Лабораторная работа №13

Матюшкин Денис Владимирович (НПИбд-02-21)

31.05.2022

RUDN University, Moscow, Russian Federation

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.

Ход работы

1. Создание файлов

- В домашнем каталоге создадим подкаталог ~/work/os/lab_prog. Создадим в нём файлы: calculate.h, calculate.c, main.c (рис. 1).

```
[dvmatyushkin@dvmatyushkin ~]$ mkdir work/os/lab_prog
[dvmatyushkin@dvmatyushkin ~]$ cd work/os/lab_prog
[dvmatyushkin@dvmatyushkin lab_prog]$ touch calculate.h calculate.c main.c
[dvmatyushkin@dvmatyushkin lab_prog]$
```

Рис. 1: Создание файлов

2. Редактирование файлов

- 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. Реализация функций калькулятора в файле calculate.c (рис. 2). Интерфейсный файл calculate.h, описывающий формат вызова функции-калькулятора (рис. 3). Основной файл main.c, реализующий интерфейс пользователя к калькулятору (рис. 4).

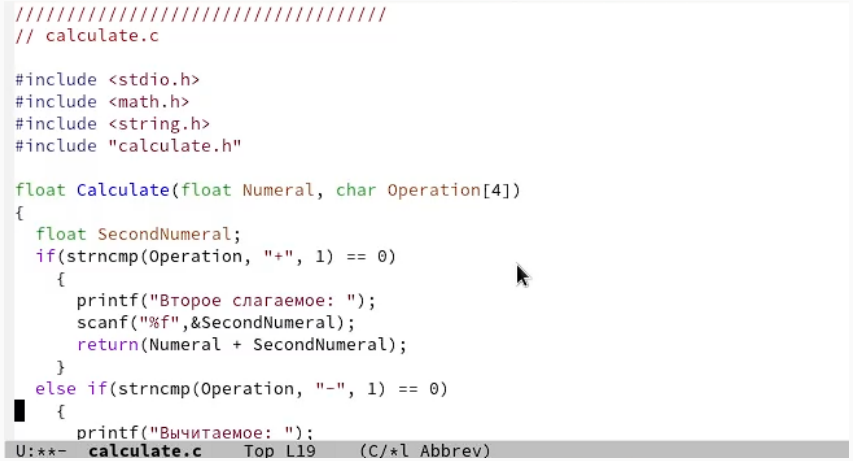

Рис. 2: Редактирование calculate.c

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

#ifndef CALCULATE_H_
#define CALCULATE_H_

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

#endif /*CALCULATE_H_*/
```

Рис. 3: Редактирование calculate.h

```
////////////////////////////////////////
// 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;
}
U:**-  main.c          All L17    (C/*l Abbrev)
```

Рис. 4: Редактирование main.c

3. Компиляция

- Выполним компиляцию программы посредством gcc (рис. 5).

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

Рис. 5: Компиляция файлов

4. Makefile

- Создадим Makefile со следующим содержанием (рис. 6). В этом файле мы создаем переменные CC, CFLAGS, LIBS. Инициализируем их. Создаем блоки, в которых прописываем какие команды будут выполняться. При этом подставляя значение нами созданных переменных.

```
#
# 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. 6: Создание Makefile

- С помощью gdb выполним отладку программы calcul (перед использованием gdb исправим Makefile) (рис. 7).

```
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 *~
```

Рис. 7: Редактирование Makefile

5.1. Запуск отладки

- Запустим отладчик GDB, загрузив в него программу для отладки (рис. 8).

```
[dvmatyushkin@dvmatyushkin lab_prog]$ make
gcc -c calculate.c -g
gcc -c main.c -g
gcc calculate.o main.o -o calcul -lm
[dvmatyushkin@dvmatyushkin lab_prog]$ gdb ./calcul
GNU gdb (GDB) Fedora 11.2-2.fc35
Copyright (C) 2022 Free Software Foundation, Inc.
```

Рис. 8: Запуск отладки

5.2. Работа с отладкой

- Для запуска программы внутри отладчика введем команду run. Для постраничного (по 9 строк) просмотра исходного код используем команду list (рис. 9).

```
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 16
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): sqrt
  4.00
[Inferior 1 (process 10051) exited normally]
(gdb) list
1       ////////////////////////////////////////
2       // main.c
3
4       #include <stdio.h>
5       #include "calculate.h"
6
7       int main (void)
8       {
9         float Numeral;
10        char Operation[4];
```

Рис. 9: Работа с отладкой

5.3. Работа с отладкой

- Для просмотра строк с 12 по 15 основного файла используем list с параметрами. Для просмотра определённых строк не основного файла используем list с параметрами (рис. 10).

```
(gdb) list 12,15
12          printf("Число: ");
13          scanf("%f",&Numeral);
14          printf("Операция (+,-,*,/,pow,sqrt,sin,cos,tan): ");
15          scanf("%s",&Operation);
(gdb) list calculate.c:20,29
20              printf("Вычитаемое: ");
21              scanf("%f",&SecondNumeral);
22              return(Numeral - SecondNumeral);
23            }
24          else if(strncmp(Operation, "*", 1) == 0)
25            {
26              printf("Множитель: ");
27              scanf("%f",&SecondNumeral);
28              return(Numeral * SecondNumeral);
29            }
```

Рис. 10: Работа с отладкой

5.4. Работа с отладкой

- Установим точку останова в файле calculate.c на строке номер 21. Выведем информацию об имеющихся в проекте точка останова (рис. 11).

```
(gdb) list calculate.c:20,27
20              printf("Вычитаемое: ");
21              scanf("%f",&SecondNumeral);
22              return(Numeral - SecondNumeral);
23            }
24          else if(strncmp(Operation, "*", 1) == 0)
25            {
26              printf("Множитель: ");
27              scanf("%f",&SecondNumeral);
(gdb) break 21
Breakpoint 1 at 0x40121e: file calculate.c, line 21.
(gdb) info breakpoints
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x000000000040121e in Calculate at calculate.c:21
```

Рис. 11: Установка точки останова

5.5. Работа с отладкой

- Запустим программу внутри отладчика и убедимся, что программа остановится в момент прохождения точки останова (рис. 12). Отладчик выдаст следующую информацию:

```
#0 Calculate (Numeral=5, Operation=0x7fffffffd280 "-")
at calculate.c:21
#1 0x0000000000400b2b in main () at main.c:17
```

```
(gdb) run
Starting program: /home/dvmatyushkin/work/os/lab_prog/calcul
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 5
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): -

Breakpoint 1, Calculate (Numeral=5, Operation=0x7fffffffdea4 "-") at calculate.c:21
21            scanf("%f",&SecondNumeral);
(gdb) backtrace
#0  Calculate (Numeral=5, Operation=0x7fffffffdea4 "-") at calculate.c:21
#1  0x00000000004014eb in main () at main.c:16
```

Рис. 12: Работа с точкой останова

5.6. Работа с отладкой

- Посмотрим, чему равно на этом этапе значение переменной Numeral. Сравним с результатом вывода на экран после использования команды. Уберем точки останова (рис. 13).

Рис. 13: Работа с отладкой

6. Утилита splint

- С помощью утилиты splint проанализируем коды файлов calculate.c и main.c (рис. 14 и рис. 15). Здесь мы можем увидеть что утилита splint выводит информацию о коде программы. Например то, что возвращаемое значение функции scanf() игнорируется .

```
[dvmatyushkin@dvmatyushkin 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:9:37: Function parameter Operation declared as manifest array (size
                    constant is meaningless)
calculate.c: (in function Calculate)
calculate.c:15:7: 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:21:7: Return value (type int) ignored: scanf("%f", &Sec...
```

Рис. 14: splint calculate.c

```
[dvmatyushkin@dvmatyushkin 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:13:3: Return value (type int) ignored: scanf("%f", &Num...
```

Рис. 15: splint main.c

Вывод

Вывод

- В ходе этой лабораторной работы мы приобрели простейшие навыки разработки, анализа, тестирования и отладки приложений в ОС типа UNIX/Linux на примере создания на языке программирования C калькулятора с простейшими функциями.

Спасибо за внимание!