Отчет по лабораторной работе №13.

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

Данила Андреевич Стариков

Содержание

1 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.

2 Выполнение лабораторной работы

1. В домашнем каталоге создали подкаталог `~/work/os/lab_prog`, в котором будут храниться файлы с кодом программ.
2. Создали в нём файлы: `calculate.h` (Листинг 2.2), `calculate.c` (Листинг 2.3), `main.c` (Листинг 2.4). Это 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.

Листинг 2.1 Текст файла calculate.h

```
#ifndef CALCULATE_H_
#define CALCULATE_H_

float Calculate(float Numeral, char Operation[4]);

#endif /*CALCULATE_H_*/
```

3. Выполнили компиляцию программы посредством `gcc` (Рисунок 2.1) и проверили ее работу (Рисунок 2.2).

```
[dastarikov@fedora lab_prog]$ gcc -c calculate.c
[dastarikov@fedora lab_prog]$ gcc -c main.c
[dastarikov@fedora lab_prog]$ gcc calculate.o main.o -o calcul -lm
```

Рис. 2.1: Компиляция программы.

```
[dastarikov@fedora lab_prog]$ ./calcul
Число: 100
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): +
Второе слагаемое: 10
110.00
[dastarikov@fedora lab_prog]$ ./calcul
Число: 1
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): sqrt
  1.00
```

Рис. 2.2: Работа программы.

4. Для упрощения компиляции создали `Makefile` (Листинг 2.4) и запустили его, чтобы убедиться в работе (Рисунок 2.3)

```
[dastarikov@fedora lab_prog]$ make clean
rm calcul *.o *~
rm: невозможно удалить '*~': Нет такого файла или каталога
make: [makefile:14: clean] Error 1 (игнорирование)
[3]-  Завершён        gedit makefile
[dastarikov@fedora lab_prog]$ make
gcc -c calculate.c
gcc -c main.c
gcc calculate.o main.o -o calcul -lm
[dastarikov@fedora lab_prog]$ ./calcul
Число: 100
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): /
Делитель: 4
 25.00
```

Рис. 2.3: Проверка работоспособности Makefile.

5. Далее выполнили отладку программы `calcul` с помощью `gdb` (Рис. 2.4, 2.12):

```
[dastarikov@fedora lab_prog]$ gdb calcul
GNU gdb (GDB) Fedora Linux 13.1-3.fc38
Copyright (C) 2023 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-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
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. 2.4: Запуск отладчика.

```
(gdb) run
Starting program: /home/dastarikov/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]) n
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 100
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): +
Второе слагаемое: 10
110.00
[Inferior 1 (process 3734) exited normally]
Missing separate debuginfos, use: dnf debuginfo-install glibc-2.37-1.fc38.x86_64
(gdb)
```

Рис. 2.5: Запуск программе в среде отладки.

```
(gdb) list
1       #include <stdio.h>
2       #include "calculate.h"
3       int main (void)
4       {
5               float Numeral;
6               char Operation[4];
7               float Result;
8               printf("Число: ");
9               scanf("%f",&Numeral);
10              printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");
(gdb) list
11              scanf("%s",&Operation);
12              Result = Calculate(Numeral, Operation);
13              printf("%6.2f\n",Result);
14              return 0;
15      }
(gdb) list
Line number 16 out of range; main.c has 15 lines.
(gdb) list 12, 15
12              Result = Calculate(Numeral, Operation);
13              printf("%6.2f\n",Result);
14              return 0;
15      }
(gdb)
```

Рис. 2.6: Просмотр кода основного файла программы.

```
(gdb) list calculate.c:20,29
20              }
21              else if(strncmp(Operation, "*", 1) == 0)
22              {
23                      printf("Множитель: ");
24                      scanf("%f",&SecondNumeral);
25                      return(Numeral * SecondNumeral);
26              }
27              else if(strncmp(Operation, "/", 1) == 0)
28              {
29                      printf("Делитель: ");
(gdb)
```

Рис. 2.7: Просмотр кода файла calculate.c

```
(gdb) list calculate.c:20,27
20              }
21              else if(strncmp(Operation, "*", 1) == 0)
22              {
23                      printf("Множитель: ");
24                      scanf("%f",&SecondNumeral);
25                      return(Numeral * SecondNumeral);
26              }
27              else if(strncmp(Operation, "/", 1) == 0)
(gdb) break 21
Breakpoint 1 at 0x401247: file calculate.c, line 21.
(gdb)
```

Рис. 2.8: Просмотр кода файла calculate.c и установка точки останова.

```
(gdb) info breakpoints
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x0000000000401247 in Calculate at calculate.c:21
(gdb)
```

Рис. 2.9: Просмотр всех точек останова

```
(gdb) run
Starting program: /home/dastarikov/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): *

Breakpoint 1, Calculate (Numeral=10, Operation=0x7fffffffdeb4 "*") at calculate.c:21
21              else if(strncmp(Operation, "*", 1) == 0)
(gdb) backtrace
#0  Calculate (Numeral=10, Operation=0x7fffffffdeb4 "*") at calculate.c:21
#1  0x00000000004014eb in main () at main.c:12
(gdb)
```

Рис. 2.10: Запуск программы до точки останова.

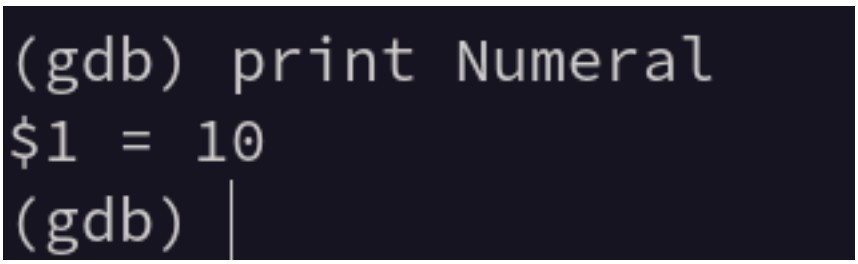

Рис. 2.11: Просмотр промежуточного значения переменной.

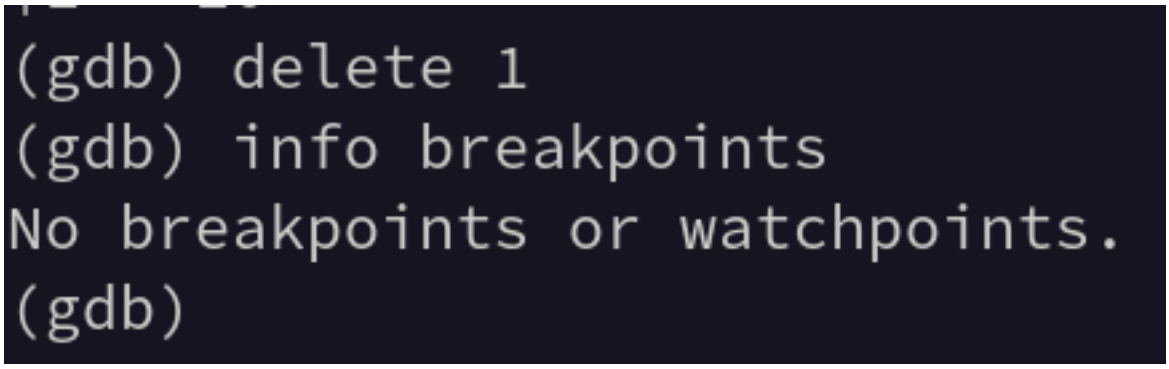

Рис. 2.12: Удаление точки останова.

6. С помощью утилиты `splint` проанализировали коды файлов `calculate.c` и `main.c` (Рис. 2.13, 2.14).

```
[dastarikov@fedora lab_prog]$ splint calculate.c
Splint 3.1.2 --- 21 Jan 2023

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:6:37: Function parameter Operation declared as manifest array (size
                    constant is meaningless)
calculate.c: (in function Calculate)
calculate.c:12:3: 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:18:3: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:24:3: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:30:3: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:31:6: 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: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:41:3: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:42:9: Return value type double does not match declared type float:
                    (pow(Numeral, SecondNumeral))
calculate.c:44:52: Return value type double does not match declared type float:
                     (sqrt(Numeral))
calculate.c:45:51: Return value type double does not match declared type float:
                     (sin(Numeral))
calculate.c:46:51: Return value type double does not match declared type float:
                     (cos(Numeral))
calculate.c:47:51: Return value type double does not match declared type float:
                     (tan(Numeral))
calculate.c:51:9: Return value type double does not match declared type float:
                    (HUGE_VAL)

Finished checking --- 15 code warnings
[dastarikov@fedora lab_prog]$
```

Рис. 2.13: Анализ calculate.c

```
[dastarikov@fedora lab_prog]$ splint main.c
Splint 3.1.2 --- 21 Jan 2023

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)
main.c: (in function main)
main.c:9:2: 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:11:13: 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:11:10: Corresponding format code
main.c:11:2: Return value (type int) ignored: scanf("%s", &Ope...

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

Рис. 2.14: Анализ main.c.

Листинг 2.2 Текст файла calculate.c

```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);
    }
}
```

Листинг 2.3 Текст файла main.c

```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;
}
```

Листинг 2.4 Текст Makefile-файла

```
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 -g -c calculate.c $(CFLAGS)

main.o: main.c calculate.h
        gcc -g -c main.c $(CFLAGS)
clean:
        -rm calcul *.o *~
```

3 Выводы

В рамках лабораторной работы получили практические навыки разработик, анализа, тестирования и отладки приложений в ОС типа UNIX/Linux на примере создания на языке программирования `C` калькулятора с простейшими функциями с помощью утилит `dgb` и `splint`.