

Отчёт по лабораторной работе №13

Паращенко Антонина Дмитриевна

Содержание

1	Цель работы	5
2	Ход лабораторной работы	6
3	Вывод	13

Список иллюстраций

2.1	Консоль	6
2.2	Скрипт	7
2.3	Скрипт	8
2.4	Скрипт	8
2.5	Компиляция	8
2.6	Компиляция	9
2.7	Скрипт кода	9
2.8	Отладка файла	9
2.9	Запуск программы	10
2.10	Просмотр файла	10
2.11	Тоска останова	11
2.12	Запуск файла	11
2.13	calculate.c	12
2.14	main.c	12

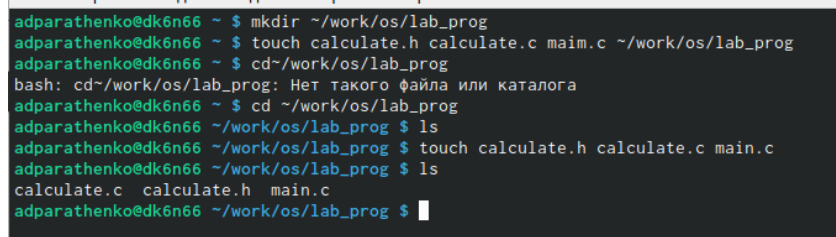
Список таблиц

1 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.

2 Ход лабораторной работы

- 1) В домашнем каталоге создаём подкаталог `~/work/os/lab_prog` и создаём в нём файлы: `calculate.h`, `calculate.c`, `main.c` (рис. 2.1)



```
adparathenko@dk6n66 ~ $ mkdir ~/work/os/lab_prog
adparathenko@dk6n66 ~ $ touch calculate.h calculate.c main.c ~/work/os/lab_prog
adparathenko@dk6n66 ~ $ cd ~/work/os/lab_prog
bash: cd~/work/os/lab_prog: Нет такого файла или каталога
adparathenko@dk6n66 ~ $ cd ~/work/os/lab_prog
adparathenko@dk6n66 ~/work/os/lab_prog $ ls
adparathenko@dk6n66 ~/work/os/lab_prog $ touch calculate.h calculate.c main.c
adparathenko@dk6n66 ~/work/os/lab_prog $ ls
calculate.c calculate.h main.c
adparathenko@dk6n66 ~/work/os/lab_prog $
```

Рис. 2.1: Консоль

- 2) Пишем скрипт файла `calculate.c` (рис. 2.2)

```

#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);
    }
}
}

```

Рис. 2.2: Скрипт

3) Пишем скрипт файла calculate.h (рис. 2.3)

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

Рис. 2.3: Скрипт

4) Пишем скрипт файла main.c (рис. 2.4)

```
#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;
}
```

Рис. 2.4: Скрипт

5) Выполняем компиляцию программы посредством gcc, исправляем ошибки и компилируем снова (рис. 2.5) - (рис. 2.6)

```
adparathenko@dk6n66 ~/work/os/lab_prog $ gcc -c calculate.c
adparathenko@dk6n66 ~/work/os/lab_prog $ gcc -c main.c
main.c: В функции «main»:
main.c:12:11: предупреждение: формат «%s» ожидает аргумент типа «char *», но аргумент 2 имеет тип «char (*)[4]» [-Wformat=]
   12 |     scanf("%s",&Operation);
      |           ~^~
      |           |
      |           | char (*)[4]
      |           char *
```

Рис. 2.5: Компиляция


```

adparathenko@dk6n66 ~/work/os/lab_prog $ gcc -c main.c
adparathenko@dk6n66 ~/work/os/lab_prog $ gcc -c -g calculate.c
adparathenko@dk6n66 ~/work/os/lab_prog $ gcc -c -g main.c
adparathenko@dk6n66 ~/work/os/lab_prog $ gcc -g calculate.o main.o -o calcul -lm
gcc: ошибка: o: Нет такого файла или каталога
gcc: ошибка: calcul: Нет такого файла или каталога
adparathenko@dk6n66 ~/work/os/lab_prog $ gcc -g calculate.o main.o -o calcul -lm
adparathenko@dk6n66 ~/work/os/lab_prog $

```

Рис. 2.6: Компиляция

- 6) Создайте Makefile, который выполняет компиляцию программы посредством gcc вместо ручного ввода. (рис. 2.7)

```

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)

clean:
    -rm calcul *.o *~

```

Рис. 2.7: Скрипт кода

- 7) Запустите отладчик GDB, загрузив в него программу для отладки. (рис. 2.8)

```

adparathenko@dk6n66 ~/work/os/lab_prog $ gdb ./calcul
GNU gdb (Gentoo 10.2 vanilla) 10.2
Copyright (C) 2021 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...
(gdb)

```

Рис. 2.8: Отладка файла

- 8) Запускаем программу внутри отладчика с помощью команды run (рис. 2.9)

```
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/d/adparathenko/work/os/lab_prog/calcul
Число: 5
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): -
Вычитаемое: 2
3.00
[Inferior 1 (process 15073) exited normally]
(gdb)
```

Рис. 2.9: Запуск программы

- 9) Для постраничного (по 9 строк) просмотра исходного код используем команду list. Для просмотра строк с 12 по 15 основного файла используем list с параметрами. Для просмотра определённых строк не основного файла используйте list с параметрами (рис. 2.10)

```
(gdb) list
1      #include<stdio.h>
2      #include"calculate.h"
3      int
4      main(void)
5      {
6          float Numeral;
7          char Operation[4];
8          float Result;
9          printf("Число: ");
10         scanf("%f",&Numeral);
(gdb) list 12, 15
12         scanf("%s",Operation);
13         Result = Calculate(Numeral, Operation);
14         printf("%6.2f\n",Result);
15         return 0;
(gdb) list calculate.c:20, 27
20     }
21     else if(strncmp(Operation,"*", 1) == 0)
22     {
23         printf("Множитель: ");
24         scanf("%f",&SecondNumeral);
25         return(Numeral* SecondNumeral);
26     }
27     else if(strncmp(Operation,"/", 1) == 0)
(gdb)
```

Рис. 2.10: Просмотр файла

- 10) Устанавливаем точку останова в файле calculate.c на строке номер 21. Выводим информацию об имеющихся в проекте точках останова. (рис. 2.11)

```

(gdb) list calculate.c:20,27
20     }
21     else if(strncmp(Operation,"*", 1) == 0)
22     {
23         printf("Множитель: ");
24         scanf("%f",&SecondNumeral);
25         return(Numeral* SecondNumeral);
26     }
27     else if(strncmp(Operation,"/", 1) == 0)
(gdb) break 21
Breakpoint 1 at 0x5555555527d: file calculate.c, line 21.
(gdb) info breakpoints
Num   Type             Disp Enb Address            What
1     breakpoint       keep y   0x00005555555527d in Calculate at calculate.c:21
(gdb)

```

Рис. 2.11: Тоска останова

- 11) С помощью команды `backtrace` смотрим весь стек вызываемых функций от начала программы до текущего места (рис. 2.12)

```

(gdb) backtrace
No stack.
(gdb)

```

Рис. 2.12: Запуск файла

- 12) С помощью утилиты `splint` попробуйте проанализировать коды файлов `calculate.c` и `main.c`. (рис. 2.13)

```

adparathenko@dk6n66 ~/work/os/lab_prog $ splint calculate.c
Splint 3.1.2 --- 13 Jan 2021

calculate.h:3:36: 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:6:30: Function parameter Operation declared as manifest array (size
constant is meaningless)
calculate.c: (in function Calculate)
calculate.c:12:7: 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:18:7: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:24:7: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:30:7: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:31:10: 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: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:42:7: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:43:13: Return value type double does not match declared type float:
(pow(Numeral, SecondNumeral))
calculate.c:46:11: Return value type double does not match declared type float:
(sqrt(Numeral))
calculate.c:48:11: Return value type double does not match declared type float:
(sin(Numeral))
calculate.c:50:11: Return value type double does not match declared type float:
(cos(Numeral))
calculate.c:52:11: Return value type double does not match declared type float:
(tan(Numeral))
calculate.c:56:13: Return value type double does not match declared type float:
(HUGE_VAL)

Finished checking --- 15 code warnings
adparathenko@dk6n66 ~/work/os/lab_prog $

```

Рис. 2.13: calculate.c

```

Finished checking --- 15 code warnings
adparathenko@dk6n66 ~/work/os/lab_prog $ splint main.c
Splint 3.1.2 --- 13 Jan 2021

calculate.h:3:36: 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:10:3: 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:12:3: Return value (type int) ignored: scanf("%s", Oper...

Finished checking --- 3 code warnings

```

Рис. 2.14: main.c

3 Вывод

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.