Лабораторная работа № 13. Средства, применяемые при разработке программного обеспечения в ОС типа UNIX/Linux

Chvanova A.D.

NEC–2022, 24 May, Moscow

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

Теоретическое введение

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;

Выполнение лабораторной работы

1. В домашнем каталоге создайте подкаталог ~/work/os/lab_prog.(рис. 1)

```
[adchvanova@fedora ~]$ mkdir -p ~/work/os/lab_prog
```

Figure 1: подкаталог ~/work/os/lab_prog

Выполнение лабораторной работы

2. Создайте в нём файлы: calculate.h, calculate.c, main.c.(рис. 2) Это будет примитивнейший калькулятор, способный складывать, вычитать, умножать и делить, возводить число в степень, брать квадратный корень, вычислять sin, cos, tan. (рис. 3,4,5,6) При запуске он будет запрашивать первое число, операцию, второе число. После этого программа выведет результат и остановится.

```
[adchvanova@fedora ~]$ cd ~/work/os/lab_prog
[adchvanova@fedora lab_prog]$ touch calculate.h calculate.c main.c
[adchvanova@fedora lab_prog]$ ls
calculate.c  calculate.h  main.c
```

Figure 2: Создание файлов

Выполнение лабораторной работы

```
////////////////////////////////////
 // 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: ");
```

Figure 3: Файл calculate.c

Выполнение лабораторной работы

```
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);
}
}
```

Figure 4: Файл calculate.c

Выполнение лабораторной работы

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

 #ifndef CALCULATE_H_
 #define CALCULATE_H_

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

 #endif /*CALCULATE_H_*/
```

Figure 5: Файл calculate.h

Выполнение лабораторной работы

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

Figure 6: Файл main.c

Выполнение лабораторной работы

3. Выполните компиляцию программы посредством gcc.(рис. 7)

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

Figure 7: Компиляция программы посредством gcc

4. При необходимости исправьте синтаксические ошибки.

Ошибок нет.

Выполнение лабораторной работы

5. Создайте Makefile.(рис. 8,9)

```
[adchvanova@fedora lab_prog]$ touch Makefile
[adchvanova@fedora lab_prog]$ ls
calcul       calculate.c~  calculate.h~  main.c   main.o
calculate.c  calculate.h   calculate.o   main.c~  Makefile
```

Figure 8: Создание Makefile

Выполнение лабораторной работы

```
#
 # 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 *~

# End Makefil
```

Figure 9: Makefile

Выполнение лабораторной работы

6. С помощью gdb выполните отладку программы calcul (перед использованием gdb исправьте Makefile). (рис. 10,11,12)

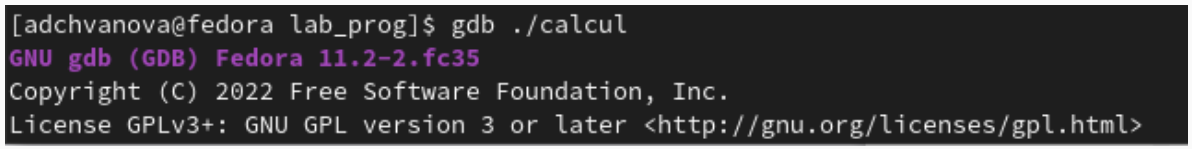

Figure 10: Запуск отладчика GDB

Выполнение лабораторной работы

```
(gdb) run
Starting program: /home/adchvanova/work/os/lab_prog/calcul
Downloading separate debug info for /home/adchvanova/work/os/lab_prog/system-sup
plied DSO at 0x7ffff7fc5000...
Downloading separate debug info for /lib64/libm.so.6...
Downloading separate debug info for /lib64/libc.so.6...
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 5
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): +
Второе слагаемое: 7
 12.00
[Inferior 1 (process 9256) exited normally]
(gdb) list
Downloading source file /usr/src/debug/glibc-2.34-33.fc35.x86_64/elf/sofini.c...
1       /* Terminate the frame unwind info section with a 4byte 0 as a sentinel;
2          this would be the 'length' field in a real FDE.  */
3
4       typedef unsigned int ui32 __attribute__ ((mode (SI)));
5       static const ui32 __FRAME_END__[]
6         __attribute__ ((used, section (".eh_frame")))
7         = { 0 };
(gdb) list 12,15
```

Figure 11: команда run, list(обычный и с параметром)

Выполнение лабораторной работы

```
(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)
```

Figure 12: list calculate.c:20,29

Выводы

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!