Лабораторная работа №14

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

Кузнецов Юрий Владимирович

Содержание

[Цель работы 1](#_Toc86264667)

[Задание 1](#_Toc86264668)

[Выполнение лабораторной работы 1](#_Toc86264669)

[Выводы 7](#_Toc86264670)

# 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.

# Задание

1. Создать каталог ~/work/os/lab\_prog.
2. Создать в нём файлы: calculate.h, calculate.c, main.c.
3. Выполнить компиляцию.
4. Создать makefile.
5. Выполнить отладку программы calcul.
6. Проанализировать коды файлов calculate.c и main.c.

# Выполнение лабораторной работы

Ознакомился с лабораторной работой №14 и приступил к выполнению заданий.

В домашнем каталоге создал подкаталог ~/work/os/lab\_prog.

В каталоге создал файлы: calculate.h, calculate.c, main.c. Это нужно для примитивнейшего калькулятора, способного складывать, вычитать,умножать и делить, возводить число в степень, брать квадратный корень, вы-числять sin, cos, tan. При запуске он будет запрашивать первое число, операцию, второе число. После этого программа должна будет вывести результат и остановится.

Ниже показаны команды запуска редактора emacs (рис. 1).Там же и на (рис. 2) показана компиляция программ с помощью gcc (опция -c создает объектные файлы, -o задает имена файлам).

Изображение выглядит как текст

Автоматически созданное описание

Figure 1: Команды

Изображение выглядит как текст

Автоматически созданное описание

Figure 2: Команды

В файл main.c записал этот код (рис. 3):

Изображение выглядит как текст

Автоматически созданное описание

Figure 3: main.c

В файл calculate.c записал этот (рис. 4):

Изображение выглядит как текст

Автоматически созданное описание

Figure 4: calculate.c

Содержимое файла calculate.h (рис. 5):

Изображение выглядит как текст

Автоматически созданное описание

Figure 5: calculate.h

Создал Makefile (рис. 6):



Figure 6: makefile

С помощью gdb выполнил отладку программы calcul. Чтобы запустить отладчик пишу ‘gdb ./calcul’. Для запуска калькулятора внутри отладчика пишу ‘run’.

Изображение выглядит как текст

Автоматически созданное описание

Figure 7: GDB

Чтобы просмотреть первые 10 строк кода пишу ‘list’. Или же использую лист с параметрами. Для точки останова пишу ‘break ’. (рис. 8)



Figure 8: GDB

Запустил программу внутри отладчика, программа остановилась в момент прохождения точки останова. Отладчик выдал следующую информацию:

‘#0 Calculate (Numeral=5, Operation=0x7fffffffcf14 “-”)at calculate.c:21’

‘#1 0x0000000000400c31 in main () at main.c:16’

Я вызвал команду backtrace и смог вывести весь стек вызываемых функций от начала программы до текущего места.

Посмотрел, чему равно на этом этапе значение переменной Numeral, введя:

print Numeral

Было выведено число 5.

Сравнил с результатом вывода на экран после использования команды:

display Numeral

Удалил точку останова (delete 1).

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст, газета

Автоматически созданное описание

Figure 9: GDB и splint

С помощью утилиты splint попробовала проанализировать коды файлов calculate.c и main.c (рис. 9) (рис. 10). Информацию о splint нашла здесь: [ссылка 2](http://espressocode.top/splint-c-program-verifier/)

Splint- это UNIX программа, позволяющая проводить статический анализ кода, написанного на Си.

Splint особенно хорош при проверке проверки типов назначений переменных и функций, эффективности, неиспользуемых переменных и идентификаторов функций, недоступного кода и возможных утечек памяти.

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Figure 10: splint

# Выводы

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