Front matter

lang: ru-RU title: Средства, применяемые при разработке программного обеспечения в ОС типа UNIX/Linux. subtitle: ЛР по ОС №14 author: Танрибергенов Эльдар Марсович group: НПИбд-02-20

Отчёт

о выполнении лабораторной работы № 14.

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

Выполнил:

студент группы НПИбд-02-20

Танрибергенов Эльдар Марсович.

Студ. билет № 1032208074

Москва 2021 г.

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).

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

Теоретическое введение

Стадии цикла разработки ПО [1]:

Анализ требований, Проектирование, Разработка, Документация, Тестирование, Внедрение

Стандартным средством для компиляции программ в ОС типа UNIX является GCC (GNU Compiler Collection). Это набор компиляторов для разного рода языков программирования (C, C++, Java, Фортран и др.). Работа с GCC производится при помощи одноимённой управляющей программы gcc, которая интерпретирует аргументы командной строки, определяет и осуществляет запуск нужного компилятора для входного файла.

Во время работы над кодом программы программист неизбежно сталкивается с появлением ошибок в ней. Использование отладчика для поиска и устранения ошибок в программе существенно облегчает жизнь программиста. В комплект программ GNU для ОС типа UNIX входит отладчик GDB (GNU Debugger).

Ещё одним средством проверки исходных кодов программ, написанных на языке C, является утилита splint. Эта утилита анализирует программный код, проверяет корректность задания аргументов использованных в программе функций и типов возвращаемых значений, обнаруживает синтаксические и семантические ошибки.

Ход работы:

Создаю нужный каталог и файлы в нём.

https://github.com/emtanribergenov/OS_labs/blob/master/14/screenshots/cdf.png

Файл calculate.c

https://github.com/emtanribergenov/OS_labs/blob/master/14/screenshots/calc%20(1).png

https://github.com/emtanribergenov/OS_labs/blob/master/14/screenshots/calc%20(2).png

https://github.com/emtanribergenov/OS_labs/blob/master/14/screenshots/calc%20(3).png

Файл calculate.h

https://github.com/emtanribergenov/OS_labs/blob/master/14/screenshots/ch.png

Файл main.c

https://github.com/emtanribergenov/OS_labs/blob/master/14/screenshots/m.png

Компиляция.

https://github.com/emtanribergenov/OS_labs/blob/master/14/screenshots/cmpl.png

Файл Makefile

https://github.com/emtanribergenov/OS_labs/blob/master/14/screenshots/mf.png

Запуск отладчика gdb

https://github.com/emtanribergenov/OS_labs/blob/master/14/screenshots/gdbr%20(1).png

Отладка началась, и мне удалось запустить программу с помощью run.

https://github.com/emtanribergenov/OS_labs/blob/master/14/screenshots/gbdr%205.png

Команда list.

https://github.com/emtanribergenov/OS_labs/blob/master/14/screenshots/l1.png

https://github.com/emtanribergenov/OS_labs/blob/master/14/screenshots/l2.png

Установка точки останова на 21 строке

https://github.com/emtanribergenov/OS_labs/blob/master/14/screenshots/l3.png

Инфа о точках останова

!https://github.com/emtanribergenov/OS_labs/blob/master/14/screenshots/bp.png

Запуск командой run . Программа остановилась на 21 строке. Ввёл команды display и print Numeral. Вывел инфу о ТО и удалил имеющуюся.

https://github.com/emtanribergenov/OS_labs/blob/master/14/screenshots/bd.png

Анализ Makefile

Первые три не-комментария - объявление переменных.

Последняя - аргументы для создания calcul

Вторая - пустая, используется при создании исполняемых файлов. Видимо, её можно как-то заполнить извне

Первая нигде не используется. Её смысл мне неясен.

Далее идут четыре метода(?) создания файла.

Первые три идентичны командам создания файлов, вводимых в работе.

Причём первый в ходе работы вызывает второй и третий, как я понял.

Последний же убирает все файлы, по сути - метод удаления.

Вывод:

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

Ответы на вопросы:

1. Просмотреть справку или искать в интернете.
2. - 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; тестирование и отладка, сохранение произведённых изменений;
   - Документирование.
3. Суффиксы нужны, когда компилятор не может определить подходящий тип переменной, либо по умолчанию определит его не так, как вам надо.
4. Компилятор языка C выполняет как собственно компиляцию - перевод исходного текста на машинный язык, результатом чего является объектный модуль, так и редактирование связей - сборку из нескольких объектных модулей (в том числе, и библиотечных) исполняемого модуля.
5. Утилита make предназначена для автоматизации сборки проектов.

6. Структура Makefile:

- цель: зависимости
- [tab] команда

7. Этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки. Запустить его командой-названием в терминале.

8. ◦ Запуск. Запускает исполняемый файл
   ◦ Статический анализ. Получение инфы и её анализ.
   ◦ Динамический анализ. Установление аргументов для запуска программы и наблюдение за ними.
   ◦ Отладка. Поэтапная проверка программы для устранения ошибок.

9. Не совсем понял, ведь предыдущий ответ вроде бы годится и для этого вопроса.

10. Указывает на синтаксические ошибки.

11. Система разработки приложений UNIX предоставляет различные средства, повышающие понимание исходного кода. К ним относятся: + – cscope - исследование функций, содержащихся в программе; + – lint — критическая проверка программ, написанных на языке Си.

12. Splint - это инструмент для статической проверки программ на C на наличие уязвимостей и ошибок кодирования. С минимальными усилиями Splint можно использовать в качестве лучшего ворса.