

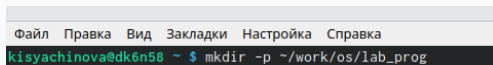
Презентация по лабораторной работе №13

Сячинова Ксения Ивановна, НПМбд-02-21

Российский Университет Дружбы Народов

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. В домашнем каталоге создаём подкаталог `~/work/os/lab_prog` с помощью команды `"mkdir -p"`. (рис. 1)

A screenshot of a terminal window. The top bar is light gray with menu items: 'Файл', 'Правка', 'Вид', 'Закладки', 'Настройка', 'Справка'. The terminal text shows a user prompt 'kisyachinova@dk6n58 ~ \$' followed by the command 'mkdir -p ~/work/os/lab_prog'.

```
Файл  Правка  Вид  Закладки  Настройка  Справка
kisyachinova@dk6n58 ~ $ mkdir -p ~/work/os/lab_prog
```

Figure 1: Создание подкаталога

2. Затем перейдём в каталог и создадим файлы: calculate.h, calculate.c, main.c. Делаю это с помощью команды “touch”.(рис. 2)

```
kisyachinova@dk6n58 ~ $ cd ~/work/os/lab_prog
kisyachinova@dk6n58 ~/work/os/lab_prog $ touch calculate.h calculate.c main.c
kisyachinova@dk6n58 ~/work/os/lab_prog $ ls
calculate.c calculate.h main.c
kisyachinova@dk6n58 ~/work/os/lab_prog $
```

Figure 2: Создание файлов

Создадим 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. Реализация функций калькулятора будет делать в файле calculate.c.(рис. 3),(рис. 4)

```
1 //////////////////////////////////////////////////
2 #include <stdio.h>
3 #include <math.h>
4 #include <string.h>
5 #include <stdlib.h>
6 #define MAX_SIZE 100
7
8 //Function to calculate the result of an operation
9
10 float calculate(float num1, char operation, float num2)
11 {
12     if (operation == '+') { return num1 + num2; }
13     else if (operation == '-') { return num1 - num2; }
14     else if (operation == '*') { return num1 * num2; }
15     else if (operation == '/') { return num1 / num2; }
16     else if (operation == '^') { return pow(num1, num2); }
17     else if (operation == 'sqrt') { return sqrt(num1); }
18     else if (operation == 'sin') { return sin(num1); }
19     else if (operation == 'cos') { return cos(num1); }
20     else if (operation == 'tan') { return tan(num1); }
21     else { return 0; }
22 }
23
24 //Function to get a float number from the user
25 float get_float()
26 {
27     float num;
28     char str[100];
29     printf("Enter a float number: ");
30     scanf("%f", &num);
31     return num;
32 }
33
34 //Function to get a char operation from the user
35 char get_operation()
36 {
37     char op;
38     printf("Enter an operation: ");
39     scanf("%c", &op);
40     return op;
41 }
42
43 //Function to get a float number from the user
44 float get_float()
45 {
46     float num;
47     char str[100];
48     printf("Enter a float number: ");
49     scanf("%f", &num);
50     return num;
51 }
```

Figure 3: Файл calculate.c

```

28     }
29     else if(strncmp(Operation, "/", 1) == 0)
30     {
31         printf("Делитель: ");
32         scanf("%f", &SecondNumeral);
33         if(SecondNumeral == 0)
34         {
35             printf("Ошибка: деление на ноль");
36             return(HUGE_VAL);
37         }
38         else
39             return(Numeral / SecondNumeral);
40     }
41     else if(strncmp(Operation, "pow", 3) == 0)
42     {
43         printf("Степень: ");
44         scanf("%f", &SecondNumeral);
45         return(pow(Numeral, SecondNumeral));
46     }
47     else if(strncmp(Operation, "sqrt", 4) == 0)
48         return(sqrt(Numeral));
49     else if(strncmp(Operation, "sin", 3) == 0)
50         return(sin(Numeral));
51     else if(strncmp(Operation, "cos", 3) == 0)
52         return(cos(Numeral));
53     else if(strncmp(Operation, "tan", 3) == 0)
54         return(tan(Numeral));
55     else
56     {
57         printf("Неправильно введено действие");
58     }
59 }
60

```

Figure 4: Файл calculate.c

Интерфейсный файл calculate.h, который описывает формат вызова функции калькулятора.(рис. 5)

```
1 //////////////////////////////////////////////////
2 // calculate.h
3 #ifndef CALCULATE_H_
4 #define CALCULATE_H_
5
6 float Calculate(float Numeral, char Operation[4]);
7
8 #endif/*CALCULATE_H_*/
```

Figure 5: Файл calculate.h

Основной файл main.c, реализующий интерфейс пользователя к калькулятору.(рис. 6)

```
1 //////////////////////////////////////////////////
2 // main.c
3
4 #include<stdio.h>
5 #include"calculate.h"
6
7 int
8 main (void)
9 {
10     float Numeral;
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 }
```

Figure 6: Файл main.c

3. Выполняем компиляцию файлов посредством “gcc”. (рис. 7)

```
kisyachinova@dk6n58 ~/work/os/lab_prog $ gcc -c calculate.c
kisyachinova@dk6n58 ~/work/os/lab_prog $ gcc -c main.c
kisyachinova@dk6n58 ~/work/os/lab_prog $ gcc calculate.o main.o -o calcul -lm
kisyachinova@dk6n58 ~/work/os/lab_prog $
```

Figure 7: Компиляция

4. В ходе компиляции ошибок не выявлено.

5. Создадим Makefile с необходимым содержанием. Он необходим для автоматической компиляции файлов calculate.c (цель calculate.o), main.c (цель main.o), а так же их объединения в один исполняемый файл 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.(рис. 8)

```
1 #
2 # Makefile
3 #
4
5 CC = gcc
6 CFLAGS=
7 LIBS= -lm
8
9 calcul: calculate.o main.o
10     gcc calculate.o main.o -o calcul $(LIBS)
11
12 calculate.o: calculate.c calculate.h
13     gcc -c calculate.c $(CFLAGS)
14
15 main.o: main.c calculate.h
16     gcc -c main.c $(CFLAGS)
17
18 clean:
19     -rm calcul *.o *~
20
21 # End Makefile
```

Figure 8: Makefile

6. Далее изменим файл. В переменную CFLAGS добавим “-g”, которая необходима для компиляции объектных файлов и их использования в программе отладчика GDB. Также, компиляция выбирается с помощью переменной CC.(рис. 9)

```
1 #
2 # Makefile
3 #
4
5 CC = gcc
6 CFLAGS = -g
7 LIBS= -lm
8
9 calcul: calculate.o main.o
10     $(CC) calculate.o main.o -o calcul $(LIBS)
11
12 calculate.o: calculate.c calculate.h
13     $(CC) -c calculate.c $(CFLAGS)
14
15 main.o: main.c calculate.h
16     $(CC) -c main.c $(CFLAGS)
17
18 clean:
19     -rm calcul *.o *~
20
21 # End Makefile
```

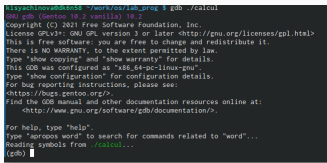
Figure 9: Makefile 2

Затем выполняем компиляцию файлов с помощью команды “make”.(рис. 10)

```
kisyachinova@dk6n58 ~/work/os/lab_prog $ make calculate.o
gcc -c calculate.c -g
kisyachinova@dk6n58 ~/work/os/lab_prog $ make main.o
gcc -c main.c -g
kisyachinova@dk6n58 ~/work/os/lab_prog $ make calcul
gcc calculate.o main.o -o calcul -lm
kisyachinova@dk6n58 ~/work/os/lab_prog $
```

Figure 10: Компиляция

После этого выполняем gdb отладку программы calcul. Запускаем GDB и загружаем в него программу для отладки, используя команду “gdb ./calcul”.(рис. 11)



```
kislayachinov@ubuntu: ~/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) 
```

Figure 11: Отладчик GDB

Далее вводим команду “run” для запуска программы внутри отладчика.(рис. 12)

```
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/k/i/kisyachinova/work/os/lab_prog/calcul
Число: 6
Операция (*,-,*,/,pow,sqrt,sin,cos,tan): +
Второе слагаемое: 12
20.00
[Inferior 1 (process 15859) exited normally]
(gdb)
```

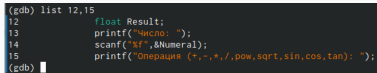
Figure 12: Запуск программы

Для постраничного просмотра исходного кода используем команду “list”.
(рис. 13)

```
(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)
11     char Operation[*];
12     float Result;
13     printf("Numco: ");
14     scanf("%f",&Numeral);
15     printf("Onesquare (%f,%f,pow,sqrt,sin,cos,tan): ");
16     scanf("%s",Operation);
17     Result = Calculate(Numeral, Operation);
18     printf("%s.%f\n",Result);
19     return 0;
20 }
```

Figure 13: Просмотр кода

Для просмотра строк с 12 по 15 основного файла используем команду “list 12,15”. (рис. 14)



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

Figure 14: Просмотр строк

Для просмотра определённых строк не основного файла используем команду “list calculate.c:20,29”.(рис. 15)

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

Figure 15: Просмотр строк не основного файла

Для установки точки в файле “calculate.c” на строке 21 используем команды “list calculate.c:20,27” и “break 21”.(рис. 16)

```
(gdb) list calculate.c:20,27
20         scanf("%f",&SecondNumeral);
21         return(Numeral - SecondNumeral);
22     }
23     else if(strncmp(Operation, "*", 1) == 0)
24     {
25         printf("Множитель: ");
26         scanf("%f",&SecondNumeral);
27         return(Numeral * SecondNumeral);
(gdb) break 21
Breakpoint 1 at 0x5555555526a: file calculate.c, line 21.
(gdb) |
```

Figure 16: Установка точки

Чтобы вывести информацию об имеющихся точках останова используем команду “info breakpoint”. (рис. 17)

```
(gdb) info breakpoints
Num   Type             Disp Enb Address            What
1     breakpoint      keep y   0x000055555555526a in calculate at calculate.c:21
(gdb)
```

Figure 17: Информация о точках

Запустил программу внутри отладчика и убедимся, что программа остановилась в момент прохождения точки останова. (рис. 18)

```
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/k/i/kisyachinova/work/os/lab_prog/calcul
Число: 5
Операция (*,-,*,/,pow,sqrt,sin,cos,tan): -
Выводимое: 3

Breakpoint 1, Calculate (Numeral=5, Operation=0x7fffffffcd64 "-") at calculate.c:21
21      return(Numeral - SecondNumeral);
(gdb) backtrace
#0 Calculate (Numeral=5, Operation=0x7fffffffcd64 "-") at calculate.c:21
#1 0x000055555555555e in main () at main.c:17
(gdb) |
```

Figure 18: Остановка программы

Посмотрим, чему на этом этапе равно значение переменной Numeral, с помощью команды “print Numeral” и сравним его с результатом вывода на экран после использования команды “display Numeral”. Значения совпадают. (рис. 19)

```
(gdb) print Numeral
$1 = 5
(gdb) display Numeral
1: Numeral = 5
(gdb) █
```

Figure 19: Просмотр значения

Уберём точки останова с помощью команды “delete 1”.(рис. 20)

```
(gdb) info breakpoints
Num      Type             Disp Enb Address            What
1        breakpoint      keep y   0x00005555555526a in calculate at calculate.c:21
          breakpoint already hit 1 time
(gdb) delete 1
(gdb)
```

Figure 20: Удаление точки останова

7. С помощью утилиты splint проанализировала коды файлов calculate.c и main.c.(рис. 21)(рис. 22)

```
kisyachinova@dk6n58 ~/work/os/lab_prog $ splint calculate.c
Splint 3.1.2 --- 13 Jan 2021

calculate.h:6:37: 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:8:32: Function parameter Operation declared as manifest array (size
      constant is meaningless)
calculate.c: (in function Calculate)
calculate.c:14:3: 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:20:3: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:26:3: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:32:3: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:33:6: 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:36:18: Return value type double does not match declared type float:
      (HUGE_VAL)
  To allow all numeric types to match, use %realtypes.
calculate.c:44:3: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:45:9: Return value type double does not match declared type float:
      (HUGE_VAL) (Warning: -fixedformalarray)
```

Figure 21: Анализ первого файла

```
kisyachinova@dk6n58 ~/work/os/lab_prog $ splint main.c
Splint 3.1.2 --- 13 Jan 2021

calculate.h:6:37: 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:14:2: 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:16:2: Return value (type int) ignored: scanf("%s", Oper...

Finished checking --- 3 code warnings
kisyachinova@dk6n58 ~/work/os/lab_prog $
```

Figure 22: Анализ второго файла

Выяснилось, что в данных файлах присутствует функция чтения `scanf`, которая возвращает целое число (`int`). Но эти числа нигде не используются и не сохраняются. Утилита выводит предупреждение о том, что в файле `calculate.c` происходит сравнение вещественного числа с нулём. Помимо этого, возвращаемые значения (`double`) в функциях `pow`, `sqrt`, `sin`, `cos`, `tan` записываются в переменную типа `float`, что говорит нам 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.