Лабораторная работа 13

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

Korshunova Polina

2023.17 апреля

RUDN 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 C 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 со заданным содержанием. Поясните в отчёте его содержание.

Задание

6. С помощью gdb выполните отладку программы calcul (перед использованием gdb исправьте Makefile):

– Запустите отладчик GDB, загрузив в него программу для отладки:

– Для запуска программы внутри отладчика введите команду run.

– Для постраничного (по 9 строк) просмотра исходного код используйте команду list.

– Для просмотра строк с 12 по 15 основного файла используйте list с параметрами.

Задание

- Для просмотра определённых строк не основного файла используйте list с параметрами.

– Установите точку останова в файле calculate.c на строке номер 21.

– Выведите информацию об имеющихся в проекте точка останова.

– Запустите программу внутри отладчика и убедитесь, что программа остановится в момент прохождения точки останова.

Задание

– Посмотрите, чему равно на этом этапе значение переменной Numeral, введя print Numeral.

– Сравните с результатом вывода на экран после использования команды: display Numeral.

– Уберите точки останова.

7. С помощью утилиты splint попробуйте проанализировать коды файлов calculate.c и main.c.

Процесс разработки программного обеспечения обычно разделяется на следующие этапы:

– планирование, включающее сбор и анализ требований к функционалу и другим характеристикам разрабатываемого приложения;

– проектирование, включающее в себя разработку базовых алгоритмов и спецификаций, определение языка программирования;

– непосредственная разработка приложения:

– кодирование — по сути создание исходного текста программы (возможно в нескольких вариантах);

– анализ разработанного кода;

– сборка, компиляция и разработка исполняемого модуля;

– тестирование и отладка, сохранение произведённых изменений;

– документирование.

Для создания исходного текста программы разработчик может воспользоваться любым удобным для него редактором текста: vi, vim, mceditor, emacs, geany и др.

После завершения написания исходного кода программы (возможно состоящей из нескольких файлов), необходимо её скомпилировать и получить исполняемый модуль.

Отладка — этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки. Чтобы понять, где возникла ошибка, приходится:

- узнавать текущие значения переменных;
- выяснять, по какому пути выполнялась программа.

Выполнение лабораторной работы

1. Создаю поддиректорию ~/work/os/lab_prog:

Выполнение лабораторной работы

2. Создаю файлы calculate.h, calculate.c, main.c и заполняю их, согласно описанию лабораторной работы:

Выполнение лабораторной работы

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.

Выполнение лабораторной работы

```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);
}
```

Рис. 1: Содержимое файла calculate.c.

Выполнение лабораторной работы

```
////////////////////////////////////////
// calculate.h

#ifndef CALCULATE_H_
#define CALCULATE_H_

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

#endif /*CALCULATE_H_*/
~
```

Рис. 2: Содержимое файла calculate.h.

Выполнение лабораторной работы

```c
//////////////////////////////////////
// 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;
}
```

Рис. 3: Содержимое файла main.c.

Выполнение лабораторной работы

3. Выполняю компиляцию программы посредством gcc:

Выполнение лабораторной работы

4. Создаю. Makefile с содержанием, согласно описанию лабораторной работы, при этом немного изменяю его:

```
#
# 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
```

Рис. 4: Содержимое Makefile.

Выполнение лабораторной работы

В Makefile указываю компилятор gcc, флаг –g и дополнительные библиотеки –lm. Описываю, какие команды необходимо запустить, чтобы получить файлы calcul, calculate.o и main.o, подключив дополнительные библиотеки и флаги. А в поле clean описывается удаление файлов calcul и файлов, оканчивающихся на “.o”.

Выполнение лабораторной работы

5. С помощью gdb выполняю отладку программы calcul:

Запускаю отладчик GDB, загрузив в него программу для отладки.

Запускаю программу внутри отладчика, введя команду run:

Выполнение лабораторной работы

Постранично просматриваю исходный код с помощью команды list.

Просматриваю строки с 12 по 15 основного файла с помощью команды list с параметрами:

```
(gdb) list
1       ////////////////////////////////////////
2       // main.c
3
4       #include <stdio.h>
5       #include "calculate.h"
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)
```

Рис. 5: Просмотр исходного кода постранично и указанных строк.

Выполнение лабораторной работы

Просматриваю определённые строки не основного файла с помощью команды list с параметрами:

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

Рис. 6: Просмотр определенных строк не основного файла.

Выполнение лабораторной работы

Устанавливаю точку останова в файле calculate.c на строке номер 21 и вывожу информацию об имеющихся в проекте точках останова:

```
(gdb) list calculate.c: 20,27
20      {
21      printf("Вычитаемое: ");
22      scanf("%f",&SecondNumeral);
23      return(Numeral - SecondNumeral);
24      }
25      else if(strncmp(Operation, "*", 1) == 0)
26      {
27      printf("Множитель: ");
(gdb) break 21
Breakpoint 1 at 0x40120f: file calculate.c, line 21.
(gdb) info breakpoints
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x000000000040120f in Calculate
                                                   at calculate.c:21
(gdb)
```

Рис. 7: Установка точки останова на строке 21 файла calculate.c и просмотр информации.

Выполнение лабораторной работы

Запускаю программу внутри отладчика и убеждаюсь, что программа остановилась в момент прохождения точки останова, затем с помощью команды backtrace просматриваю весь стек вызываемых функций от начала программы до текущего места:

Выполнение лабораторной работы

Просматриваю, чему равно значение переменной Numeral, введя сначала print Numeral, она равна 5, а затем сравниваю с выводом команды display Numeral. Можем заметить, что выводы разные, но значение одно - 5. Затем удаляю точку останова:

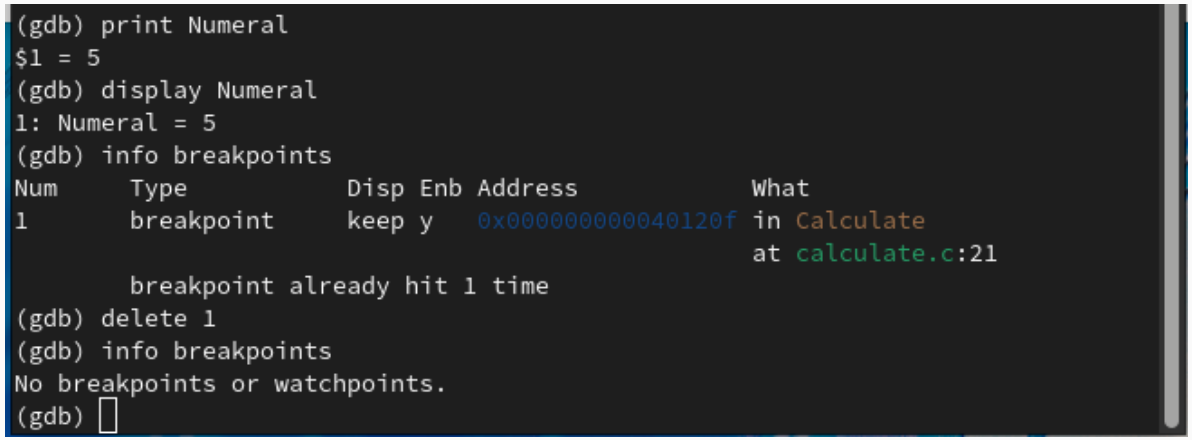

Рис. 8: Просмотр значения переменной Numeral и удаление точки останова.

Выполнение лабораторной работы

6. С помощью заранее установленной утилиты splint анализирую коды файла calculate.c (вижу 15 предупреждений) и файла main.c (вижу 4 предупреждения)

Выполнение лабораторной работы