Отчёт по лабораторной работе 13

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

Осокин Г.И

Содержание

Список иллюстраций

0.1 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.

1 Выполнение лабораторной работы

1.1 Написание программ

Создали директорию ~/work/os/lab_prog:

```
mkdir -p ~/work/os/lab_prog
```

Рис. 1.1: Создание директории

Написали программу main.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;
}
```

```
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;
}
```

Рис. 1.2: `main.c`

Написали программу `calculate.h`:

```
#ifndef CALCULATE_H_
#define CALCULATE_H_

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

#endif /*CALCULATE_H_*/
```

```
#ifndef CALCULATE_H_
#define CALCULATE_H_

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

#endif /*CALCULATE_H_*/
```

Рис. 1.3: calculate.h

Написали программу 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);
    }
}
```

```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);
    }
```

Рис. 1.4: `calculate.c`

Скомпилировали программы при помощи команд

```
gcc -c calculate.c
gcc -c main.c
```

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

Рис. 1.5: Компиляция программ

Для автоматизации сборки написали `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 *~
```

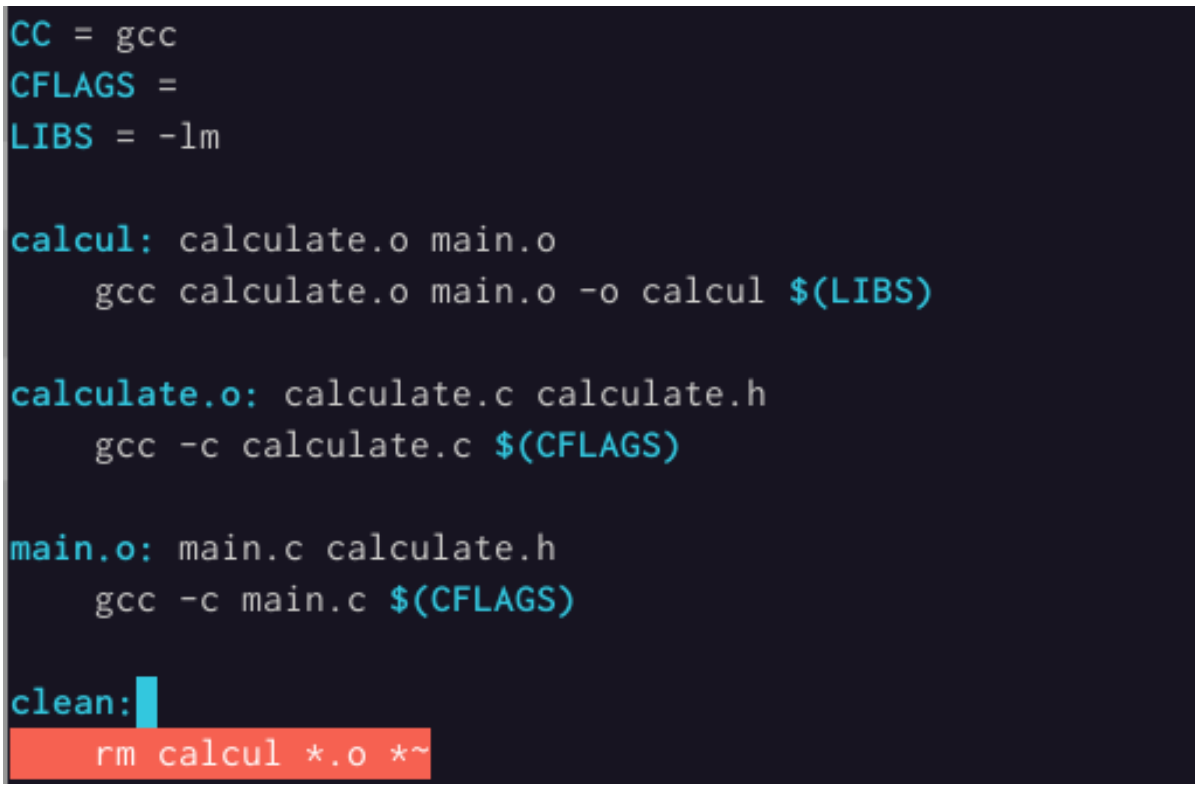

Рис. 1.6: `Makefile`

1.2 Отладка программы

Прежде чем начать отладку, запустили `Makefile`:

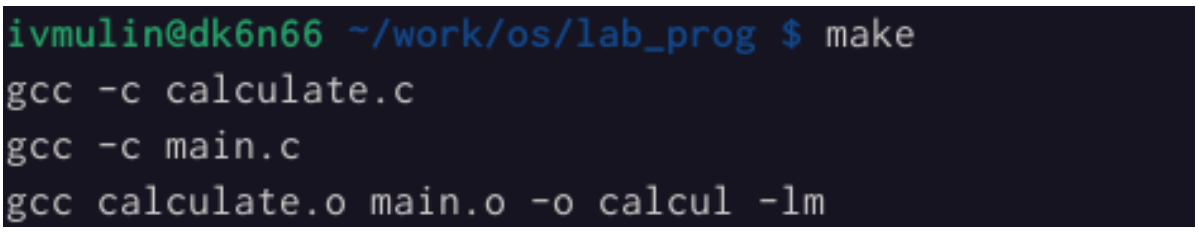

Рис. 1.7: Запуск `Makefile`

Запустили отладчик gdb и провели отладку кода:

```
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/i/v/ivmulin/work/os/lab_prog/calcul
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/usr/lib64/libthread_db.so.1".
Число: 2
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): +
Второе слагаемое: 1
  3.00
[Inferior 1 (process 18580) exited normally]
(gdb)
```

Рис. 1.8: Отладка кода

При помощи команды `splint` проанализировали код программ `calculate.c` и `main.c`:

```
ivmulin@dk6n66 ~/work/os/lab_prog $ splint calculate.c
Splint 3.1.2 --- 07 Dec 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: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:42:3: Return value (type int) ignored: scanf("%f", &Sec...
```

Рис. 1.9: Отладка кода программы `calculate.c`

```
ivmulin@dk6n66 ~/work/os/lab_prog $ splint main.c
Splint 3.1.2 --- 07 Dec 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)
main.c: (in function main)
main.c:10:5: 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:12:5: Return value (type int) ignored: scanf("%s", Oper...

Finished checking --- 3 code warnings
```

Рис. 1.10: Отладка кода программы `main.c`

1.3 Ответы на контрольные вопросы

1. Как получить информацию о возможностях программ gcc, make, gdb и др.?

Ответ: при помощи программы `man`.

2. Назовите и дайте краткую характеристику основным этапам разработки приложений в UNIX.

Ответ: 1. Выбор названия 2. Выбор языка программирования 3. Попытка выполнить работу всю зараз 4. Отрицание 5. Гнев 6. Торг 7. Депрессия 8. Принятие

3. Что такое суффикс в контексте языка программирования? Приведите примеры использования.

Ответ: финальная часть названия программы, обычно отделяемая точкой.

4. Каково основное назначение компилятора языка C в UNIX?

Ответ: компилятор языка C в UNIX в основном компилирует программы языка C в UNIX, написанные на языке C в UNIX.

5. Для чего предназначена утилита make?

Ответ: для того чтобы `делать`.

6. Приведите пример структуры Makefile. Дайте характеристику основным элементам этого файла.

Ответ: всё выполнил, как и просили.

7. Назовите основное свойство, присущее всем программам отладки. Что необходимо сделать, чтобы его можно было использовать?

Ответ: свойство - анализ кода; для анализа необходимо скомпилировать программу.

8. Назовите и дайте основную характеристику основным командам отладчика gdb.

Ответ: см. ответ к вопросу 6.

9. Опишите по шагам схему отладки программы, которую вы использовали при выполнении лабораторной работы.

Ответ: 1. Вначале я запустил `gdb` 2. Затем я его закрыл

10. Прокомментируйте реакцию компилятора на синтаксические ошибки в программе при его первом запуске.

Ответ: когда я увидел реакцию компилятора на синтаксические ошибки в программе при его первом запуске,я был возмущён, поражён, обескуражен, ошеломлён, фрустрирован и изумлён. Но использовал совершенно другие выражения.

11. Назовите основные средства, повышающие понимание исходного кода программы.

Ответ: здравый смысл и нормальный код.

12. Каковы основные задачи, решаемые программой splint?

Ответ: анализ кода.

1.4 Заключение

В ходе выполнения лабораторной работы были изучены простейшие навыки разработки, анализа, тестирования и отладки приложений в ОС типа UNIX/Linux на примере создания на языке программирования C калькулятора с простейшими функциями. Цель работы была достигнута.