Презентация по лабораторной работе

Elizaveta Savchenko

03.06.2021

[1]RUDN University, Moscow, Russian Federation

Цель работы

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.

Выполнение лабораторной работы

Выполнение лабораторной работы

1. В домашнем каталоге создала подкаталог ~/work/os/lab_prog.

```
ensavchenko@dk8n62 ~ $ mkdir work
mkdir: невозможно создать каталог «work»: Файл существует
ensavchenko@dk8n62 ~ $ cd work
ensavchenko@dk8n62 ~/work $ mkdir os
mkdir: невозможно создать каталог «os»: Файл существует
ensavchenko@dk8n62 ~/work $ cd ~/work/os
ensavchenko@dk8n62 ~/work/os $ mkdir lab_prog
ensavchenko@dk8n62 ~/work/os $ cd ~/work/os/lab_prog
```

2. Создала в нём файлы: calculate.h, calculate.c, main.c. Это примитивнейший калькулятор, способный складывать, вычитать, умножать, делить, возводить число в степень, вычислять квадратный корень, вычислять sin, cos, tan. При запуске он запрашивает первое число, операцию, второе число. После этого программа выводит результат и останавливается.

```
ensavchenko@dk8n62 ~/work/os/lab_prog $ touch calculate.h
ensavchenko@dk8n62 ~/work/os/lab_prog $ touch calculate.c
ensavchenko@dk8n62 ~/work/os/lab_prog $ touch main.c
ensavchenko@dk8n62 ~/work/os/lab_prog $ ls
calculate.c  calculate.h  main.c
```

Реализация функций калькулятора в файле calculate.h:

```c
#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("Second term:");
      scanf("%f" , &SecondNumeral);
      return(Numeral + SecondNumeral);
    }
  else if(strncmp(Operation, "-", 1) ==0)
    {
      printf("Subtrahend:");
      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)
    {
```

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

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

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

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

File Edit Options Buffers Tools C Help

```c
#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;
}
```

3. Выполнила компиляцию программы посредством gcc:

```
ensavchenko@dk8n62 ~/work/os/lab_prog $ gcc -c calculate.c
ensavchenko@dk8n62 ~/work/os/lab_prog $ gcc -c main.c
main.c: В функции «main»:
main.c:12:12: предупреждение: формат «%s» ожидает аргумент типа «char *», но аргумент 2 имеет тип «char (*)
[4]» [-Wformat=]
   12 |   scanf ("%s",&Operation);
      |           ~^  ~~~~~~~~~~
      |            |  |
      |            |  char (*)[4]
      |            char *
ensavchenko@dk8n62 ~/work/os/lab_prog $ gcc calculate.o main.o -o calcul -lm
```

4. Исправила синтаксические ошибки.
5. Создала Makefile

```
 2
 3 CC = gcc
 4 CFLAGS =
 5 LIBS = -lm
 6 calcul: calculate.o main.o
 7         gcc calculate.o main.o  -o calcul $ (LIBS)
 8 calculate.o: calculate.c calculate.h
 9              gcc -c calculate.c $(CFLAGS)
10 main.o: main.c calculate.h
11         gcc -c main.c $(CFLAGS)
12 clean:
13        -rm calcul *.o *~
```

В содержании файла указаны флаги компиляции, тип компилятора и файлы, которые должен собрать сборщик.

6. С помощью gdb выполнила отладку программы calcul (перед использованием gdb исправила Makefile): – запустите отладчик GDB, загрузив в него программу для отладки: gdb ./calcul – для запуска программы внутри отладчика ввела команду run

```
ensavchenko@dk8n62 ~/work/os/lab_prog $ gdb ./calcul
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/e/n/ensavchenko/work/os/lab_prog/calcul
Число: 3
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): *
Множитель: 4
 12.00
[Inferior 1 (process 10073) exited normally]
(gdb)
```

– для постраничного (по 9 строк) просмотра исходного код использовала команду list – для просмотра строк с 12 по 15 основного файла использовала list с параметрами: list 12,15

```
(gdb) list
1       #include <stdio.h>
2       #include <math.h>
3       #include <string.h>
4       #include "calculate.h"
5       float
6       Calculate(float Numeral, char Operation[4])
7       {
8         float SecondNumeral;
9         if(strncmp(Operation, "+", 1) == 0)
10          {
(gdb) list 12,15
12            scanf("%f", &SecondNumeral);
13            return(Numeral + SecondNumeral);
14          }
15        else if(strncmp(Operation, "-", 1) == 0)
(gdb)
```

– для просмотра определённых строк не основного файла использовала list с параметрами: list calculate.c:20,29 – установила точку останова в файле calculate.c на строке номер 21: list calculate.c:20,27 break 21 – вывела информацию об имеющихся в проекте точка останова: info breakpoints

```
(gdb) list calculate.c:20,29
20          }
21        else if(strncmp(Operation, "*", 1) == 0)
22          {
23            printf("Factor: ");
24            scanf("%f", &SecondNumeral);
25            return(Numeral * SecondNumeral);
26          }
27        else if(strncmp(Operation, "/", 1) == 0)
28          {
29            printf("Divisor: ");
(gdb) list calculate.c:20,27
20          }
21        else if(strncmp(Operation, "*", 1) == 0)
22          {
23            printf("Factor: ");
24            scanf("%f", &SecondNumeral);
25            return(Numeral * SecondNumeral);
26          }
27        else if(strncmp(Operation, "/", 1) == 0)
(gdb) break 21
Breakpoint 1 at 0x1319: file calculate.c, line 21.
(gdb) info breakpoints
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x0000000000001319 in Calculate at calculate.c:21
(gdb)
```

– запустила программу внутри отладчика и убедилась, что программа остановится в момент прохождения точки останова – отладчик выдал следующую информацию, а команда backtrace показала весь стек вызываемых функций от начала программы до текущего места: – посмотрела, чему равно на этом этапе значение переменной Numeral, введя: print Numeral display Numeral – убрала точки останова: info breakpoints delete 1

```
(gdb) run
Starting program: /home/pdarzhankina/work/os/lab_prog/calcul
Numeral: 7
Operation (+,-,*,/,pow,sqrt,sin,cos,tan): pow

Breakpoint 1, Calculate (Numeral=7, Operation=0x7fffffffde14 "pow") at calculate.c:21
21        else if(strncmp(Operation, "*", 1) == 0)
(gdb) backtrace
#0  Calculate (Numeral=7, Operation=0x7fffffffde14 "pow") at calculate.c:21
#1  0x000055555555555bd in main ()
(gdb) print Numeral
$1 = 7
(gdb) display Numeral
1: Numeral = 7
(gdb) info breakpoints
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x0000555555555319 in Calculate at calculate.c:21
        breakpoint already hit 1 time
(gdb) delete 1
(gdb)
```

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

```
ensavchenko@dk8n62 ~/work/os/lab_prog $ splint calculate.c
Splint 3.1.2 --- 13 Jan 2021

calculate.h:3:37: 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:6:31: 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))
```

```
ensavchenko@dk8n62 ~/work/os/lab_prog $ splint main.c
Splint 3.1.2 --- 13 Jan 2021

calculate.h:3:37: 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: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:12:15: 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:12:12: Corresponding format code
main.c:12:3: Return value (type int) ignored: scanf("%s", &Ope...

Finished checking --- 4 code warnings
```

Выводы

Выводы

Я приобрела простейшие навыки разработки, анализа, тестирования и отладки приложений в ОС типа UNIX/Linux на примере создания на языке программирования C калькулятора с простейшими функциями.