Лабораторная работа № 13

Операционные системы

Голованова Мария Константиновна

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 Задание

Создать на языке программирования С 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 Теоретическое введение

## 3.1 Компиляция исходного текста и построение исполняемого файла

Стандартным средством для компиляции программ в ОС типа UNIX является GCC (GNU Compiler Collection). Это набор компиляторов для разного рода языков программирования (С, 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. Файлы с расширением (суффиксом) .c воспринимаются gcc как программы на языке С, файлы с расширением .cc или .C — как файлы на языке C++, а файлы c расширением .o считаются объектными. Для сборки разрабатываемого приложения и собственно компиляции полезно воспользоваться утилитой make. Она позволяет автоматизировать процесс преобразования файлов программы из одной формы в другую, отслеживает взаимосвязи между файлами. Для работы с утилитой make необходимо в корне рабочего каталога с Вашим проектом создать файл с названием makefile или Makefile, в котором будут описаны правила обработки файлов Вашего программного комплекса. Сначала задаётся список целей, разделённых пробелами, за которым идёт двоеточие и список зависимостей. Затем в следующих строках указываются команды. Строки с командами обязательно должны начинаться с табуляции. В качестве цели в Makefile может выступать имя файла или название какого-то действия. Зависимость задаёт исходные параметры (условия) для достижения указанной цели. Зависимость также может быть названием какого-то действия. Команды — собственно действия, которые необходимо выполнить для достижения цели.

## 3.2 Тестирование и отладка

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). Для использования GDB необходимо скомпилировать анализируемый код программы таким образом, чтобы отладочная информация содержалась в результирующем бинарном файле. Для этого следует воспользоваться опцией -g компилятора gcc: \* gcc -c file.c -g После этого для начала работы с gdb необходимо в командной строке ввести одноимённую команду, указав в качестве аргумента анализируемый бинарный файл: \* gdb file.o Затем можно использовать по мере необходимости различные команды gdb. Для выхода из gdb можно воспользоваться командой quit (или её сокращённым вариантом q) или комбинацией клавиш Ctrl-d . Более подробную информацию по работе с gdb можно получить с помощью команд gdb -h и man gdb.

# 4 Выполнение лабораторной работы

1. Я создала в домашнем каталоге подкаталог ~/work/os/lab\_prog, а затем создала в нём файлы: calculate.h, calculate.c, main.c. (рис. [1](#fig:001)).

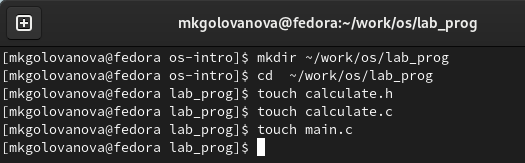


Figure 1: Создание подкаталога ~/work/os/lab\_prog и файлов calculate.h, calculate.c, main.c в нём

1. Я ввела в файл calculate.с текст для реализации функций калькулятора (рис. [2](#fig:002), рис. [3](#fig:003)).

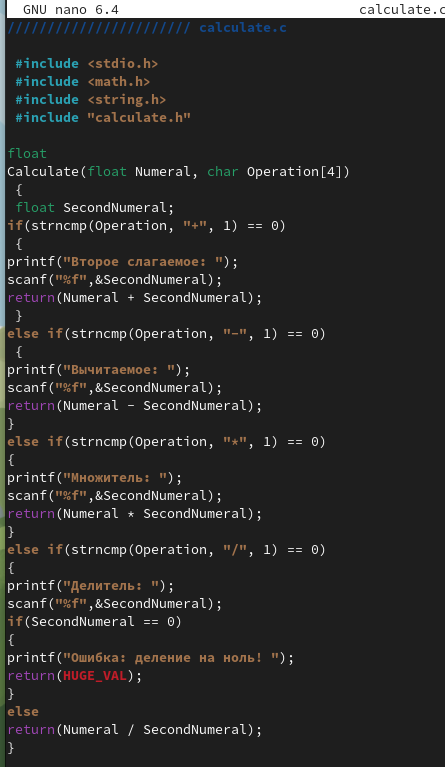


Figure 2: Текст файла calculate.с (1)



Figure 3: Текст файла calculate.с (2)

Я ввела текст интерфейсного файла calculate.h, описывающий формат вызова функции калькулятора (рис. [4](#fig:004)).

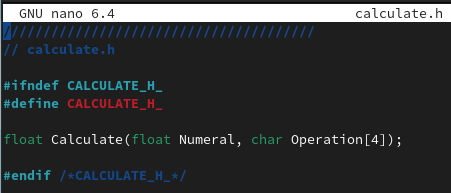


Figure 4: Текст интерфейсного файла calculate.h

Я ввела текст основного файла main.c, реализующий интерфейс пользователя к калькулятору (рис. [5](#fig:005)).



Figure 5: Текст основного файла main.c

1. Я выполнила компиляцию программы посредством gcc (рис. [6](#fig:006)).

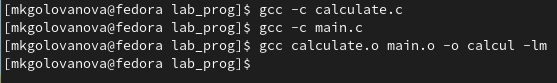


Figure 6: Компиляция программы посредством gcc

1. Я создала Makefile со следующим содержанием (рис. [7](#fig:007)).

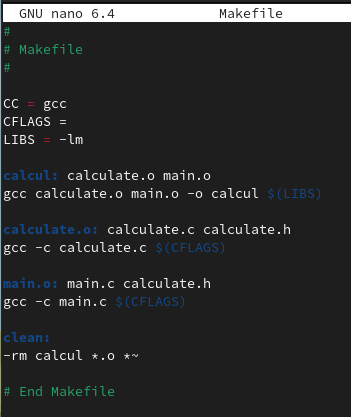


Figure 7: Текст файла Makefile

1. Я исправила Makefile (рис. [8](#fig:008)).

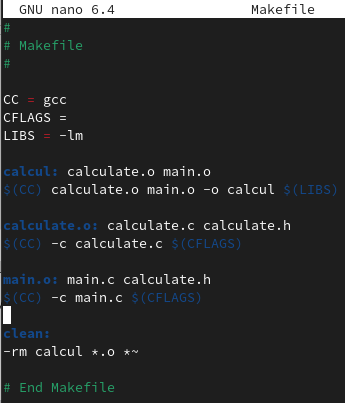


Figure 8: Исправленный Makefile

1. С помощью gdb я выполнила отладку программы calcul:

* Запустила отладчик GDB, загрузив в него программу для отладки (рис. [9](#fig:009)).

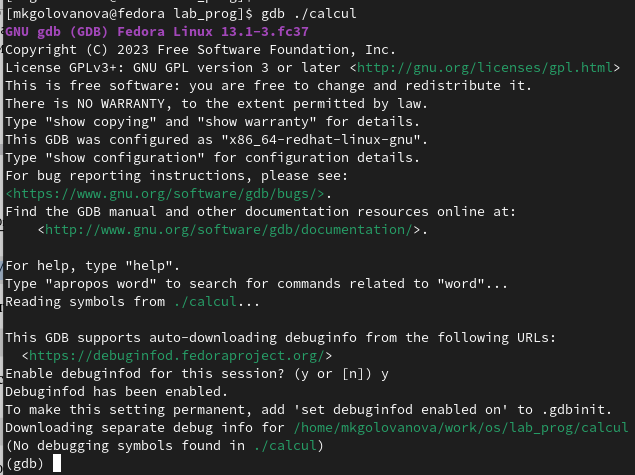


Figure 9: Запуск отладчика GDB

* Для запуска программы внутри отладчика я ввела команду run (рис. [10](#fig:010)).

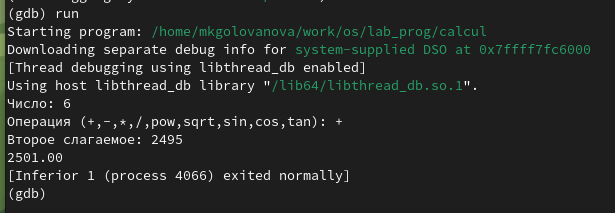


Figure 10: Запуск программы внутри отладчика с помощью команды run

* Для постраничного (по 9 строк) просмотра исходного кода я использовала команду list (рис. [11](#fig:011)).

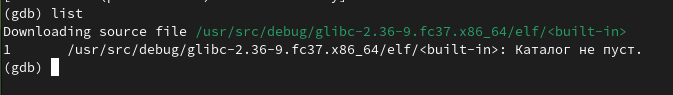


Figure 11: Команда list

* Для просмотра строк с 12 по 15 основного файла я использовала list с параметрами (рис. [12](#fig:012)).

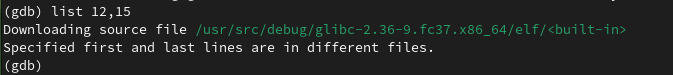


Figure 12: Команда list 12,15

* Для просмотра определённых строк не основного файла я использовала list с параметрами (рис. [13](#fig:013)).

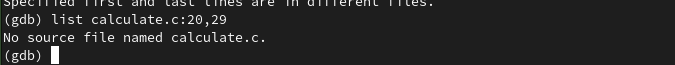


Figure 13: Команда list calculate.c:20,29

* Я установила точку останова в файле calculate.c на строке номер 21 и вывела информацию об имеющихся в проекте точка останова (рис. [14](#fig:014)).

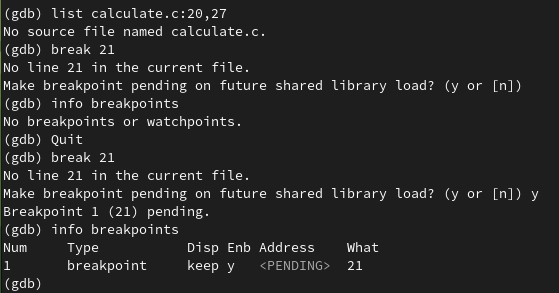


Figure 14: Установка точки останова в файле calculate.c на строке номер 21

* Я запустила программу внутри отладчика и убедилась, что программа остановится в момент прохождения точки останова (рис. [15](#fig:015)).

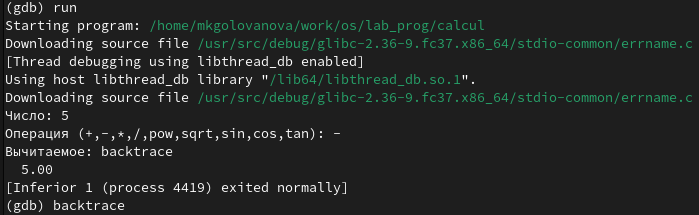


Figure 15: Запуск программы внутри отладчика

Отладчик выдал следующую информацию (рис. [16](#fig:016)).

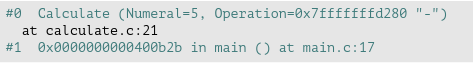


Figure 16: Результат работы откладчика

* Я посмотрела, чему равно на этом этапе значение переменной Numeral и сравнила с результатом вывода на экран после использования команды display Numeral (рис. [17](#fig:017)).

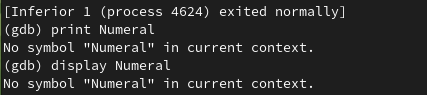


Figure 17: Результаты работы команды Numeral и команды display Numeral

* Я убрала точки останова (рис. [18](#fig:018)).

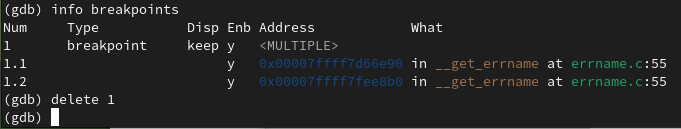


Figure 18: Удаление точек останова

1. С помощью утилиты splint я проанализировала коды файлов calculate.c и main.c (рис. [19](#fig:019), рис. [20](#fig:020)).

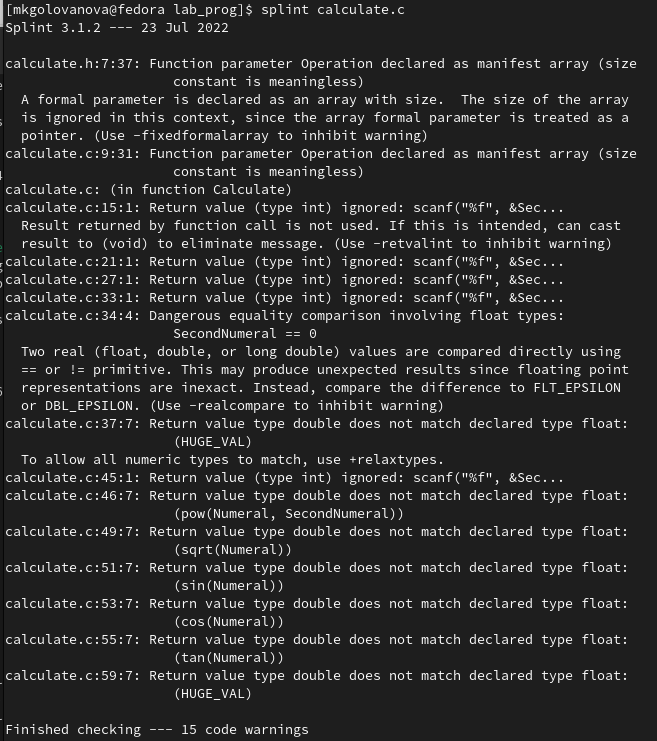


Figure 19: Команда splint calculate.c

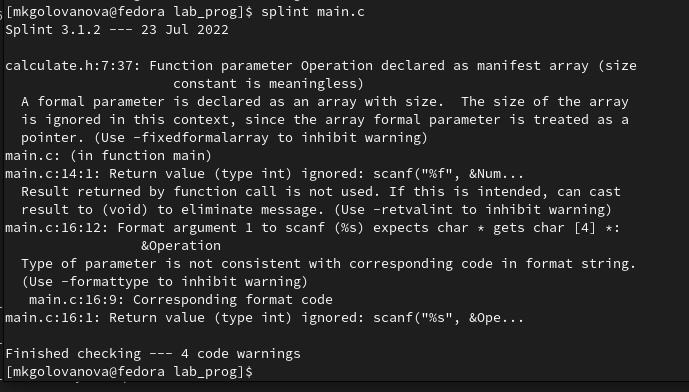


Figure 20: Команда splint main.c

# 5 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