

Лабораторная работа №14

Средства, применяемые при разработке программного обеспечения в ОС типа UNIX/Linux.

Кузнецов Юрий Владимирович

Содержание

Цель работы.....	1
Задание	1
Выполнение лабораторной работы.....	1
Выводы.....	6

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.

Задание

1. Создать каталог `~/work/os/lab_prog`.
2. Создать в нём файлы: `calculate.h`, `calculate.c`, `main.c`.
3. Выполнить компиляцию.
4. Создать `makefile`.
5. Выполнить отладку программы `calcul`.
6. Проанализировать коды файлов `calculate.c` и `main.c`.

Выполнение лабораторной работы

Ознакомился с лабораторной работой №14 и приступил к выполнению заданий.

В домашнем каталоге создал подкаталог `~/work/os/lab_prog`.

В каталоге создал файлы: `calculate.h`, `calculate.c`, `main.c`. Это нужно для примитивнейшего калькулятора, способного 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 должна будет вывести результат и остановится.

Ниже показаны команды запуска редактора emacs (рис. 1). Там же и на (рис. 2) показана компиляция программ с помощью gcc (опция -c создает объектные файлы, -o задает имена файлам).

```
emacs calculate.h
emacs calculate.c
emacs main.c
gcc -c calculate.c
gcc -c main.c
```

Figure 1: Команды

```
gcc calculate.o main.o -o calcul -lm
emacs Makefile
```

Figure 2: Команды

В файл main.c записал этот код (рис. 3):

```
1 //////////////////////////////////////////////////
2 // main.c
3 #include <stdio.h>
4 #include "calculate.h"
5
6 int
7 main (void)
8 {
9     float Numeral;
10    char Operation[4];
11    float Result;
12    printf("Число: ");
13    scanf("%f",&Numeral);
14    printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");
15    scanf("%s",&Operation);
16    Result = Calculate(Numeral, Operation);
17    printf("%.2f\n",Result);
18    return 0;
19 }
```

Figure 3: main.c

В файл calculate.c записал этот (рис. 4):

```
1 //////////////////////////////////////////////////
2 // calculate.c
3 #include <stdio.h>
4 #include <math.h>
5 #include <string.h>
6 #include "calculate.h"
7
8 float
9 Calculate(float Numeral, char Operation<3>)
10 {
11     float SecondNumeral;
12     if(strcmp(Operation,"+")==0)
13     {
14         printf("Второе слагаемое: ");
15         scanf("%f",&SecondNumeral);
16         return(Numeral+SecondNumeral);
17     }
18     else if(strcmp(Operation,"-")==0)
19     {
20         printf("Вычитаемое: ");
21         scanf("%f",&SecondNumeral);
22         return(Numeral-SecondNumeral);
23     }
24     else if(strcmp(Operation,"*")==0)
25     {
26         printf("Умножитель: ");
27         scanf("%f",&SecondNumeral);
28         return(Numeral*SecondNumeral);
29     }
30     else if(strcmp(Operation,"/")==0)
31     {
32         printf("Делитель: ");
33         scanf("%f",&SecondNumeral);
34         if(SecondNumeral==0)
35         {
36             printf("Ошибка: деление на ноль! ");
37             return(MGE_VAL);
38         }
39         else return(Numeral/SecondNumeral);
40     }
41     else if(strcmp(Operation,"pow",1)==0)
42     {
43         printf("Степень: ");
44         scanf("%f",&SecondNumeral);
45         return(pow(Numeral, SecondNumeral));
```

Figure 4: calculate.c

Содержимое файла calculate.h (рис. 5):

```

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

```

Figure 5: calculate.h

Создал Makefile (рис. 6):

```

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

```

Figure 6: makefile

С помощью gdb выполнил отладку программы calcul. Чтобы запустить отладчик пишу 'gdb ./calcul'. Для запуска калькулятора внутри отладчика пишу 'run'.

```

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...
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/i/a/iaakopyan/work/os/lab_prog/calcul
Число: 9
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): -
Вычитаемое: 2
    7.00
[Inferior 1 (process 4600) exited normally]
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/i/a/iaakopyan/work/os/lab_prog/calcul
Число: 6
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): +
Второе слагаемое: 54
    60.00

```

Figure 7: GDB

Чтобы просмотреть первые 10 строк кода пишу 'list'. Или же использую лист с параметрами. Для точки останова пишу 'break'. (рис. 8)

```
[Inferior 1 (process 4605) exited normally]
(gdb) list
1      //////////////////////////////////////
2      // main.c
3      #include <stdio.h>
4      #include "calculate.h"
5
6      int
7      main (void)
8      {
9          float Numeral;
10         char Operation[4];
(gdb) list 12,15
12         printf("Число: ");
13         scanf("%f",&Numeral);
14         printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");
15         scanf("%s",&Operation);
(gdb) list calculate.c:20,29
20         printf("Вычитаемое: ");
21         scanf("%f",&SecondNumeral);
22         return(Numeral-SecondNumeral);
23     }
24     else if(strncmp(Operation,"*",1)==0)
25     {
26         printf("Множитель: ");
27         scanf("%f",&SecondNumeral);
28         return(Numeral*SecondNumeral);
29     }
(gdb) list calculate.c:20,27
20         printf("Вычитаемое: ");
21         scanf("%f",&SecondNumeral);
22         return(Numeral-SecondNumeral);
23     }
24     else if(strncmp(Operation,"*",1)==0)
25     {
26         printf("Множитель: ");
27         scanf("%f",&SecondNumeral);
(gdb) break 21
Breakpoint 1 at 0x555555400966: file calculate.c, line 21.
(gdb) info breakpoints
Num      Type           Disp Enb Address                  What
1        breakpoint     keep y   0x0000555555400966 in Calculate at calculate.c:21
(gdb) run
```

Figure 8: GDB

Запустил программу внутри отладчика, программа остановилась в момент прохождения точки останова. Отладчик выдал следующую информацию:

```
'#0 Calculate (Numeral=5, Operation=0x7fffffff14 "-") at calculate.c:21'
'#1 0x000000000400c31 in main () at main.c:16'
```

Я вызвал команду `backtrace` и смог вывести весь стек вызываемых функций от начала программы до текущего места.

Посмотрел, чему равно на этом этапе значение переменной `Numeral`, введя:

```
print Numeral
```

Было выведено число 5.

Сравнил с результатом вывода на экран после использования команды:

display Numeral

Удалил точку останова (delete 1).

```
Starting program: /afs/.dk.sci.pfu.edu.ru/home/i/a/iaakopyan/work/os/lab_prog/calcul
Число: 5
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): -

Breakpoint 1, Calculate (Numeral=5, Operation=0x7fffffff14 "-") at calculate.c:21
21      scanf("%f",&SecondNumeral);
(gdb) backtrace
#0 Calculate (Numeral=5, Operation=0x7fffffff14 "-") at calculate.c:21
#1 0x000055555400c31 in main () at main.c:16
(gdb) print Numeral
$1 = 5
(gdb) display Numeral
1: Numeral = 5
(gdb) info breakpoints
Num  Type      Disp Enb Address          What
1     breakpoint keep y  0x000055555400966 in Calculate at calculate.c:21
      breakpoint already hit 1 time
(gdb) delete 1
(gdb) q
A debugging session is active.
```

Inferior 1 [process 4677] will be killed.

Quit anyway? (y or n) y

Splint 3.1.2 --- 13 Jan 2021

```
calculate.h:5:37: 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:9:31: Function parameter Operation declared as manifest array (size
      constant is meaningless)
calculate.c: (in function Calculate)
calculate.c:15:7: 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:21:7: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:27:7: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:33:7: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:34:10: 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
```

Figure 9: GDB u splint

С помощью утилиты splint попробовала проанализировать коды файлов calculate.c и main.c (рис. 9) (рис. 10). Информацию о splint нашла здесь: [ссылка 2](#)

Splint- это UNIX программа, позволяющая проводить статический анализ кода, написанного на Си.

Splint особенно хорош при проверке типов назначений переменных и функций, эффективности, неиспользуемых переменных и идентификаторов функций, недоступного кода и возможных утечек памяти.

```

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:37:10: Return value type double does not match declared type float:
(HUGE_VAL)
To allow all numeric types to match, use +relaxtypes.
calculate.c:44:7: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:45:13: Return value type double does not match declared type float:
(pow(Numeral, SecondNumeral))
calculate.c:48:11: Return value type double does not match declared type float:
(sqrt(Numeral))
calculate.c:50:11: Return value type double does not match declared type float:
(sin(Numeral))
calculate.c:52:11: Return value type double does not match declared type float:
(cos(Numeral))
calculate.c:54:11: Return value type double does not match declared type float:
(tan(Numeral))
calculate.c:58:11: Return value type double does not match declared type float:
(HUGE_VAL)

Finished checking --- 15 code warnings

Splint 3.1.2 --- 13 Jan 2021

calculate.h:5:37: 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:13:3: 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:15:14: 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:15:11: Corresponding format code
main.c:15:3: Return value (type int) ignored: scanf("%s", &Ope...

Finished checking --- 4 code warnings _

```

Figure 10: splint

Выводы

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