

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: Джунусова Рузель

Группа: НПИбд-01-20

МОСКВА

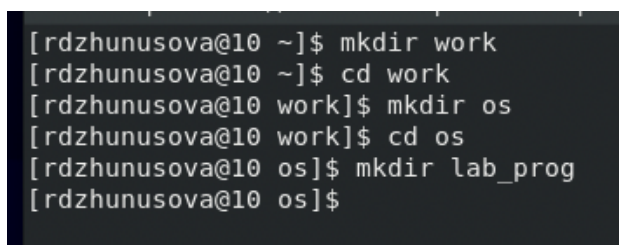
2021

Цель работы:

В данной лабораторной работе мне будет необходимо 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.

Выполнение лабораторной работы:

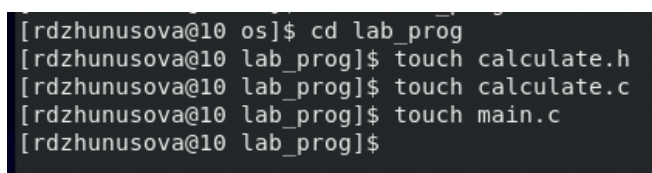
1. В домашнем каталоге создала подкаталог `~/work/os/lab_prog`.
Использовала команды `mkdir` и `cd`. (рис. 4.1)



```
[rdzhunusova@10 ~]$ mkdir work
[rdzhunusova@10 ~]$ cd work
[rdzhunusova@10 work]$ mkdir os
[rdzhunusova@10 work]$ cd os
[rdzhunusova@10 os]$ mkdir lab_prog
[rdzhunusova@10 os]$
```

Figure 4.1: Создание подкаталога

2. Создала в нём файлы: `calculate.h`, `calculate.c`, `main.c`. (рис. 4.2)



```
[rdzhunusova@10 os]$ cd lab_prog
[rdzhunusova@10 lab_prog]$ touch calculate.h
[rdzhunusova@10 lab_prog]$ touch calculate.c
[rdzhunusova@10 lab_prog]$ touch main.c
[rdzhunusova@10 lab_prog]$
```

Figure 4.2: Создание файлов

3. Реализация функций калькулятора в файле `calculate.c`. (рис. 4.3)

```

1 ////////////////////////////////////////////////// calculate.c
2
3 #include <stdio.h>
4 #include <math.h>
5 #include <string.h>
6 #include "calculate.h"
7
8 float
9 Calculate(float Numeral, char Operation[4])
10 {
11     float SecondNumeral;
12     if(strncmp(Operation, "+", 1) == 0)
13     {
14         printf("Второе слагаемое: ");
15         scanf("%f", &SecondNumeral);
16         return(Numeral + SecondNumeral);
17     }
18     else if(strncmp(Operation, "-", 1) == 0)
19     {
20         printf("Вычитаемое: ");
21         scanf("%f", &SecondNumeral);
22         return(Numeral - SecondNumeral);
23     }
24     else if(strncmp(Operation, "*", 1) == 0)
25     {
26         printf("Множитель: ");
27         scanf("%f", &SecondNumeral);
28         return(Numeral * SecondNumeral);
29     }
30     else if(strncmp(Operation, "/", 1) == 0)
31     {
32         printf("Делитель: ");
33         scanf("%f", &SecondNumeral);
34         if(SecondNumeral == 0)
35         {
36             printf("Ошибка: деление на ноль! ");
37             return(HUGE_VAL);
38         }
39         else
40             return(Numeral / SecondNumeral);
41     }
42     else if(strncmp(Operation, "pow", 3) == 0)
43     {
44         printf("Степень: ");
45         scanf("%f", &SecondNumeral);

```

Figure 4.3: Файл calculate.c

4. Интерфейсный файл calculate.h, описывающий формат вызова функции калькулятора. (рис. 4.4)

```

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

```

Figure 4.4: Файл calculate.h

5. Основной файл main.c, реализующий интерфейс пользователя к калькулятору. (рис. 4.5)

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

Figure 4.5: Файл main.c

6. Выполнила компиляцию программы посредством gcc: gcc -c calculate.c gcc -c main.c gcc calculate.o main.o -o calcul -lm (рис. 4.6)
7. Создал Makefile. (рис. 4.7)

```
#
# 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 4.7: Makefile

8. С помощью gdb выполнила отладку программы calcul (перед использованием gdb исправила Makefile). Запустила отладчик GDB, загрузив в него программу для отладки gdb ./calcul. Для запуска программы внутри отладчика ввела команду run. (рис. 4.8)

```

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
Starting program: /afs/.dk.sci.pfu.edu.ru/home/g/a/gabihstrov/work/os/lab_prog/calcul
Число: 1
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): +
Второе слагаемое: 1
      2.00
[Inferior 1 (process 17805) exited normally]

```

Figure 4.8: Отладка программы calcul

9. Для постраничного (по 9 строк) просмотра исходного код использовала команду list. (рис. 4.9)

```

(gdb) list
1      //////////////////////////////////////////// main.c
2
3      #include <stdio.h>
4      #include "calculate.h"
5
6      int
7      main (void)
8      {
9      float Numeral;
10     char Operation[4];
(gdb)

```

Figure 4.9: Команда list

10. Для просмотра строк с 12 по 15 основного файла использовала list с параметрами list 12,15. (рис. 4.10)

```

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

```

Figure 4.10: Просмотр строк

11. Для просмотра определённых строк не основного файла используйте list с параметрами: list calculate.c:20,29. (рис. 4.11)

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

Figure 4.11: Просмотр определённых строк

12. Установите точку останова в файле calculate.c на строке номер 21: list calculate.c:20,27 break 21 (рис. 4.12)

```
(gdb) list calculate.c:20,27
20     printf("Вычитаемое: ");
21     scanf("%f",&SecondNumeral);
22     return(Numeral - SecondNumeral);
23 }
24 else if(strncmp(Operation, "*", 1) == 0)
25 {
26     printf("Множитель: ");
27     scanf("%f",&SecondNumeral);
(gdb) break 21
Breakpoint 1 at 0x555555400966: file calculate.c, line 21.
(gdb)
```

Figure 4.12: Установила точку останова

13. Выведите информацию об имеющихся в проекте точка останова: info breakpoints. (рис. 4.13)

```
(gdb) info breakpoints
Num   Type             Disp Enb Address            What
1     breakpoint       keep y 0x0000555555400966 in Calculate at calculate.c:21
(gdb)
```

Figure 4.13: Информация о точке останова

14. Запустила программу внутри отладчика и убедилась, что программа остановилась в момент прохождения точки останова (рис. 4.14)

```
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/g/a/gabihstrov/work/os/lab_prog/calcul
Число: 2
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): -

Breakpoint 1, Calculate (Numeral=2, Operation=0x7fffffffce14 "-") at calculate.c:21
21      scanf("%f",&SecondNumeral);
(gdb)
```

Figure 4.14: Остановка программы

15. Посмотрите, чему равно на этом этапе значение переменной Numeral, введя:
print Numeral. На экран выведено число 2. (рис. 4.15)

```
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/g/a/gabihstrov/work/os/lab_prog/calcul
Число: 2
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): -

Breakpoint 1, Calculate (Numeral=2, Operation=0x7fffffffce14 "-") at calculate.c:21
21      scanf("%f",&SecondNumeral);
(gdb) print Numeral
$1 = 2
(gdb)
```

Figure 4.15: Переменная Numeral

16. Сравнила с результатом вывода на экран после использования команды: display Numeral. Цифры сходятся. (рис. 4.16)

```
(gdb) print Numeral
$1 = 2
(gdb) display Numeral
1: Numeral = 2
(gdb)
```

Figure 4.16: Сравнение цифр

17. Убрала точки останова: info breakpoints delete 1. (рис. 4.17)

```
(gdb) info breakpoints
Num   Type             Disp Enb Address          What
1     breakpoint       keep y  0x0000555555400966 in Calculate at calculate.c:21
      breakpoint already hit 1 time
(gdb) delete 1
(gdb)
```

Figure 4.17: Убрала точки

18. С помощью утилиты splint проанализировал коды файлов calculate.c и main.c. (рис. 4.18) (рис. 4.19)

```
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:9:31: Function parameter Operation declared as manifest array (size
constant is meaningless)
calculate.c: (in function Calculate)
calculate.c:15:1: 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:21:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:27:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:33:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:34:4: 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:37:7: Return value type double does not match declared type float:
(HUGE_VAL)
To allow all numeric types to match, use +relaxtypes.
calculate.c:45:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:46:7: Return value type double does not match declared type float:
(pow(Numeral, SecondNumeral))
calculate.c:49:7: Return value type double does not match declared type float:
(sqrt(Numeral))
calculate.c:51:7: Return value type double does not match declared type float:
(sin(Numeral))
calculate.c:53:7: Return value type double does not match declared type float:
(cos(Numeral))
calculate.c:55:7: Return value type double does not match declared type float:
(tan(Numeral))
calculate.c:59:7: Return value type double does not match declared type float:
(HUGE_VAL)
```

Figure 4.18: calculate.c

```
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:13:1: 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:15:12: 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:15:9: Corresponding format code
main.c:15:1: Return value (type int) ignored: scanf("%s", &Op...
```

Figure 4.19: main.c

другое.

Вывод:

В данной лабораторной работе мне успешно удалось 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 Библиографический список

1. Разработка Unix-приложений (<https://qna.habr.com/q/17069>)
2. Статический анализ кода (<https://www.jetinfo.ru/staticeskij-analiz-kodachto-mogut-instrumentalnye-sredstva/>)
3. Sprint layout аналог для linux (<https://a174.ru/sprint-layout-analog-dlyalinux/>)