Отчёт по лабораторной работе №13

Средства для создания приложений в ОС UNIX

Карими Мохаммад Валид НФИбд-02-21

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Вывод	13
4	Контрольные вопросы	14

List of Figures

2.1	Компиляция	9
2.2	Использование make	10
2.3	Использование отладчика	11
2.4	Использование отладчика	11
2.5	Использование отладчика	12
2.6	Использование splint	12

1 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.

2 Выполнение лабораторной работы

- 1. Создали подкаталог для файлов лаб работы
- 2. Создал 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.

Код файла calculate.c (реализует функции калькулятора)

```
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);
   }
}
 Код файла calculate.h (описывает формат вызова функции калькулятора)
// calculate.h
#ifndef CALCULATE H
#define CALCULATE H
float Calculate(float Numeral, char Operation[4]);
#endif /*CALCULATE H */
Код файла main.c (реализует интерфейс пользователя к калькулятору)
```

```
// 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;
}
```

3. Выполнили компиляцию программы посредством дсс:

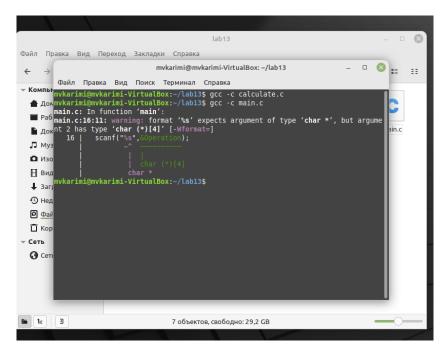


Figure 2.1: Компиляция

- 4. При необходимости исправили синтаксические ошибки.
- 5. Создали Makefile со следующим содержанием:

```
#
# 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 *~

End Makefile

С помощью программы make получаем различные варианты построения исполняемого модуля.

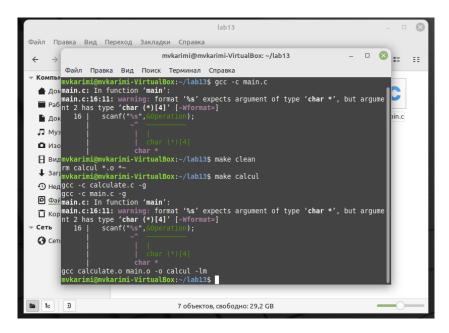


Figure 2.2: Использование make

4. С помощью gdb выполнил отладку программы calcul

Figure 2.3: Использование отладчика

```
Правка Вид Переход Закладки Справка
                                                                                                        □ ② :=
                                    mvkarimi@mvkarimi-VirtualBox: ~/lab13
ПББ 14 scanf("%f",&Numeral);

Дом15 printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");

[gdb) list calculate.c:20,29
Док<mark>21</mark>
22
                      printf("Вычитаемое: ");
scanf("%f",&SecondNumeral);
return(Numeral - SecondNumeral);
Му3<mark>23</mark>
                 else if(strncmp(Operation, "*", 1) == 0)
Вид 26
27
3arı
28
                       printf("Множитель: ");
                      scanf("%f",&SecondNumeral);
return(Numeral * SecondNumeral);
<sub>Нед</sub>29 re
(gdb) break 21
<u>Фай</u>Вгеакроint 1 at
Kop (gdb) run
Starting program: /home/mvkarimi/lab13/calcul
    Число: 7
Сети
     Breakpoint 1, Calculate (Numeral=7, Operation=0x7fffffffdec4 "-")
    21
(gdb)
                       printf("Вычитаемое: ");
```

Figure 2.4: Использование отладчика

```
lab13
Файл Правка Вид Переход Закладки Справка
                                                                                                                        □ ⊗ ::
                                              mvkarimi@mvkarimi-VirtualBox: ~/lab13
            Файл Правка Вид Поиск Терминал Справка
  <sup>Компы</sup> (gdb) break 21
<mark>▲ Дом</mark>Breakpoint 1 at (
  Дообтеанров 1 at (gdb) run

Pao Starting program: /home/mvkarimi/lab13/calcul

Дож
Число: 7
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): -
  Л Муз
  ☐ N30Breakpoint 1, Calculate (Numeral=7, Operation=8x7fffffffdec4 "-")
  at
⊞ Вид<mark>21</mark>
                               printf("Вычитаемое: ");
   ↓ 3ari#0 Calculate (
                                                  Operation=0x7ffffffffdec4 "-") at calculate.c:21
  (gdb) print Numeral
Δαμές = 7
 O Φaŭşî = 7

☐ Kopp (gdb) display Numeral
1: Numeral = 7

Cett (gdb) info breakpoints
3 Cett Num Type
1 breakpoint
                                            Disp Enb Address
keep y 0x000055
                       breakpoint already hit 1 time
i i: 3
                                                     7 объектов, свободно: 29,2 GB
```

Figure 2.5: Использование отладчика

5. С помощью утилиты splint попробовали проанализировать коды файлов

```
mvkarimi@mvkarimi-VirtualBox:~/lab13 — □ ②

Файл Правка Вид Поиск Терминал Справка

обрабатываются триггеры для man-db (2.9.1-1) ...

пототофmvkarimi-VirtualBox:/home/mvkarimi/lab13# exit
exit
mvkarimi@mvkarimi-VirtualBox:~/lab13$ splint calculate.c

px Splint 3.1.2 --- 20 Feb 2018

Узсаlculate.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)

IT calculate.c:10:31: Function parameter Operation declared as manifest array (size constant is meaningless)

calculate.c: (in function Calculate)

average (size constant is meaningless)

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:34:7: 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
```

Figure 2.6: Использование splint

3 Вывод

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

4 Контрольные вопросы

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

Ответ: Для этого есть команда man и предлагающиеся к ней файлы.

2. Назовите и дайте краткую характеристику основным этапам разработки

прило-жений в UNIX. Ответ: Кодировка, Компиляция, Тест.

3. Что такое суффикс в контексте языка программирования? Приведите при-

меры использования. Ответ: Это расширения файлов.

4. Каково основное назначение компилятора языка С в UNIX? Ответ: Програм-

ма дсс, которая интерпретирует к определенному языку программирования

аргументы командной строки и определяет запуск нужного компилятора

для нужного файла.

5. Для чего предназначена утилита make? Ответ: Для компиляции группы

файлов. Собрания из них программы, и последующего удаления.

6. Приведите пример структуры Makefile. Дайте характеристику основным

элементам этого файла. Ответ:

program: main.o lib.o

cc -o program main.o lib.o

main.o lib.o: defines.h

В имени второй цели указаны два файла и для этой же цели не указана ко-

манда компиляции. Кроме того, нигде явно неуказана зависимость объектных

14

файлов от «*.c»-файлов. Дело в том, что программа make имеет предопределённые правила для получения файлов с определёнными расширениями. Так, для цели-объектного файла (расширение «.o») при обнаружении соответствующего файла с расширением «.c» будет вызван компилятор «сс -с» с указанием в параметрах этого «.c»-файла и всех файлов-зависимостей.

- 7. Назовите основное свойство, присущее всем программам отладки. Что необходимо сделать, чтобы его можно было использовать? Ответ: Программы для отладки нужны для нахождения ошибок в программе. Для их использования надо скомпилировать программу таким образом, чтобы отладочная информация содержалась в конечном бинарном файле.
- 8. Назовите и дайте основную характеристику основным командам отладчика gdb. Ответ:

backtrace – выводит весь путь к текущей точке останова, то есть названия всех функций, начиная от main(); иными словами, выводит весь стек функций;

break – устанавливает точку останова; параметром может быть номер строки или название функции;

clear – удаляет все точки останова на текущем уровне стека (то есть в текущей функции);

continue – продолжает выполнение программы от текущей точки до конца; delete – удаляет точку останова или контрольное выражение;

display – добавляет выражение в список выражений, значения которых отображаются каждый раз при остановке программы;

finish – выполняет программу до выхода из текущей функции; отображает возвращаемое значение, если такое имеется;

info breakpoints – выводит список всех имеющихся точек останова;

info watchpoints – выводит список всех имеющихся контрольных выражений;

list – выводит исходный код; в качестве параметра передаются название файла исходного кода, затем, через двоеточие, номер начальной и конечной строки;

next – пошаговое выполнение программы, но, в отличие от команды step, не выполняет пошагово вызываемые функции;

print – выводит значение какого-либо выражения (выражение передаётся в качестве параметра);

run – запускает программу на выполнение;

set - устанавливает новое значение переменной

step - пошаговое выполнение программы;

watch – устанавливает контрольное выражение, программа остановится, как только значение контрольного выражения изменится;

- 9. Опишите по шагам схему отладки программы, которую Вы использовали при выполнении лабораторной работы. Ответ:
- 10. gdb –silent ./calcul
- 11. run
- 12. list
- 13. backtrace
- 14. breakpoints
- 15. print Numeral
- 16. Splint (Не использовался по причине отсутствия команды в консоли).
- 17. Прокомментируйте реакцию компилятора на синтаксические ошибки в программе при его первом запуске. Ответ: Консоль выводит ошибку с номером строки и ошибочным сегментом, но при этом есть возможность выполнить программу сразу.

- 18. Назовите основные средства, повышающие понимание исходного кода программы. Ответ:
 - а) Правильный синтаксис
 - b) Наличие комментариев
 - с) Разбиение большой сложной программы на несколько сегментов попроще.
- 12. Каковы основные задачи, решаемые программой splint? Ответ: split разбиение файла на меньшие, определённого размера. Может разбивать текстовые файлы по строкам и любые по байтам. По умолчанию читает со стандартного ввода и создает файлы с именами вида хаа, хаb и т.д. По умолчанию разбиение идёт по 1000 строк в файле.