## 13 Лабораторная работа

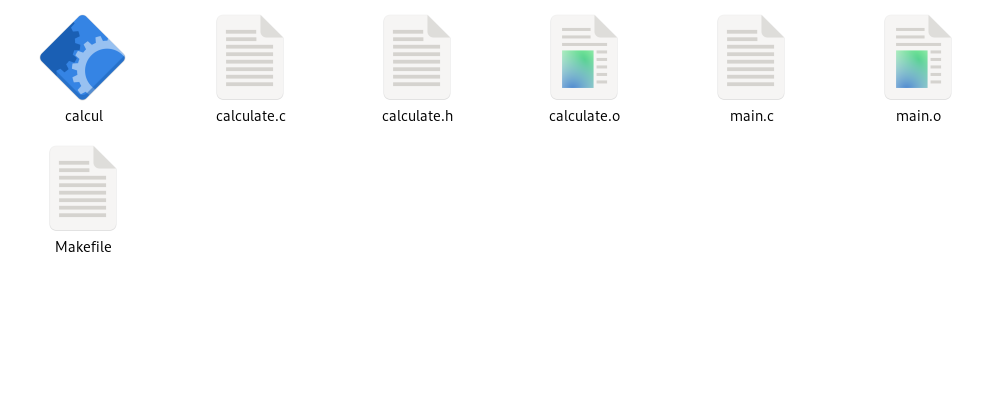
Прищепво Александр

## 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.

## Выполнение лабораторной работы

1. В домашнем каталоге создаем подкаталог ~/work/os/lab\_prog и в нем уже создаем три файла: calculate.h, calculate.c, main.c (рис. 1). 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.



изображение

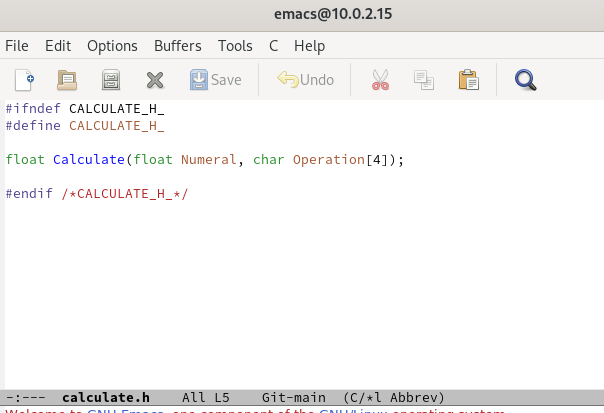
1. В созданных файлах напишем программы для работы калькулятора, которые нам предоставили (рис. 2), (рис. 3), (рис. 4).

рис 2:



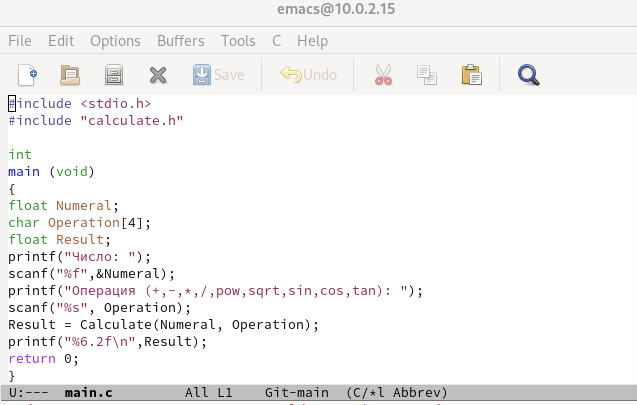
изображение

рис 3:



изображение

рис 4:

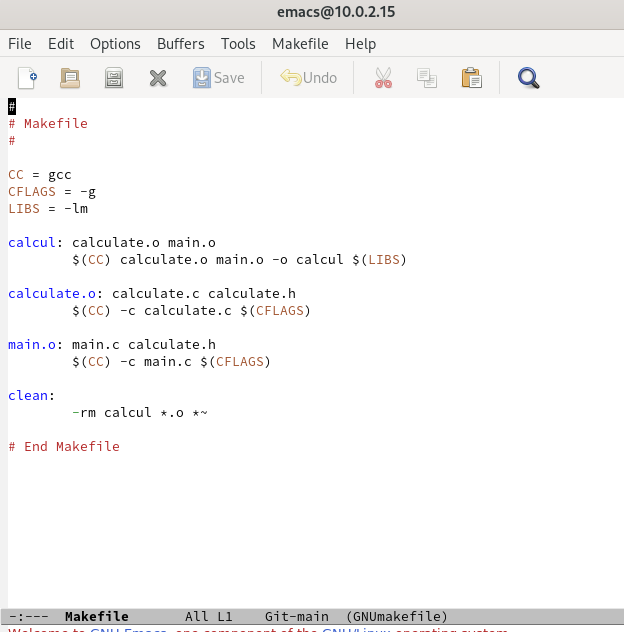


изображение

## 3. Выполним компиляцию программы посредством gcc и при необходимости исправим синтаксические ошибки

1. Создадим Makefile и введем в него предложенное содержимое (рис. 5).

рис 5:



изображение

Данный файл необходим для автоматической компиляции файлов calculate.c (цель calculate.o), main.c (цель main.o), а также их объединения в один исполняемый файл calcul (цель calcul). Цель clean нужна для автоматического удаления файлов. Переменная CC отвечает за утилиту для компиляции. Переменная CFLAGS отвечает за опции в данной утилите. Переменная LIBS отвечает за опции для объединения объектных файлов в один исполняемый файл.

## 5. Далее исправим Makefile. В переменную CFLAGS добавил опцию -g, необходимую для компиляции объектных файлов и их использования в программе отладчика GDB. Сделаем так, что утилита компиляции выбирается с помощью переменной CC.

После этого удалим исполняемые и объектные файлы из каталога с помощью команды make clean. Выполним компиляцию файлов, используя команды make calculate.o, make main.o, make calcul.

1. Далее с помощью команды gdb ./calcul запустим отладку программы

* Для запуска программы внутри отладчика введем команду run
* Для постраничного (по 9 строк) просмотра исходного код используем командуlist
* Для просмотра строк с 12 по 15 основного файла используем list с параметрами
* Для просмотра определённых строк не основного файла используем list с параметрами
* Установим точку останова в файле calculate.c на строке номер 18 и выведем информацию об имеющихся в проекте точка останова
* Запустим программу внутри отладчика и убедитесь, что программа остановится в момент прохождения точки останова
* Введем команду backtrace, которая покажет весь стек вызываемых функций от начала программы до текущего места
* Посмотрим, чему равно на этом этапе значение переменной Numeral, введя команду print Numeral и сравним с результатом команды display Numeral
* Уберем точки останова

## 7. С помощью утилиты splint проанализируем коды файлов calculate.c и main.c. Воспользуемся командами splint calculate.c и splint main.c.

C помощью утилиты splint выяснилось, что в файлах calculate.c и main.c присутствует функция чтения scanf, возвращающая целое число (тип int), но эти числа не используются и нигде не сохранятся. Утилита вывела предупреждение о том, что в файле calculate.c происходит сравнение вещественного числа с нулем. Также возвращаемые значения (тип double) в функциях pow, sqrt, sin, cos и tan записываются в переменную типа float, что свидетельствует о потери данных.

## Выводы

Здесь кратко описываются итоги проделанной работы.