Лабораторная работа №13

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

Галиева Аделина Руслановна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Приобрести простейшие навыки разработки, анализа, тестирования и отладки приложений в ОС типа UNIX/Linux на примере создания на языке программирования. С калькулятора с простейшими функциями.

2 Выполнение лабораторной работы

1. В домашнем каталоге создаём подкаталог ~/work/os/lab_prog. (рис. 2.1)

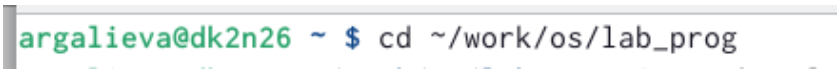

Рис. 2.1: Создаем подкаталог

2. Создаём в нём файлы: calculate.h, calculate.c, main.c. (рис. 2.2) (рис. 2.3)

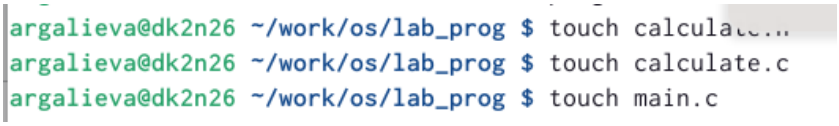

Рис. 2.2: Создаем файлы

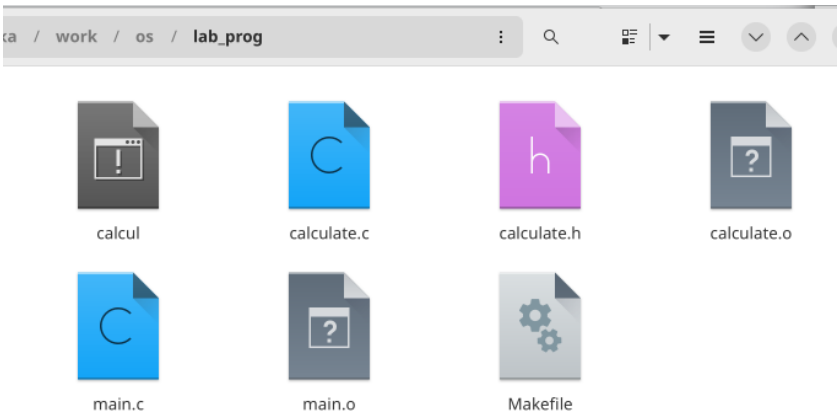

Рис. 2.3: Файлы

3. Выполняем компиляцию программы посредством gcc. (рис. 2.4)

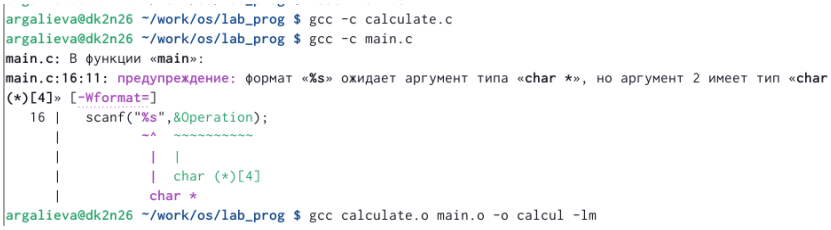

Рис. 2.4: Компиляция

4. С помощью gdb выполняем отладку программы calcul (перед использованием gdb исправили Makefile). (рис. 2.5) (рис. 2.6)

Рис. 2.5: Использование откладчика

```
(gdb) list
No symbol table is loaded.  Use the "file" command.
(gdb) list 12,15
No symbol table is loaded.  Use the "file" command.
(gdb) list calculate.c:20,29
No symbol table is loaded.  Use the "file" command.
(gdb) list calculate.c:20,27
No symbol table is loaded.  Use the "file" command.
(gdb) break 21
No symbol table is loaded.  Use the "file" command.
(gdb) info breakpoints
No breakpoints or watchpoints.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/a/r/argalieva/work/os/lab_prog/calcul
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/usr/lib64/libthread_db.so.1".
Число: 5
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): -
Вычитаемое:
backtrace
  5.00
[Inferior 1 (process 3829) exited normally]
(gdb) print Numeral
No symbol table is loaded.  Use the "file" command.
(gdb) display Numeral
No symbol table is loaded.  Use the "file" command.
(gdb) info breakpoints
No breakpoints or watchpoints.
(gdb) delete 1
```

Рис. 2.6: Использование откладчика

5. С помощью утилиты splint анализируем коды файлов calculate.c и main.c. (рис. 2.7) (рис. 2.8)

```
argalieva@dk2n26 ~/work/os/lab_prog $ splint main.c
Splint 3.1.2 --- 07 Dec 2021

calculate.h:7:38: 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:14:3: 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:16:14: Format argument 1 to scanf (%s) expects char * gets char [4] *:
                &Operation
  Type of parameter is not consistent with corresponding code in format string.
  (Use -formattype to inhibit warning)
   main.c:16:11: Corresponding format code
main.c:16:3: Return value (type int) ignored: scanf("%s", &Ope...

Finished checking --- 4 code warnings
```

Рис. 2.7: splint main.c

```
argalieva@dk2n26 ~/work/os/lab_prog $ splint calculate.c
Splint 3.1.2 --- 07 Dec 2021

calculate.h:7:38: 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:10:31: Function parameter Operation declared as manifest array
                     (size constant is meaningless)
calculate.c: (in function Calculate)
calculate.c:16:7: 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:22:7: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:28:7: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:34:7: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:35:10: Dangerous equality comparison involving float types:
                     SecondNumeral == 0
  Two real (float, double, or long double) values are compared directly using
```

Рис. 2.8: splint calculate.c

3 Контрольные вопросы

1. Как получить информацию о возможностях программ gcc, make, gdb и др.? Ответ: Для этого есть команда man и предлагающиеся к ней файлы.

2. Назовите и дайте краткую характеристику основным этапам разработки прило- жений в UNIX. Ответ: Кодировка, Компиляция, Тест.

3. Что такое суффикс в контексте языка программирования? Приведите примеры использования. Ответ: Это расширения файлов.

4. Каково основное назначение компилятора языка C в UNIX? Ответ: Программа gcc, которая интерпретирует к определенному языку программирования аргументы командной строки и определяет запуск нужного компилятора для нужного файла.

5. Для чего предназначена утилита make? Ответ: Для компиляции группы файлов. Собрания из них программы, и последующего удаления.

6. Приведите пример структуры Makefile. Дайте характеристику основным элементам этого файла. Ответ:

```
program: main.o lib.o
cc -o program main.o lib.o
main.o lib.o: defines.h
```

В имени второй цели указаны два файла и для этой же цели не указана команда компиляции. Кроме того, нигде явно неуказана зависимость объектных

файлов от «*.c»-файлов. Дело в том, что программа make имеет предопределённые правила для получения файлов с определёнными расширениями. Так, для цели-объектного файла (расширение «.o») при обнаружении соответствующего файла с расширением «.c» будет вызван компилятор «cc -c» с указанием в параметрах этого «.c»-файла и всех файлов-зависимостей.

7. Назовите основное свойство, присущее всем программам отладки. Что необходимо сделать, чтобы его можно было использовать? Ответ: Программы для отладки нужны для нахождения ошибок в программе. Для их использования надо скомпилировать программу таким образом, чтобы отладочная информация содержалась в конечном бинарном файле.

8. Назовите и дайте основную характеристику основным командам отладчика gdb. Ответ:

backtrace – выводит весь путь к текущей точке останова, то есть названия всех функций, начиная от main(); иными словами, выводит весь стек функций;

break – устанавливает точку останова; параметром может быть номер строки или название функции;

clear – удаляет все точки останова на текущем уровне стека (то есть в текущей функции);

continue – продолжает выполнение программы от текущей точки до конца;

delete – удаляет точку останова или контрольное выражение;

display – добавляет выражение в список выражений, значения которых отображаются каждый раз при остановке программы;

finish – выполняет программу до выхода из текущей функции; отображает возвращаемое значение,если такое имеется;

info breakpoints – выводит список всех имеющихся точек останова;

info watchpoints – выводит список всех имеющихся контрольных выражений;

list – выводит исходный код; в качестве параметра передаются название файла исходного кода, затем, через двоеточие, номер начальной и конечной строки;

next – пошаговое выполнение программы, но, в отличие от команды step, не выполняет пошагово вызываемые функции;

print – выводит значение какого-либо выражения (выражение передаётся в качестве параметра);

run – запускает программу на выполнение;

set – устанавливает новое значение переменной

step – пошаговое выполнение программы;

watch – устанавливает контрольное выражение, программа остановится, как только значение контрольного выражения изменится;

9. Опишите по шагам схему отладки программы, которую Вы использовали при выполнении лабораторной работы. Ответ:
10. gdb –silent ./calcul
11. run
12. list
13. backtrace
14. breakpoints
15. print Numeral
16. Splint (Не использовался по причине отсутствия команды в консоли).
17. Прокомментируйте реакцию компилятора на синтаксические ошибки в программе при его первом запуске. Ответ: Консоль выводит ошибку с номером строки и ошибочным сегментом, но при этом есть возможность выполнить программу сразу.
18. Назовите основные средства, повышающие понимание исходного кода программы. Ответ:

a) Правильный синтаксис
b) Наличие комментариев
c) Разбиение большой сложной программы на несколько сегментов попроще.

12. Каковы основные задачи, решаемые программой splint? Ответ: split – разбиение файла на меньшие, определённого размера. Может разбивать текстовые файлы по строкам и любые – по байтам. По умолчанию читает со стандартного ввода и создает файлы с именами вида xaa, xab и т.д. По умолчанию разбиение идёт по 1000 строк в файле.

4 Выводы

Я приобрела простейшие навыки разработки, анализа, тестирования и отладки приложений в ОС типа UNIX/Linux на примере создания на языке программирования. С калькулятора с простейшими функциями.

Список литературы