~$ mkdir -p ~/work/os/lab_prog
KamilaMukhtarova@camila-VirtualBox:~$ cd ~/work/os/lab_prog
KamilaMukhtarova@camila-VirtualBox:~/work/os/lab_prog$ touch calculate.h calculate.c main.c
KamilaMukhtarova@camila-VirtualBox:~/work/os/lab_prog$ ls
calculate.c calculate.h main.c
KamilaMukhtarova@camila-VirtualBox:~/work/os/lab_prog$
```

(рис. 1) - создание

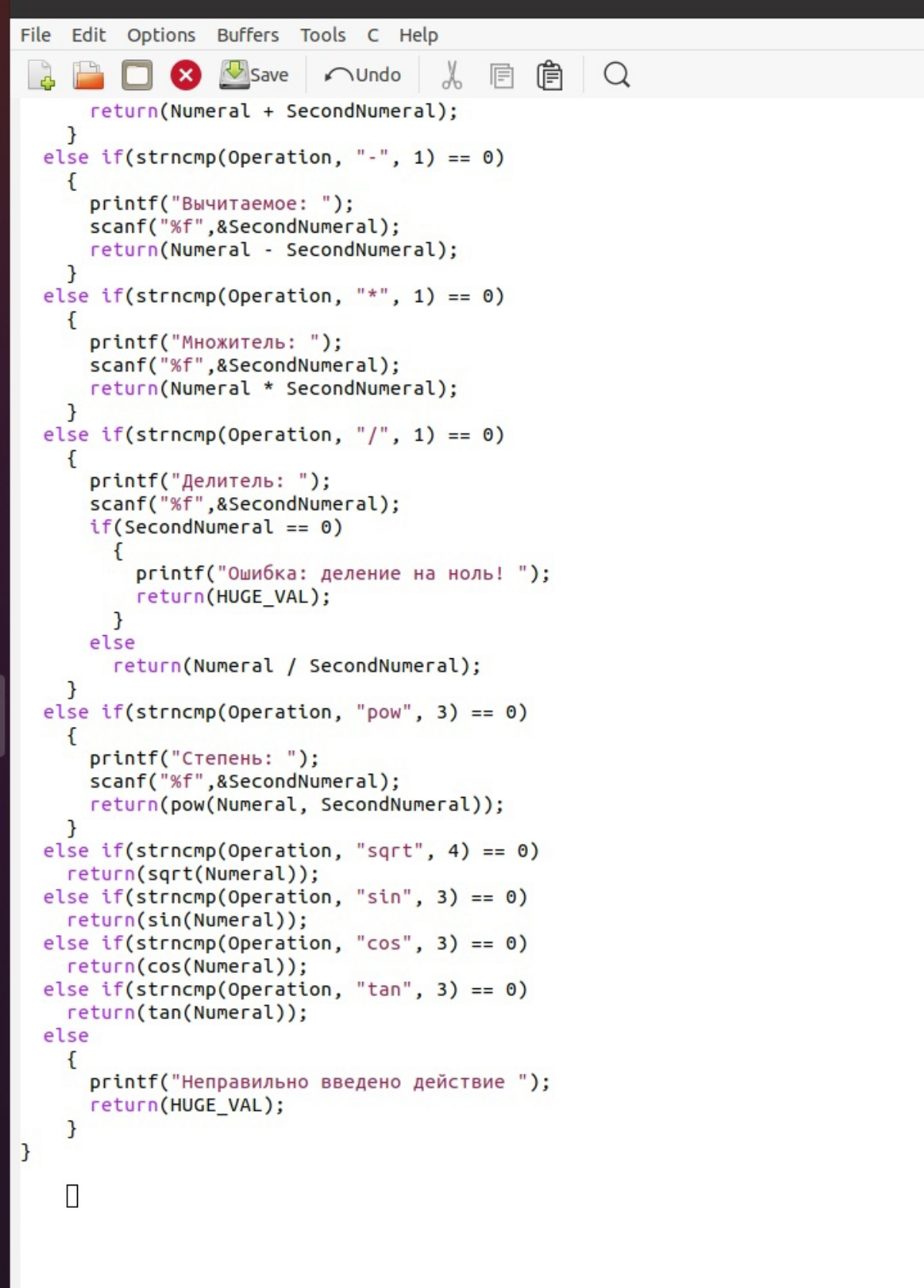
Реализация функций калькулятора в файле `calculate.c`

File Edit Options Buffers Tools C 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));
    else if(strncmp(Operation, "tan", 3) == 0)
        return(tan(Numeral));
    else
    {
        printf("Неправильно введено действие ");
    }
}
```



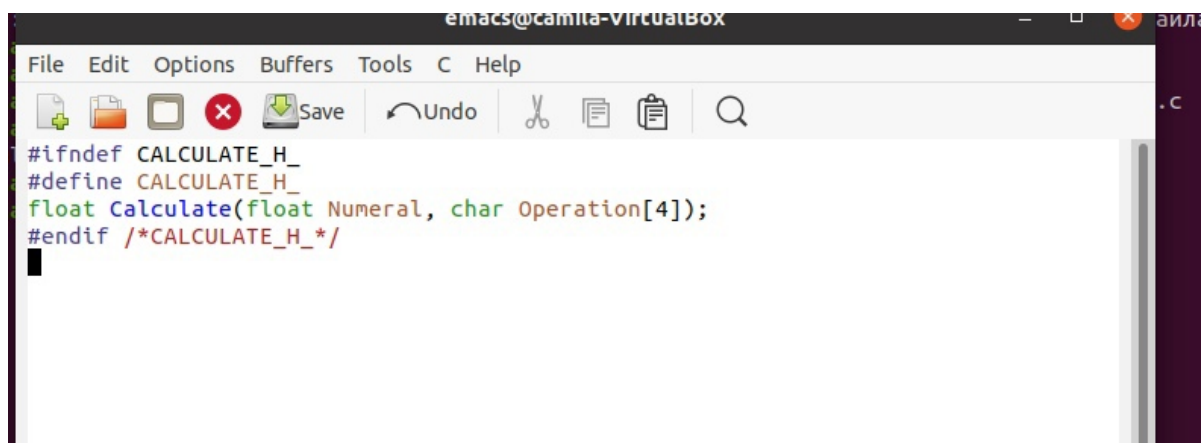
```
File Edit Options Buffers Tools C Help
[Icons: New, Open, Save, Undo, Cut, Copy, Paste, Find]

    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);
}
}

□
```

(рис. 2, 3) - скрипт

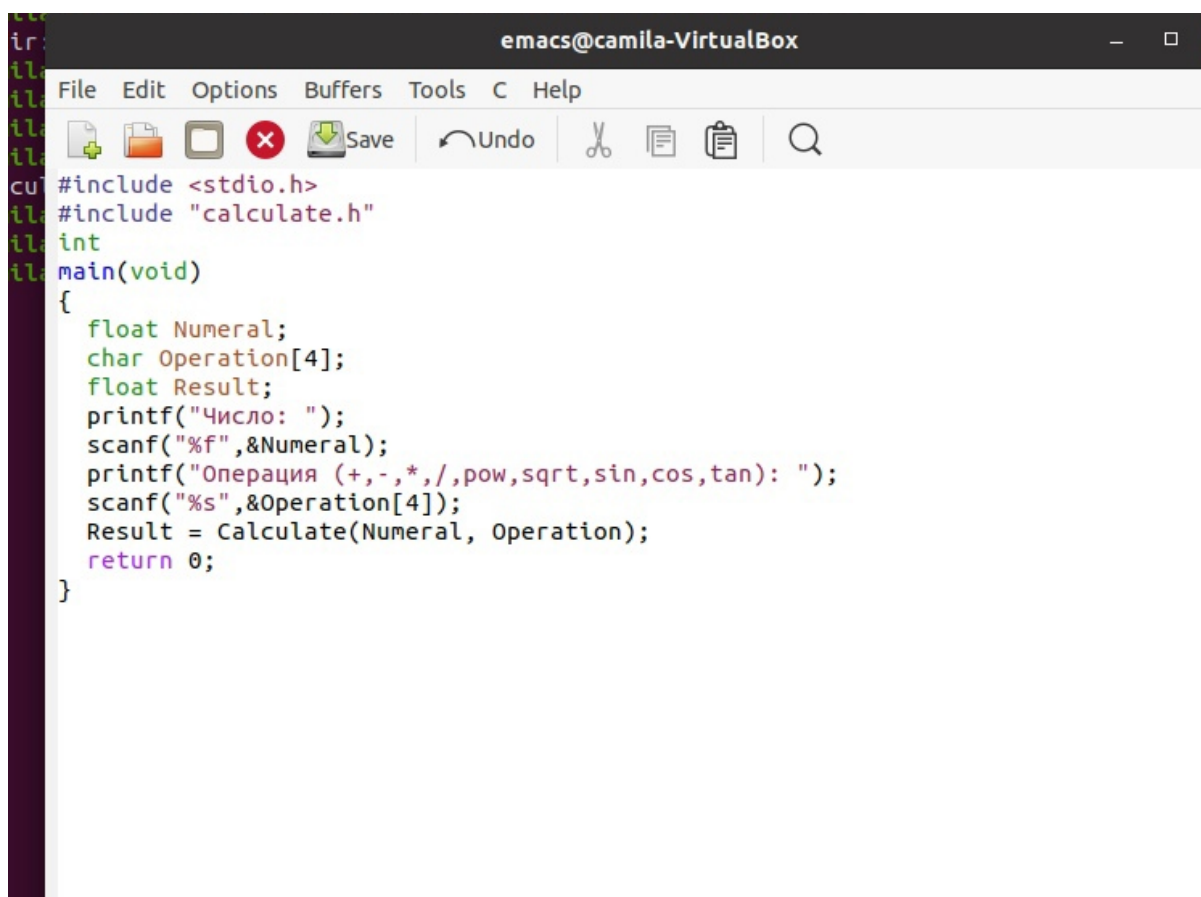
Интерфейсный файл calculate.h, описывающий формат вызова функции калькулятора



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

(рис.4) - скрипт

Основной файл main.c, реализующий интерфейс пользователя к калькулятору



```
#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[4]);
    Result = Calculate(Numeral, Operation);
    return 0;
}
```

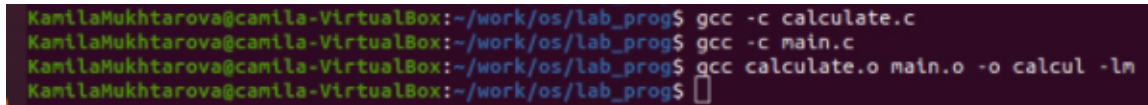
(рис.5) - скрипт

- Выполним компиляцию программы посредством gcc:

```
gcc -c calculate.c
```

```
gcc -c main.c
```

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



```
KamilaMukhtarova@camila-VirtualBox:~/work/os/lab_prog$ gcc -c calculate.c
KamilaMukhtarova@camila-VirtualBox:~/work/os/lab_prog$ gcc -c main.c
KamilaMukhtarova@camila-VirtualBox:~/work/os/lab_prog$ gcc calculate.o main.o -o calcul -lm
KamilaMukhtarova@camila-VirtualBox:~/work/os/lab_prog$
```

(рис. 6) - компиляция

В ходе компиляции программы никаких ошибок выявлено не было.

- Создадим Makefile со следующим содержанием:

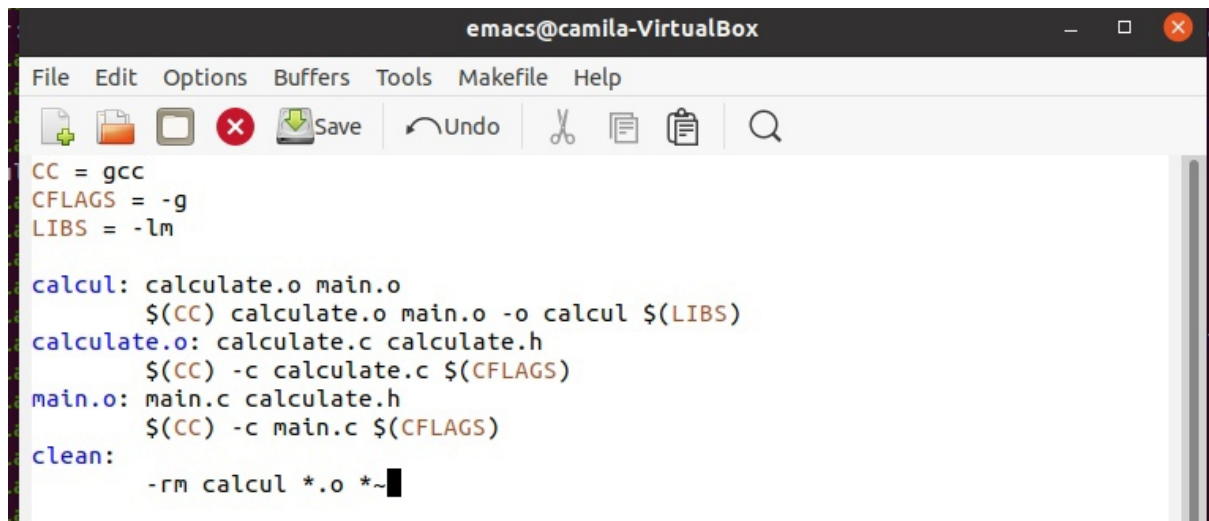
```
## Makefile # CC = gcc CFLAGS = LIBS = -lm calcul: calculate.o main.o gcc calculate.o
main.o -o calcul $(LIBS) calculate.o: calculate.c calculate.h gcc -c calculate.c $(CFLAGS)
main.o: main.c calculate.h gcc -c main.c $(CFLAGS) clean: -rm calcul *.o *~ # End Makefile
```

Данный файл необходим для автоматической компиляции файлов calculate.c (цель calculate.o), main.c (цель main.o), а также их объединения в один исполняемый файл calcul (цель calcul). Цель clean нужна для автоматического удаления файлов.

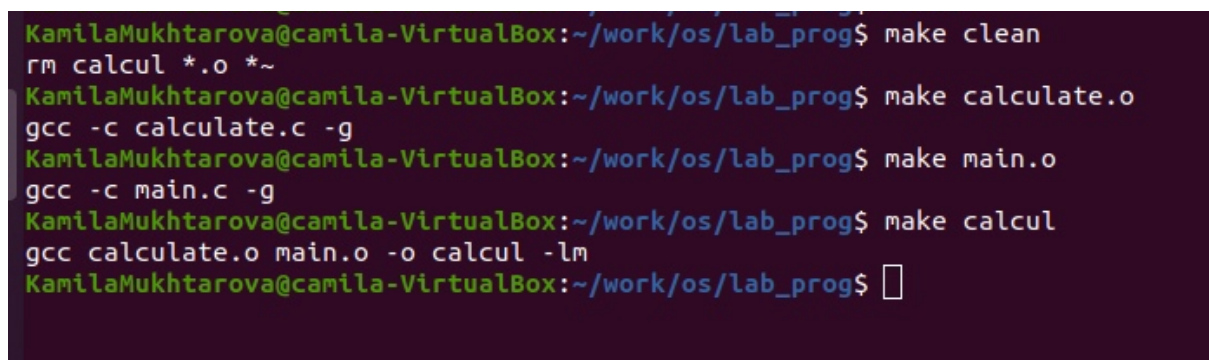
Переменная CC отвечает за утилиту для компиляции. Переменная CFLAGS отвечает за опции в данной утилите. Переменная LIBS отвечает за опции для объединения объектных файлов в один исполняемый файл.

В переменную CFLAGS добавила опцию -g, необходимую для компиляции объектных файлов и их использования в программе отладчика GDB. Сделала так, что утилита компиляции выбирается с помощью переменной CC. После этого я удалила исполняемые и объектные файлы из каталога с помощью команды «make clean» . Выполнила компиляцию файлов, используя команды «make calculate.o», «make main.o», «male calcul»





(рис. 7) - Makefile

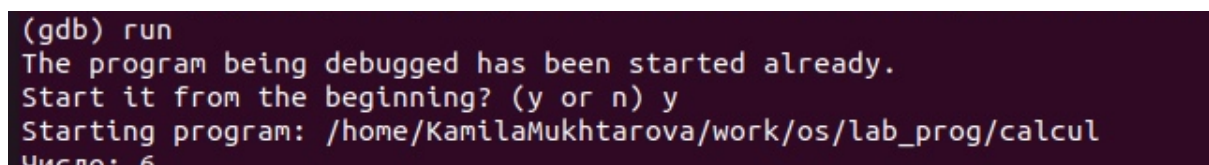


(рис. 7) - компиляция

- С помощью gdb выполним отладку программы calcul

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

Для запуска программы внутри отладчика ввела команду «run»



(рис.8) - запуск программы

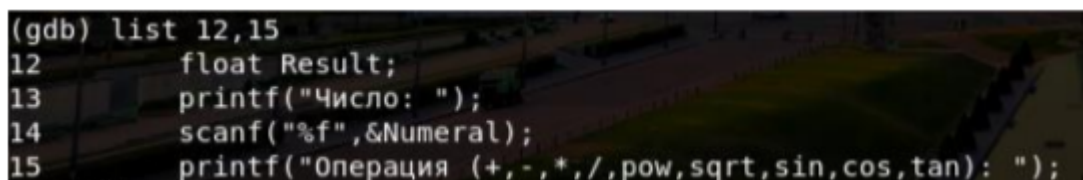
Для постраничного (по 10 строк) просмотра исходного кода использовала команду «list»



```
(gdb) list
1  ///////////////////////////////////////////////////
2  // main.c
3
4  #include <stdio.h>
5  #include "calculate.h"
6
7  int
8  main (void)
9  {
10     float Numeral;
(gdb) list
11     char Operation[4];
12     float Result;
13     printf("Число: ");
14     scanf("%f",&Numeral);
15     printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");
16     scanf("%s",&Operation);
17     Result = Calculate(Numeral, Operation);
18     printf("%6.2f\n",Result);
19     return 0;
20 }
```

(рис. 9) - просмотр по 10 строк

Для просмотра строк с 12 по 15 основного файла использовала команду «list 12,15»



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

(рис. 10) - просмотр с 12 по 15 строк

Для просмотра определенных строк не основного файла использовала команду «list calculate.c:20,29»



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

(рис. 11) - определённые строки

Установила точку останова в файле calculate.c на строке номер 21, используя команды «list calculate.c:20,27» и «break 21»

```
(gdb) break 21
Breakpoint 1 at 0x55555555226: file calculate.c, line 21.
(gdb) █
```

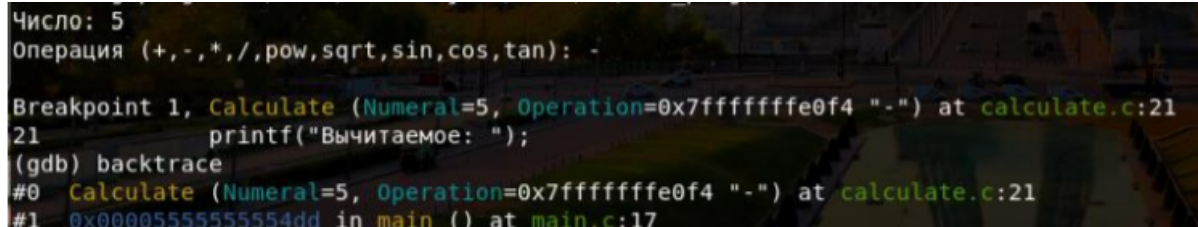
(рис. 12) - точка останова

Вывела информацию об имеющихся в проекте точках останова с помощью команды «info breakpoints»

```
(gdb) info breakpoints
Num   Type             Disp Enb Address                  What
1     breakpoint       keep y   0x000055555555226 in Calculate at calculate.c:21
(gdb) █
```

(рис.13) - информация о точке останова

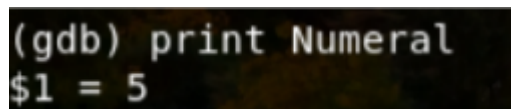
Запустила программу внутри отладчика и убедилась, что программа остановилась в момент прохождения точки останова. Использовала команды «run», «5», «-» и «backtrace»



```
Число: 5
Операция (+, -, *, /, pow, sqrt, sin, cos, tan): -
Breakpoint 1, Calculate (Numeral=5, Operation=0x7fffffffef4 "-") at calculate.c:21
21      printf("Вычитаемое: ");
(gdb) backtrace
#0 Calculate (Numeral=5, Operation=0x7fffffffef4 "-") at calculate.c:21
#1 0x0000555555555555 in main () at main.c:17
```

(рис. 14) - запуск программы

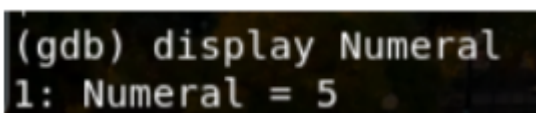
Посмотрела, чему равно на этом этапе значение переменной Numeral, введя команду «print Numeral»



```
(gdb) print Numeral
$1 = 5
```

(рис. 15) - просмотр значения

Сравнила с результатом вывода на экран после использования команды «display Numeral». Значения совпадают



```
(gdb) display Numeral
1: Numeral = 5
```

(рис. 16) - сравнение

Убрала точки останова с помощью команд «info breakpoints» и «delete 1»

```
(gdb) info breakpoints
Num      Type          Disp Enb Address              What
1        breakpoint    keep y   0x0000555555555226 in Calculate at calculate.c:21
breakpoint already hit 1 time
(gdb) delete 1
(gdb)
```

(рис. 17) - удаление точки останова

Далее воспользовалась командами «splint calculate.c» и «splint main.c». С помощью утилиты splint выяснилось, что в файлах calculate.c и main.c присутствует функция чтения scanf, возвращающая целое число (тип int), но эти числа не используются и нигде не сохраняются. Утилита вывела предупреждение о том, что в файле calculate.c происходит сравнение вещественного числа с нулем. Также возвращаемые значения (тип double) в функциях pow, sqrt, sin, cos и tan записываются в переменную типа float, что свидетельствует о потере данных.

```

Splint 3.1.2 --- 05 Sep 2017

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:10:32: Function parameter Operation declared as manifest array
                    (size constant is meaningless)
calculate.c: (in function Calculate)
calculate.c:16:7: 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:22:7: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:28:7: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:34:7: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:35:10: 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:38:10: Return value type double does not match declared type float:
                    (HUGE_VAL)
  To allow all numeric types to match, use +relaxtypes.
calculate.c:46:7: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:47:13: Return value type double does not match declared type float:
                    (pow(Numeral, SecondNumeral))
calculate.c:50:11: Return value type double does not match declared type float:
                    (sqrt(Numeral))
calculate.c:52:11: Return value type double does not match declared type float:
                    (sin(Numeral))
calculate.c:54:11: Return value type double does not match declared type float:
                    (cos(Numeral))
calculate.c:56:11: Return value type double does not match declared type float:
                    (tan(Numeral))
calculate.c:60:13: Return value type double does not match declared type float:

```

```

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)
main.c: (in function main)
main.c:14:3: 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:16:14: 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:16:11: Corresponding format code
main.c:16:3: Return value (type int) ignored: scanf("%s", &Ope...
Finished checking --- 4 code warnings

```

(рис. 18, 19) - утилита splint

Вывод: В ходе выполнения данной лабораторной работы я приобрела простейшие навыки разработки, анализа, тестирования и отладки приложений в ОС типа UNIX/Linux на примере создания на языке программирования C калькулятора с простейшими функциями