Отчёт по лабораторной работе №14

Серегин Денис Алексеевич

Содержание

List of Tables

List of Figures

1 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.

2 Задания

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):

   – Запустите отладчик GDB, загрузив в него программу для отладки: gdb ./calcul

   – Для запуска программы внутри отладчика введите команду run: run

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

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

   – Для просмотра определённых строк не основного файла используйте list с параметрами: list calculate.c:20,29

– Установите точку останова в файле calculate.c на строке номер 21: list calculate.c:20,27 break 21

– Выведите информацию об имеющихся в проекте точка останова: info breakpoints

– Запустите программу внутри отладчика и убедитесь, что программа остановится в момент прохождения точки останова: run 5 - backtrace

– Посмотрите, чему равно на этом этапе значение переменной Numeral, введя: print Numeral На экран должно быть выведено число 5.

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

– Уберите точки останова: info breakpoints delete 1

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

3 Выполнение лабораторной работы

1. В домашнем каталоге создал подкаталог ~/work/os/lab_prog. 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.

   (рис 1. 3.1)

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

Figure 3.1: Создание файлов и директории

2. Выполнил компиляцию программы посредством gcc:

   gcc -c calculate.c

   gcc -c main.c

   gcc calculate.o main.o -o calcul –lm

   (рис 2. 3.2)

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

Figure 3.2: Компиляция программы с помощью gcc

3. Исправил синтаксические ошибки:

   scanf(“%s”,Operation)

4. Создал Makefile с содержанием из указаний к лабораторной работе

5. С помощью gdb выполнил отладку программы calcul (перед использованием gdb исправил Makefile: CFLAGS=-g)

   – Запустил отладчик GDB, загрузив в него программу для отладки: gdb ./calcul

   (рис 3. 3.3)

```
[daseregin@localhost lab_prog]$ gdb ./calcul
GNU gdb (GDB) Red Hat Enterprise Linux 7.6.1-120.el7
Copyright (C) 2013 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".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/daseregin/work/os/lab_prog/calcul...(no debugging symbols found)...done.
(gdb)
```

Figure 3.3: Запуск калькулятора в отладчике GDB

   – Для запуска программы внутри отладчика ввел команду run

   (рис 4. 3.4)

```
(gdb) run
Starting program: /home/daseregin/work/os/lab_prog/./calcul
Число: 5
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): +
Второе слагаемое: 9
 14.00
[Inferior 1 (process 7258) exited normally]
Missing separate debuginfos, use: debuginfo-install glibc-2.17-317.el7.x86_64
(gdb)
```

Figure 3.4: Работа команды run

6. С помощью утилиты splint попробовал проанализировать коды файлов calculate.c и main.c.

(рис 5. 3.5)

```
[root@localhost lab_prog]# splint calculate.c
Splint 3.1.2 --- 11 Oct 2015

calculate.h:4:37: 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:7:31: Function parameter Operation declared as manifest array (size
                    constant is meaningless)
calculate.c: (in function Calculate)
calculate.c:13:1: 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:19:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:25:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:31:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:32:4: 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:35:7: Return value type double does not match declared type float:
                    (HUGE_VAL)
  To allow all numeric types to match, use +relaxtypes.
calculate.c:43:1: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:44:7: Return value type double does not match declared type float:
                    (pow(Numeral, SecondNumeral))
calculate.c:47:7: Return value type double does not match declared type float:
                    (sqrt(Numeral))
calculate.c:49:7: Return value type double does not match declared type float:
                    (sin(Numeral))
calculate.c:51:7: Return value type double does not match declared type float:
                    (cos(Numeral))
calculate.c:53:7: Return value type double does not match declared type float:
                    (tan(Numeral))
calculate.c:57:7: Return value type double does not match declared type float:
                    (HUGE_VAL)

Finished checking --- 15 code warnings
```

Figure 3.5: Работа команды splint

(рис 6. 3.6)

```
[root@localhost lab_prog]# splint main.c
Splint 3.1.2 --- 11 Oct 2015

calculate.h:4:37: 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:11:1: Return value (type int) ignored: scanf("%f", &Num...
  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)
main.c:13:1: Return value (type int) ignored: scanf("%s", Oper...

Finished checking --- 3 code warnings
```

Figure 3.6: Работа команды splint

4 Вывод

В ходе лабораторной работы мне удалось 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.

5 Использованные ресурсы

https://onstartup.ru/razrabotka/splint/