Лабораторная работа №13

Курс “Операционные Системы”

Шилоносов Д.В., НКАбд-03-22

06 мая 2023

Российский университет дружбы народов, Москва, Россия

Информация

Докладчик

- Шилоносов Данил Вячеславович
- студент группы НКАбд-03-22
- кафедры Компьютерные и информационные науки
- Российский университет дружбы народов
- 1132221810@pfur.ru

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.

Задачи

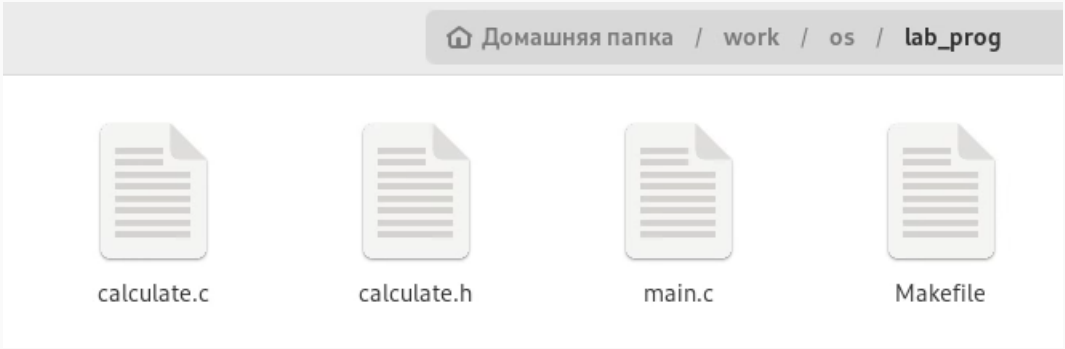
1. В домашнем каталоге создайте подкаталог ~/work/os/lab_prog.
2. Создайте в нём файлы: calculate.h, calculate.c, main.c. Это будет примитивнейший калькулятор, способный складывать, вычитать, умножать и делить, возводить число в степень, брать квадратный корень, вычислять sin, cos, tan. При запуске он 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).
7. С помощью утилиты splint попробуйте проанализировать коды файлов calculate.c и main.c.

Выполнение лабораторной работы

Редактирование, компиляция и линковка исходных файлов

Исходные файлы

Создадим файлы calculate.c, calculate.h, main.c и Makefile



Далее, вставим в эти файлы соответствующий код из условия задания.

Редактирование Makefile

Исправим описание Makefile (добавим опцию -g для того, чтобы сохранялась отладочная информация)

```
#
# 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 *~
```

```
#
# Makefile
#

CC = gcc
CFLAGS = -g
LIBS = -lm

calcul: calculate.o main.o
	$(CC) calculate.o main.o -o calcul $(LIBS)

calculate.o: calculate.c calculate.h
	$(CC) -c calculate.c $(CFLAGS)

main.o: main.c calculate.h
	$(CC) -c main.c $(CFLAGS)

clean:
	-rm calcul *.o *~
```

С помощью компилятора GCC скомпилируем объектных файлы, затем слинкуем их в исполняемую программу

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

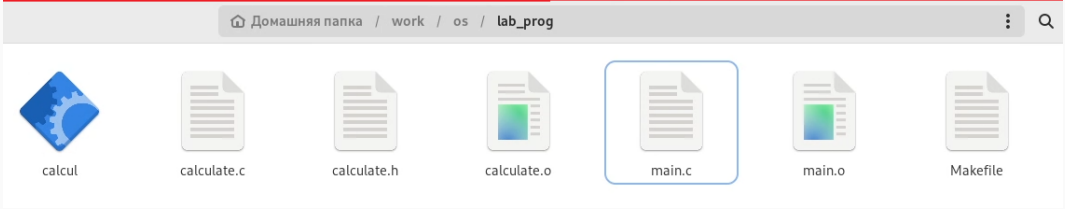


Рис. 2: Компиляция и линковка

Работа с отладчиком gdb

Запуск отладчика gdb

```
[dvshilonosov@dvshilonosov lab_prog]$ gdb ./calcul
GNU gdb (GDB) Fedora Linux 13.1-3.fc37
Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./calcul...
(gdb)
```

Запуск программы

В отладчике запустим программу и введем требуемые значения

```
(gdb) run
Starting program: /home/dvshilonosov/work/os/lab_prog/calcul

This GDB supports auto-downloading debuginfo from the following URLs:
  <https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 5
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): pow
Степень: 3
125.00
[Inferior 1 (process 6075) exited normally]
```

Рис. 4: Запуск программы в отладчике gdb

С помощью команды list отобразим нужную для просмотра часть кода, а затем, с помощью команды break поставим точку остановы:

```
(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) 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: ");
```

Запуск программы

Вновь запустив программу, видим дополнительную информацию, касающуюся ее выполнения

```
(gdb) run
Starting program: /home/dvshilonosov/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=0x7fffffffdf24 "-") at calculate.c:21
21      printf("Вычитаемое: ");
(gdb) backtrace
#0  Calculate (Numeral=5, Operation=0x7fffffffdf24 "-") at calculate.c:21
#1  0x00000000004014eb in main () at main.c:16
(gdb) print Numeral
$1 = 5
(gdb) display Numeral
1: Numeral = 5
```

Утилита splint

Утилита splint

Запустим с помощью утилиты splint файл calculate.c и увидим информацию, которые содержит предупреждения о программе:

```
[dvshilonosov@dvshilonosov lab_prog]$ splint calculate.c
  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:22:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:28:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:34:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:35:4: 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:38:7: Return value type double does not match declared type float:
                    (HUGE_VAL)
  To allow all numeric types to match, use +relaxtypes.
calculate.c:46:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:47:7: Return value type double does not match declared type float:
                    (pow(Numeral, SecondNumeral))
calculate.c:50:7: Return value type double does not match declared type float:
                    (sqrt(Numeral))
calculate.c:52:7: Return value type double does not match declared type float:
                    (sin(Numeral))
calculate.c:54:7: Return value type double does not match declared type float:
```

Результаты

Выводы

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