Лабораторная работа №13

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

Панченко Денис Дмитриевич

Содержание

# 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

В домашнем каталоге создадим подкаталог ~/work/os/lab\_prog (рис. [1](#fig:001)).

Figure 1: Создание подкаталога

Figure 1: Создание подкаталога

Создадим в нём файлы: calculate.h, calculate.c, main.c (рис. [2](#fig:002)).

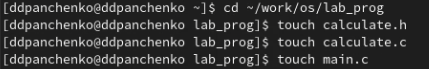


Figure 2: Создание файлов

Реализуем функцию калькулятора в файле calculate.c (рис. [3](#fig:003)).

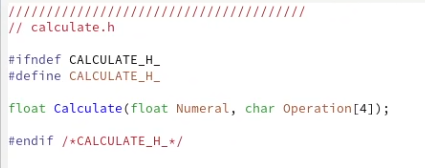


Figure 3: Реализация функции

Реализуем интерфейсный файл calculate.h, описывающий формат вызова функции-калькулятора (рис. [4](#fig:004)).



Figure 4: Реализация интерфейсного файла

Реализуем основной файл main.c, реализующий интерфейс пользователя к калькулятору (рис. [5](#fig:005)).

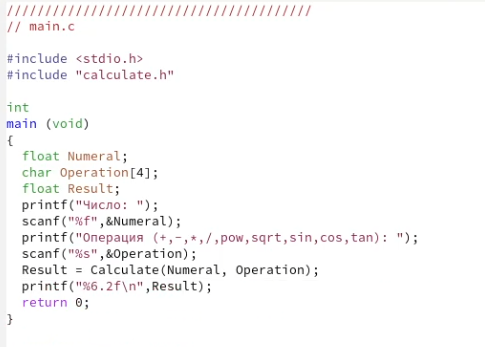


Figure 5: Реализация основного файла

Выполним компиляцию программы посредством gcc (рис. [6](#fig:006)).

Figure 6: Компиляция программы

Figure 6: Компиляция программы

Создадим Makefile со следующим содержанием (рис. [7](#fig:007) - [8](#fig:008)).

Figure 7: Создание файла

Figure 7: Создание файла

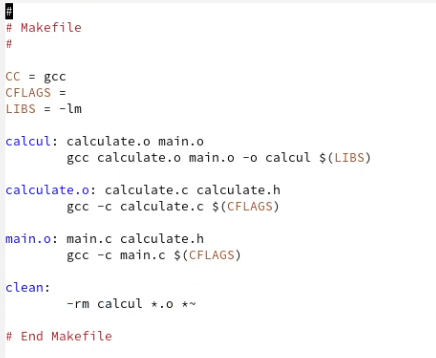


Figure 8: Содержание файла

С помощью gdb выполним отладку программы calcul (рис. [9](#fig:009) - [19](#fig:019)).

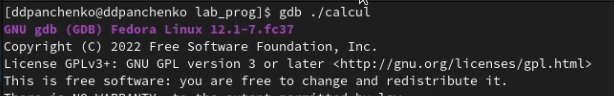


Figure 9: Запуск

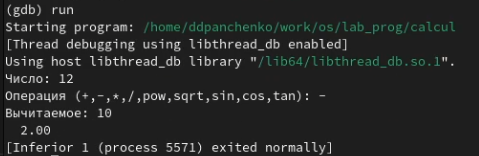


Figure 10: Запуск программы

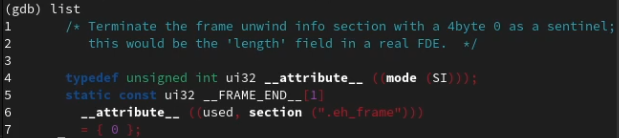


Figure 11: Просмотр кода

Figure 12: Просмотр строк кода

Figure 12: Просмотр строк кода

Figure 13: Просмотр строк неосновного кода

Figure 13: Просмотр строк неосновного кода

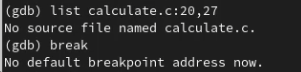


Figure 14: Установка точки останова

Figure 15: Вывод информации о точке

Figure 15: Вывод информации о точке

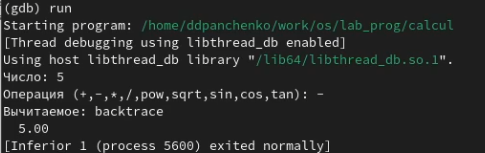


Figure 16: Запуск программы

Figure 17: Значение переменной

Figure 17: Значение переменной

Figure 18: Сравнение

Figure 18: Сравнение

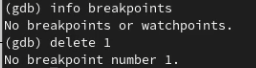


Figure 19: Удаление точки останова

# 3 Вывод

Я 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.

# 4 Контрольные вопросы

1. Информацию о возможностях программ gcc, make, gdb и др. можно получить из их официальной документации, а также из различных руководств и учебников по программированию на языке С и UNIX.
2. Основными этапами разработки приложений в UNIX являются:

* проектирование архитектуры приложения и выбор используемых технологий;
* написание и отладка исходного кода на языке С;
* компиляция исходного кода в исполняемый файл с помощью компилятора gcc;
* тестирование и отладка приложения с помощью отладчика gdb;
* сборка приложения и создание пакета для установки с помощью утилиты make.

1. Суффикс в контексте языка программирования - это часть имени файла, которая указывает на его тип или формат. Например, суффикс .c обозначает исходный код на языке С, а .o - объектный файл, полученный в результате компиляции исходного кода.
2. Основное назначение компилятора языка С в UNIX - преобразование исходного кода на языке С в машинный код, который может быть исполнен на компьютере.
3. Утилита make предназначена для автоматизации процесса сборки программы. Она позволяет определить зависимости между файлами и компонентами программы и автоматически проводить перекомпиляцию только тех файлов, которые изменились.
4. Пример структуры Makefile: CC=gcc CFLAGS=-Wall -Werror LDFLAGS=-lm

main: main.o utils.o $(CC) $(LDFLAGS) -o main main.o utils.o

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

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

Основные элементы: - переменные, например CC, CFLAGS, LDFLAGS, которые определяют используемый компилятор, флаги компиляции и линковки; - цели, например main, которые соответствуют именам файлов, которые нужно собрать; - зависимости, например main.o: main.c, которые определяют зависимости между целями и файлами; - команды, например $(CC) $(LDFLAGS) -o main main.o utils.o, которые определяют действия, необходимые для сборки файла.

1. Основное свойство, присущее всем программам отладки, - возможность остановки выполнения программы в определенной точке и пошагового ее выполнения. Чтобы его можно было использовать, необходимо включить в исходный код программы отладочную информацию, которую компилятор добавляет в объектный файл при использовании опции -g.
2. Основные команды отладчика gdb:

* run - запустить программу;
* break - установить точку останова;
* next - выполнить следующую строку кода, не заходя внутрь функций;
* step - выполнить следующую строку кода, заходя внутрь функций;
* print - вывести значение переменной;
* watch - установить точку останова при изменении значения переменной;
* quit - выйти из отладчика.

1. Схема отладки программы, которую я использовал при выполнении лабораторной работы:

* компиляция исходного кода с опцией -g;
* запуск отладчика gdb;
* установка точек останова в соответствующих местах кода;
* выполнение программы пошагово с помощью команд next и step;
* вывод значений переменных с помощью команды print;
* исправление ошибок в исходном коде и повторная компиляция.

1. Компилятор при первом запуске реагирует на синтаксические ошибки в программе и сообщает о них, указывая номер строки и тип ошибки. Это позволяет исправить ошибки и повторно скомпилировать программу.
2. Основные средства, повышающие понимание исходного кода программы:

* комментарии в коде, поясняющие его структуру и логику;
* правильное именование переменных, функций и классов;
* форматирование кода для улучшения его читаемости;
* использование отладочных сообщений, которые выводят информацию о ходе выполнения программы.

1. Основные задачи, решаемые программой splint - проверка кода на соответствие стандартам и рекомендациям по программированию на языке С. Она позволяет выявлять потенциальные ошибки и уязвимости в коде, а также улучшать его читаемость и поддерживаемость.