

Лабораторная работа №13. Средства, применяемые при разработке программного обеспечения в ОС типа UNIX/Linux

Латыпова Диана. НФИбд-02-21

¹RUDN University, Moscow, Russian Federation

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.

Задание

1. В домашнем каталоге создайте подкаталог `~/work/os/lab_prog`.
2. Создайте в нём файлы: `calculate.h`, `calculate.c`, `main.c`.

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.

3. Выполните компиляцию программы посредством `gcc`.
4. При необходимости исправьте синтаксические ошибки.

5. Создайте Makefile с содержанием...
6. С помощью gdb выполните отладку программы calcul (перед использованием gdb исправьте Makefile).
7. С помощью утилиты splint попробуйте проанализировать коды файлов calculate.c и main.c.

Выполнение лабораторной работы

Я создала в домашнем каталоге каталог lab_prog:

```
mkdir ~/work/os/lab_prog
```

В созданном каталоге создала еще 3 файла: calculate.h, calculate.c, main.c:

```
1 touch calculate.h
```

```
2 touch calculate.c
```

```
3 touch main.c
```


После чего открыла каждый файл в редакторе emacs и вписала содержимое(рис. 1):

```
[dlatihpova@fedora lab_prog]$ emacs calculate.h  
[dlatihpova@fedora lab_prog]$ emacs calculate.c  
[dlatihpova@fedora lab_prog]$ emacs main.c
```

Figure 1: Emacs

Листинг файла calculate.c

```
////////////////////////////////////  
// calculate.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("Второе слагаемое: ");
    scanf("%f",&SecondNumeral);
    return(Numeral + SecondNumeral);
}
else if(strncmp(Operation, "-", 1) == 0)
{
    printf("Вычитаемое: ");
    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)
    {
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);
}
}
```

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

```
#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;  
}
```

Далее я выполнила компиляцию программы посредством gcc:

```
1 gcc -c calculate.c
```

```
2 gcc -c -g main.c
```

```
3 gcc calculate.o main.o -o calcul -lm
```

И посмотрели, что создался исполняемый файл calcul(рис. 2):

```
[dlatihpova@fedora lab_prog]$ ls  
calcul      calculate.c~ calculate.h~ main.c      main.o  
calculate.c calculate.h  calculate.o main.c~     Makefile~
```

Figure 2: ls

Листинг файла 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 *~
```

Листинг исправленного файла Makefile

```
CC = gcc
CFLAGS = -g
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
```

С помощью gdb выполнила отладку программы calcul:

```
gdb ./calcul
```

Запуск программы внутри отладчика

Запустила внутри отладчика программу(рис. 3):

run

```
(gdb) run
Starting program: /home/dlatihpova/work/os/lab_prog/calcul

This GDB supports auto-downloading debuginfo from the following URLs:
https://debuginfod.fedoraproject.org/
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 9
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): *
Множитель: 3
27.00
```

Figure 3: Запуск программы внутри отладчика

Сначала постранично просмотрела код, затем с 12 по 15 строку(рис. 4):

1 list

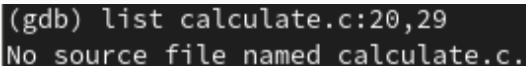
2 list 12,15

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

Figure 4: list

Затем попробовала посмотреть определённые строки не основного файла. Но не вышло(рис. 5):

```
list calculate.c:20,29
```

A screenshot of a terminal window with a dark background. The text is white and shows a GDB command being entered and the resulting error message.

```
(gdb) list calculate.c:20,29  
No source file named calculate.c.
```

Figure 5: Просмотр определённых строк не основного файла

Попробовала поставить точку останова на строке номер 21, однако такой строки нет:

`break 21`

Поэтому поставила точку останова на 16 строке(рис. 6):

`break 16`

```
(gdb) break 16
Breakpoint 2 at 0x4014c2: file main.c, line 16.
(gdb) info breakpoints
Num   Type             Disp Enb Address            What
1     breakpoint       keep y   <PENDING>          21
2     breakpoint       keep y   0x00000000004014c2 in main at main.c:16
```

Figure 6: Точка останова на 16 строке

Запустила программу внутри отладчика и убедилась, что программа останавливается в момент прохождения точки останова. Команда `backtrace` показала нам весь стек вызываемых функций от начала программы до текущего места(рис. 7):

Запуск программы внутри отладчика

```
(gdb) run
Starting program: /home/dlatihpova/work/os/lab_prog/calcul
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 5

Breakpoint 2, main () at main.c:16
16      scanf("%s",&operation);
(gdb) backtrace
#0  main () at main.c:16
```

Figure 7: Проверка точки останова

Далее посмотрела, чему равно на этом этапе значение переменной Numeral и сравнила с результатом вывода на экран после использования команды(рис. 8):

1 `print Numeral`

2 `display Numeral`

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

Figure 8: Numeral

Удалила точки останова(рис. 9):

1 info breakpoints

2 delete 1

3 delete 2

4 info breakpoints

```
(gdb) info breakpoints
Num      Type             Disp Enb Address                  What
1        breakpoint      keep y   <PENDING>                21
2        breakpoint      keep y   0x0000000000004014c2 in main at main.c:16
          breakpoint already hit 1 time
(gdb) delete 1
(gdb) delete 2
(gdb) info breakpoints
No breakpoints or watchpoints.
```

Figure 9: Удаление точек останова

И наконец, с помощью утилиты `splint` проанализировала коды файлов `calculate.c` и `main.c`(рис. 10)(рис. 11):

1 `splint calculate.c`

2 `splint main.c`

```
[dlatihpova@fedora lab_prog]$ splint calculate.c
Splint 3.1.2 --- 23 Jul 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:16:7: 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:22:7: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:28:7: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:34:7: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:35:10: 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
```

Figure 10: Анализ кода файла calculate.c

```
[dlatihpova@fedora lab_prog]$ splint main.c
Splint 3.1.2 --- 23 Jul 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:14:3: 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:16:14: 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:16:11: Corresponding format code
main.c:16:3: Return value (type int) ignored: scanf("%s", &Ope...

Finished checking --- 4 code warnings
[dlatihpova@fedora lab_prog]$
```

Figure 11: Анализ кода файла main.c

Выводы

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.