

# Laboratory №14

---

Krupennikova V.

MAY-2021

RUDN University, Moscow, Russian Federation

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.

1. Ознакомиться с теоретическим материалом.
2. Изучить основы программирования в оболочке ОС UNIX/Linux.
3. Выполнить упражнения.
4. Ответить на контрольные вопросы.

- 1) В домашнем каталоге создаю подкаталог `~/work/os/lab_prog` помощью команды «`mkdir -p ~/work/os/lab_prog`»

```
vakrupennikova@dk6n66 ~ $ mkdir -p ~/work/os/lab_prog
```

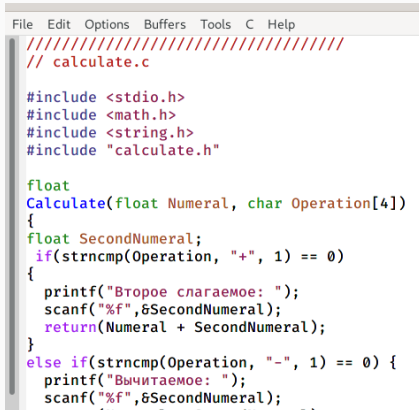
Figure 1: Создание подкаталога

2) Создала в каталоге файлы: calculate.h, calculate.c, main.c, используя команды «cd ~/work/os/lab\_prog» и «touch calculate.h calculate.c main.c»

```
vakrupennikova@dk6n66 ~ $ cd ~/work/os/lab_prog
vakrupennikova@dk6n66 ~/work/os/lab_prog $ touch calculate.h calculate.c main.c
vakrupennikova@dk6n66 ~/work/os/lab_prog $ ls
calculate.c calculate.h main.c
```

Figure 2: Создание файлов для работы

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. Открыв редактор Emacs, приступила к редактированию созданных файлов. Реализация функций калькулятора в файле `calculate.c`



```
File Edit Options Buffers Tools C Help
////////////////////////////////////
// calculate.c

#include <stdio.h>
#include <math.h>
#include <string.h>
#include "calculate.h"

float
Calculate(float Numeral, char Operation[4])
{
    float SecondNumeral;
    if(strcmp(Operation, "+", 1) == 0)
    {
        printf("Второе слагаемое: ");
        scanf("%f", &SecondNumeral);
        return(Numeral + SecondNumeral);
    }
    else if(strcmp(Operation, "-", 1) == 0) {
        printf("Вычитаемое: ");
        scanf("%f", &SecondNumeral);
```

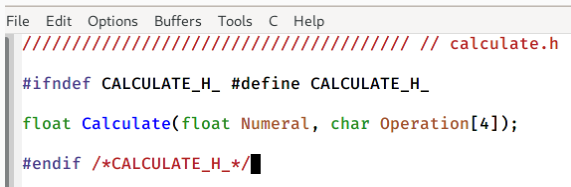
```

} 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);
} }

```

Figure 4: Продолжение кода

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

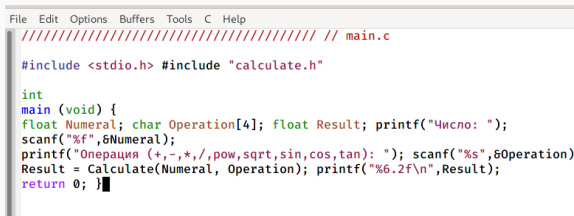
A screenshot of a code editor window. The title bar shows 'File Edit Options Buffers Tools C Help'. The code is as follows:

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

Figure 5: calculate.h



Основной файл main.c, реализующий интерфейс пользователя к калькулятору



```
File Edit Options Buffers Tools C Help
//////////////////////////////////// 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; }
```

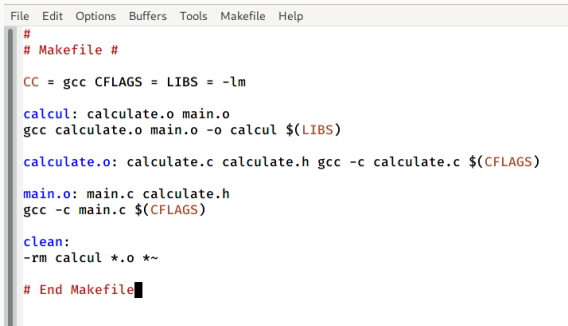
Figure 6: main.c

- 3) Выполнила компиляцию программы посредством gcc (версия компилятора:8.3.0-19),используя команды «gcc-ccalculate.c», «gcc-cmain.c» и «gcccalculate.omain.o-ocalcul-lm»

```
vakrupennikova@dk6n66 ~/work/os/lab_prog $ gcc -c calculate.c
vakrupennikova@dk6n66 ~/work/os/lab_prog $ gcc -c main.c
vakrupennikova@dk6n66 ~/work/os/lab_prog $ gcc calculate.o main.o -o calcul -lm
```

Figure 7: Команды gcc

- 4) В ходе компиляции программы никаких ошибок выявлено не было.
- 5) Создала Makefile с необходимым 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)

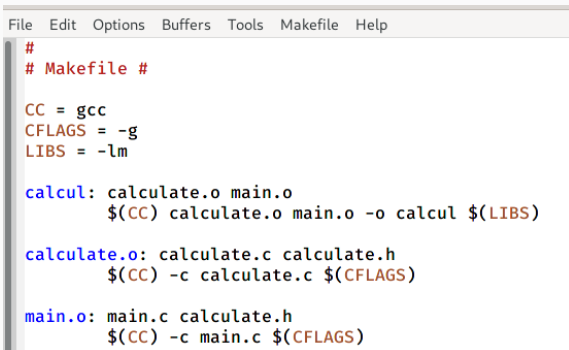
clean:
-rm calcul *.o *~

# End Makefile
```

Figure 8: 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.

6)Далее исправила Makefile



```
#  
# Makefile #  
  
CC = gcc  
CFLAGS = -g  
LIBS = -lm  
  
calcul: calculate.o main.o  
    $(CC) calculate.o main.o -o calcul $(LIBS)  
  
calculate.o: calculate.c calculate.h  
    $(CC) -c calculate.c $(CFLAGS)  
  
main.o: main.c calculate.h  
    $(CC) -c main.c $(CFLAGS)
```

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», «make calcul».

Далее с помощью gdb выполнила отладку программы calcul. Запустила отладчик GDB, загрузив в него программу для отладки, используя команду: «gdb ./calcul»

```
vakrupennikova@dk6n66 ~/work/os/lab_prog $ gdb ./calcul
GNU gdb (Gentoo 10.1 vanilla) 10.1
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-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
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) run
```

Figure 10: Запуск GDB

Для запуска программы внутри отладчика ввела команду «run»

```
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /afs/.dk.sci.pfu.edu.ru/home/m/m/mmbezruk/calcul
Число: 5
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): *
Множитель: 7
35.00
```

Figure 11: Команда «run»

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("%.2f\n",Result);
19     return 0;
20 }
```

Figure 12: Команда «list»



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

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

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

Figure 13: Команда «list calculate.c:20,29»

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

```
(gdb) list calculate.c:20,27
20     {
21     printf("Вычитаемое: ");
22     scanf("%f",&SecondNumeral);
23     return(Numeral - SecondNumeral);
24     }
25     else if(strncmp(Operation, "*", 1) == 0)
26     {
27     printf("Множитель: ");
(gdb) break 21
Breakpoint 1 at 0x555555400955: file calculate.c, line 21.
```

Figure 14: 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   0x000055555400955 in Calculate at calculate.c:21
```

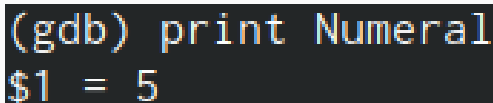
Figure 15: Вывод информации

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

```
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/m/m/mmbezruk/calcul
Число: 5
Операция (+, -, *, /, pow, sqrt, sin, cos, tan): -
Breakpoint 1, Calculate (Numeral=5, Operation=0x7fffffffcea4 "-") at calculate.c:21
21     printf("Вычитаемое: ");
```

Figure 16: Запуск программы

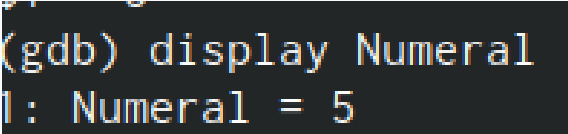
Посмотрела, чему равно на этом этапе значение переменной Numeral, введя команду «print Numeral»



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

Figure 17: Переменная Numeral

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



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

Figure 18: 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   0x000055555400955 in Calculate at calculate.c:21
      breakpoint already hit 1 time
(gdb) delete 1
```

Figure 19: Удаление точки останова

7) С помощью утилиты splint проанализировала коды файлов calculate.c и main.c.

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.



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 С калькулятора с простейшими функциями.

The end.