

Презентация по лабораторной работе №13

Шмырин Михаил Сергеевич

Российский Университет Дружбы Народов

Цель работы

Цель данной лабораторной работы - приобретение простейших навыков разработки, анализа, тестирования и отладки приложений в ОС типа Linux на примере создания на языке программирования C калькулятора с простейшими функциями.

Выполнение лабораторной работы

1. В домашнем каталоге создал подкаталог `~/work/os/lab_prog` и в нем уже создал три файла `calculate.h`, `calculate.c`, `main.c` (рис. 1). 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.

```
nldhs@msshmlhrt ~ $ cd work
nldhs@msshmlhrt ~/work $ cd os
nldhs@msshmlhrt ~/work/os $ mkdir lab_prog
nldhs@msshmlhrt ~/work/os $ cd lab_prog
nldhs@msshmlhrt ~/work/os/lab_prog $ touch calculate.h calculate.c main.c
nldhs@msshmlhrt ~/work/os/lab_prog $ ls
calculate.c calculate.h main.c
```

Рис. 1: Создание каталогов и файлов

2. В созданных файлах написал программы для работы калькулятора, которые были предоставлены (рис. 2 , 3 , 4 , 5)

```
#include <stdio.h>
#include <math.h>
#include <string.h>
#include "calculate.h"

float
Calculate(float Numeral, char Operation[4])
{
    float SecondNumeral;
    if(strcmp(Operation, "+") == 0)
    {
        printf("Input characters: ");
        scanf("%f", &SecondNumeral);
        return(Numeral + SecondNumeral);
    }
    else if(strcmp(Operation, "-") == 0)
    {
        printf("Input characters: ");
        scanf("%f", &SecondNumeral);
        return(Numeral - SecondNumeral);
    }
    else if(strcmp(Operation, "*") == 0)
    {
        printf("Input characters: ");
        scanf("%f", &SecondNumeral);
        return(Numeral * SecondNumeral);
    }
    else if(strcmp(Operation, "/") == 0)
    {

```

Рис. 2: calculate.c

```
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. 3: calculate.c

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

Рис. 4: calculate.h


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

Рис. 5: main.c

3. Выполнил компиляцию программы посредством gcc и при необходимости исправил синтаксис (рис. 6)

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

Рис. 6: Компиляция

4. Создал Makefile (рис. 7) и ввел в него предложенное содержимое (рис. 8)

```
midhs@msshmlhrln ~/work/os/lab_prog $ touch Makefile
```

Рис. 7: Создал Makefile

```
#  
# 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
```

Рис. 8: Makefile

5. Далее исправил Makefile (рис. 9). В переменную CFLAGS добавил опцию -g необходимую для компиляции объектных файлов и их использования в программе отладчика GDB. Сделал так, что утилита компиляции выбирается с помощью переменной CC

```
#  
# Makefile  
#  
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)
```

Рис. 9: Makefile исправленный

После этого удалил исполняемые файлы (make clean) и выполнил компиляцию файлов, используя команды make calculate.o, make main.o, make calcul (рис. 10)

```
midhs@msshmihrin ~/work/os/lab_prog $ make clean
rm calcul *.o *~
rm: невозможно удалить 'calcul': Нет такого файла или каталога
make: [Makefile:19: clean] Ошибка 1 (игнорирование)
midhs@msshmihrin ~/work/os/lab_prog $ make calcul
gcc -c calculate.c -g
gcc -c main.c -g
gcc calculate.o main.o -o calcul -lm
midhs@msshmihrin ~/work/os/lab_prog $ ls
calcul calculate.c calculate.h calculate.o main.c main.o Makefile
```

Рис. 10: clean and make

6. Далее с помощью команды `gdb ./calcul` запустил отладку программы (рис. 11)

```
mids@msshmihrin ~/work/os/lab_prog $ gdb ./calcul
GNU gdb (Gentoo 11.2 vanilla) 11.2
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...
```

Рис. 11: отладка

Для запуска программы внутри отладчика ввел команду run (рис. 12)

```
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/k/v/kvpodjhyarova/work/os/lab_prog/calcul
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число:
10
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): *
Множитель: 5
50.00
[Inferior 1 (process 7982) exited normally]
```

Рис. 12: run

Для постраничного (по 9 строк) просмотра исходного кода использовал команду list (рис. 13)

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

Рис. 13: list

Для просмотра строк с 12 по 15 основного файла использовал list с параметрами (рис. 14)

```
(gdb) list 12,15
12     printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");
13     scanf("%s",Operation);
14     Result = Calculate(Numeral, Operation);
15     printf("%6.2f\n",Result);
```

Рис. 14: list с параметрами

Для просмотра определенных строк не основного файла использовал list с параметрами (рис. 15)

```
no source file named calculate.c.  
(gdb) list calculate.c:20,25  
20     return(Numeral - SecondNumeral);  
21     }  
22     else if(strncmp(Operation, "*", 1) == 0)  
23     {  
24         printf("Множитель: ");  
25         scanf("%f",&SecondNumeral);
```

Рис. 15: 15

Установил точку останова в файле calculate.c на строке 18 и вывел информацию об имеющихся в проекте точках (рис. 16)

```
(gdb) list calculate.c:15,20
15     }
16     else if(strncmp(Operation, "-", 1) == 0)
17     {
18         printf("Вычитаемое: ");
19         scanf("%f",&SecondNumeral);
20         return(Numeral - SecondNumeral);
(gdb) break 18
Breakpoint 1 at 0x55555555247: file calculate.c, line 18.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/k/v/kvpodjhyarova/work/os/lab_prog/calcul
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 12
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): -

Breakpoint 1, Calculate (Numeral=12, Operation=0x7fffffffccbb4 "-") at calculate.c:18
18     printf("Вычитаемое: ");
```

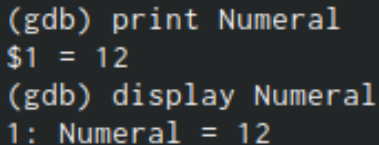
Рис. 16: точки останова

Ввел команду `backtrace` которая показал весь стек вызываемых функций от начал программы до текущего места (рис. 17)

```
(gdb) print(&backtrace)
(gdb) backtrace
#0 Calculate (Numeral=12, Operation=0x7fffffffccb4 "-") at calculate.c:18
#1 0x00005555555555a5 in main () at main.c:14
(gdb) print Numeral
```

Рис. 17: backtrace

Посмотрел чему равно на этом этапе значение переменной Numeral введя команду `print Numeral` и сравнил с результатом команды `display Numeral` (рис. 18)

A screenshot of a GDB terminal window with a dark background. The text is displayed in a monospaced font with syntax highlighting: the prompt '(gdb)' is in light blue, and the commands and their outputs are in yellow. The first command is 'print Numeral', which returns '\$1 = 12'. The second command is 'display Numeral', which returns '1: Numeral = 12'.

```
(gdb) print Numeral
$1 = 12
(gdb) display Numeral
1: Numeral = 12
```

Рис. 18: Numeral

Убрал точки останова (рис. 19)

```
(gdb) info breakpoints
Num      Type          Disp Enb Address              What
1        breakpoint    keep y   0x000055555555247 in calculate at calculate.c:18
          breakpoint already hit 1 time
(gdb) delete 1 2
No breakpoint number 2.
(gdb) info breakpoints
No breakpoints or watchpoints.
(gdb) continue
```

Рис. 19: Убрал точки останова

7. С помощью утилиты splint проанализировал коды файлов calculate.c main.c . Воспользовалсь командлй splint calculate.c и splint main.c (рис. 21) (рис. 20). С помощью этой команды выяснилось, что в файлах присутствует функция чтения, возвращающая целое число, но эти числа не используются и нигде не сохраняются. Утилита вывел предупреждение о том, что в файле происходит сравнение вещественного числа с нулем. Также возвращаемые значения в фугкциях записываются в переменную, что свидетельствует о потери данных

```
oldring@oldring:~/work/na/lab_prog $ splint main.c
Splint 3.1.2 --- 13 Jan 2021

calculate.c:4:27: Function parameter Operation declared as manifest array (also
      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 -fignoreformalarray to inhibit warning)
main.c: (in function main)
main.c:11:1: Return value (type int) ignored: scanf("%d", &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:13:1: Return value (type int) ignored: scanf("%s", &str,...
finished checking --- 3 code warnings
```

Рис. 20: splint main.c

```

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

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:7:31: Function parameter Operation declared as manifest array (size
        constant is meaningless)
calculate.c: (in function Calculate)
calculate.c:13:1: 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:19:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:25:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:31:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:32:4: 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:35:7: Return value type double does not match declared type float:
        (HUGE_VAL)
    To allow all numeric types to match, use +relaxtypes.
calculate.c:43:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:44:7: Return value type double does not match declared type float:
        (pow(Numeral, SecondNumeral))
calculate.c:47:7: Return value type double does not match declared type float:
        (sqrt(Numeral))
calculate.c:49:7: Return value type double does not match declared type float:
        (sin(Numeral))
calculate.c:51:7: Return value type double does not match declared type float:

```

Рис. 21: splint calculate.c

Выводы

В ходе данной лабораторной работы я приобрел навыки разработки, анализа, тестирования и отладки приложений в ОС на примере создания на языке программирования С калькулятора с простейшими функциями.