

Презентация по лабораторной работе №13

Операционные системы

Сячинова Ксения Ивановна

07 апреля 2023

Российский университет дружбы народов, Москва, Россия

Цель работы

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”.

```
kisyachinova1@dk2n26 ~ $ mkdir -p ~/work/os/lab_prog  
kisyachinova1@dk2n26 ~ $
```

Рис. 1: Создание подкаталога

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

```
kisyachinova1@dk2n26 ~/work/os/lab_prog $ touch calculate.h calculate.c main.c
kisyachinova1@dk2n26 ~/work/os/lab_prog $ ls
calculate.c calculate.h main.c
kisyachinova1@dk2n26 ~/work/os/lab_prog $
```

Рис. 2: Создание подкаталога

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.

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

Рис. 3: Файл calculate.c

```

36     {
37         printf("Ошибка: деление на ноль! ");
38         return(HUGE_VAL);
39     }
40     else
41         return(Numeral / SecondNumeral);
42 }
43 else if(strncmp(Operation, "pow", 3) == 0)
44 {
45     printf("Степень: ");
46     scanf("%f",&SecondNumeral);
47     return(pow(Numeral, SecondNumeral));
48 }
49 else if(strncmp(Operation, "sqrt", 4) == 0)
50     return(sqrt(Numeral));
51 else if(strncmp(Operation, "sin", 3) == 0)
52     return(sin(Numeral));
53 else if(strncmp(Operation, "cos", 3) == 0)
54     return(cos(Numeral));
55 else if(strncmp(Operation, "tan", 3) == 0)
56     return(tan(Numeral));
57 else
58 {
59     printf("Неправильно введено действие ");
60     return(HUGE_VAL);
61 }
62

```

Рис. 4: Файл calculate.c

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

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

Рис. 5: Файл calculate.h

Основной файл main.c, реализующий интерфейс пользователя к калькулятору.

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

Рис. 6: Файл main.c

3. Далее выполним компиляцию программы посредством gcc.

```
kisyachinova1@dk2n26 ~/work/os/lab_prog $ gcc -c calculate.c
kisyachinova1@dk2n26 ~/work/os/lab_prog $ gcc -c main.c
kisyachinova1@dk2n26 ~/work/os/lab_prog $ gcc calculate.o main.o -o calcul -lm
```

Рис. 7: Компиляция файла

4. 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.

```
1 #
2 # Makefile
3 #
4
5 CC = gcc
6 CFLAGS =
7 LIBS = -ln
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
```

Рис. 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.

```
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
```

Рис. 9: Изменения

После выполняем компиляцию файлов.

```
kisyachinova1@dk2n26 ~/work/os/lab_prog $ make calculate.o
gcc -c calculate.c -g
kisyachinova1@dk2n26 ~/work/os/lab_prog $ make main.o
gcc -c main.c -g
kisyachinova1@dk2n26 ~/work/os/lab_prog $ make calcul
gcc calculate.o main.o -o calcul -lm
kisyachinova1@dk2n26 ~/work/os/lab_prog $
```

Рис. 10: Компиляция

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

```
kisyachinova1@dk2n26 ~/work/os/lab_prog $ gdb ./calcul
GNU gdb (Gentoo 12.1 vanilla) 12.1
Copyright (C) 2022 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. 11: Отладчик

Далее вводим команду “run” для запуска программы внутри отладчика.

```
(gdb) run
Starting program: /afs/dfw.sci.gfx.edu.ru/home/k/i/kisyachinova/work/os/lab_prog/calcul
[thread debugging using libthread_db enabled]
Using host libthread_db library "/usr/lib64/libthread_db.so.1".
Число: 8
Операция (*,-,*,/,pow,sqrt,sin,cos,tan): +
Второе слагаемое: 15
23.00
[inferior 1 (process 16716) exited normally]
```

Рис. 12: Запуск программы

Для постраничного просмотра исходного кода используем 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;
```

Рис. 13: Просмотр кода

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

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

Рис. 14: Просмотр строк

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

```
(gdb) list calculate.c:20,29
20      {
21          printf("Вчитаемое: ");
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);
(gdb) █
```

Рис. 15: Просмотр строк

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

```
(gdb) break 21
Breakpoint 1 at 0x55555555247: file calculate.c, line 21.
(gdb) █
```

Рис. 16: Установка точки

Чтобы вывести информацию об имеющихся точках останова используем команду “info breakpoint”

```
(gdb) info breakpoints
Num      Type             Disp Enb Address            What
1        breakpoint      keep y   0x000055555555247 in Calculate
at calculate.c:21
(gdb) 
```

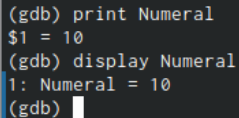
Рис. 17: Информация о точках

Запустим программу внутри отладчика и убедимся, что программа остановилась в момент прохождения точки останова.

```
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/k/i/kisyachinova1/work/os/lab_prog/calcul
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/usr/lib64/libthread_db.so.1".
Число: 10
Операция (*,-,*,/,pow,sqrt,sin,cos,tan): -
Breakpoint 3, Calculate (Numeral=10, Operation=0x7fffffffcf84 "-") at calculate.c:21
21      printf("Вычитаемое: ");
(gdb) █
```

Рис. 18: Остановка программы

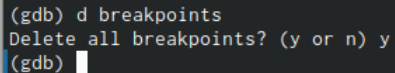
Посмотрим, чему на этом этапе равно значение переменной Numeral, с помощью команды “print Numeral” и сравним его с результатом вывода на экран после использования команды “display Numeral”. Значения совпадают.



```
(gdb) print Numeral
$1 = 10
(gdb) display Numeral
1: Numeral = 10
(gdb) 
```

Рис. 19: Просмотр значения

Уберём точки останова с помощью команды “d breakpoints”



```
(gdb) d breakpoints
Delete all breakpoints? (y or n) y
(gdb) 
```

Рис. 20: Удаление точки останова

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

```
kisyachinova1@dk2n26 ~/work/os/lab_prog $ splint calculate.c
Splint 3.1.2 --- 07 Dec 2021

calculate.h:7:37: 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:10:31: Function parameter Operation declared as manifest array
(size constant is meaningless)
calculate.c: (in function Calculate)
calculate.c:16:3: 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:22:3: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:28:3: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:34:3: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:35:6: 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:38:9: Return value type double does not match declared type float:
(HUGE_VAL)
    To allow all numeric types to match, use +relaxtypes.
```

Рис. 21: Анализ файла 1


```
Finished checking --- 3 code warnings
kisyachinova1@dk2n26 ~/work/os/lab_prog $ splint main.c
Splint 3.1.2 --- 07 Dec 2021

calculate.h:7:37: 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
```

Рис. 22: Анализ файла 2

Выводы

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.