

Лабораторная работа №14

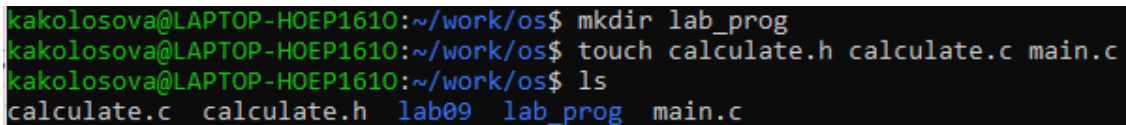
Колосова Кристина Александровна

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.

Выполнение лабораторной работы

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



```
kakolosova@LAPTOP-НОЕР1610:~/work/os$ mkdir lab_prog
kakolosova@LAPTOP-НОЕР1610:~/work/os$ touch calculate.h calculate.c main.c
kakolosova@LAPTOP-НОЕР1610:~/work/os$ ls
calculate.c calculate.h lab09 lab_prog main.c
```

Рис. 1. Создание подкаталога

2. Создала 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.

файл 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));
else { printf("Неправильно введено действие "); return(HUGE_VAL); } }

```

```

#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 0!");
            return (HUGE_VAL);
        }
        else return (Numeral/SecondNumeral);
    }
    else if(strncmp(Operation, "pow", 3)==0)
    {
        printf("степень: ");
        scanf("%f", &SecondNumeral);

```

Рис. 2. файл calculate.c

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_*/

```

```

#ifndef CALCULATE_H_
#define CALCULATE_H_

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

#endif /*CALCULATE_H_*/

```

Рис. 3. файл calculate.h

Основной файл main.c, реализующий интерфейс пользователя к калькулятору:
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;
}

```

```

#include <stdio.h>
#include "calculate.h"
int
main(void)
{
float Numeral;
char Operation[4];
float Result;
printf("Число: ");
scanf("%f", &Numeral);
printf("Операция (+, -, /, *, sqrt, sin, cos, tan): ");
scanf("%s", Operation);
Result=Calculate(Numeral, Operation);
printf("%6.2f\n", Result);
return 0;
}

```

Рис. 4. файл main.c

3. Выполнила компиляцию программы посредством gcc:

gcc -c calculate.c

gcc -c main.c

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

```

kakoosova@LAPTOP-НОЕР1610:/$ sudo gcc -c calculate.c
kakoosova@LAPTOP-НОЕР1610:/$ sudo gcc -c main.c
kakoosova@LAPTOP-НОЕР1610:/$ sudo gcc calculate.o main.o -o calcul -lm

```

Рис. 5. компиляция

4. Исправила синтаксические ошибки.
5. Создала Makefile со следующим содержанием:

```
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*~
```

Рис. 6. Makefile

Тут прописано какие команды будут выполняться: компиляция и создание файлов.

6. С помощью gdb выполнила отладку программы calcul:

- Запустила отладчик GDB, загрузив в него программу для отладки: `gdb ./calcul`
- Для запуска программы внутри отладчика ввела команду `run`:

```
kakolosova@LAPTOP-HOEP1610:/$ 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) run
Starting program: ./calcul
Число: 12
Операция (+, -, /, *, sqrt, sin, cos, tan): +
Второе слагаемое: 15
27.00
[Inferior 1 (process 847) exited normally]
(gdb)
```

Рис. 7. Загрузка и отладка

– Для постраничного (по 9 строк) просмотра исходного код используйте команду `list`: `list`

ДЛЯ ВЫПОЛНЕНИЯ ЭТИХ ПУНКТОВ ПРИШЛОСЬ ЗАХОДИТЬ ЧЕРЕЗ ВИРТУАЛЬНУЮ МАШИНУ, ПОЭТОМУ СТИЛЬ СКРИНШОТОВ ОТЛИЧАЕТСЯ! ПРИ ПОПЫТКЕ ВЫПОНИТЬ КОМАНДУ LIST ВСЕГДА ПОЯВЛЯЛОСЬ

```
(gdb) list
1      init-first.c: No such file or directory.
(gdb) list
1      in init-first.c
```

Рис. 8. Просмотр кода когда не получилось

А вот как должно быть:

(С русскими символами выводилось некорректно, исправила на транслит...)

```
(gdb) list
1      #include <stdio.h>
2      #include "calculate.h"
3
4      main(void){
5
6          float Numeral;
7          char Operation[4];
8          float Result;
9          printf("Num: \n");
10         scanf("%f",&Numeral);
```

Рис. 9. 1-9 строки

– Для просмотра строк с 12 по 15 основного файла используйте list с параметрами: list 12,15

```
(gdb) list 12,15
12         scanf("%s",Operation);
13         Result = Calculate(Numeral,Operation);
14         printf("%6.2f\n",Result);
15         return 0;
(gdb)
```

Рис. 10. 12-15 строки

– Для просмотра определённых строк не основного файла используйте list с параметрами: list calculate.c:20,29

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

Рис. 11. 20-29 строки

– Установите точку останова в файле calculate.c на строке номер 21: list calculate.c:20,27
break 21

– Вывела информацию об имеющихся в проекте точка останова: info breakpoints

```
(gdb) b 21
Breakpoint 1 at 0x7ffff7ca0f20: file init-first.c, line 44.
(gdb) info b
Num      Type           Disp Enb Address              What
1        breakpoint     keep y   0x00007ffff7ca0f20 in __libc_init_first at init-first.c:44
(gdb)
```

Рис. 12. Точка останова

– Запустила программу внутри отладчика и убедилась, что программа остановится в момент прохождения точки останова: run 5 - backtrace

– Отладчик выдает следующую информацию:

#0 Calculate (Numeral=5, Operation=0x7fffffd280 "-")

at calculate.c:21

#1 0x000000000400b2b in main () at main.c:1778

– Уберите точки останова: info breakpoints delete 1

```
(gdb) d breakpoints 1
(gdb)
```

Рис. 13. Информация о точке останова

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

```
kakolosova@LAPTOP-HOEP1610:/$ 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:10:1: 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:12:1: Return value (type int) ignored: scanf("%s", Oper...)

Finished checking --- 3 code warnings
kakolosova@LAPTOP-HOEP1610:/$ 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:6:31: Function parameter Operation declared as manifest array
(size constant is meaningless)
/calculate.c: (in function Calculate)
/calculate.c:12:2: 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:18:2: Return value (type int) ignored: scanf("%f", &Sec...)
/calculate.c:24:2: Return value (type int) ignored: scanf("%f", &Sec...)
/calculate.c:30:2: Return value (type int) ignored: scanf("%f", &Sec...)
/calculate.c:31:5: Dangerous equality comparison involving float types:
```

Рис. 14. Анализ с помощью splint

Выводы

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.

Библиография

Вот тут посмотрела про init

<https://stackoverflow.com/questions/31379422/why-is-init-from-glibcs-csu-init-first-c-called-before-start-even-if-start-i>

тут про корректную передачу массива символов как аргумент

[Занятие 14. Строки и символьные массивы в языке Си. \(youngcoder.net\)](#)