Лабораторная работа №14

Данилова Анастасия Сергеевна

# 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 файлы, в которых будут коды данные в теоретическом материале. 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.
2. С помощью gdb выполнить отладку программы calcul.
3. С помощью утилиты splint проанализируем коды файлов calculate.c и main.c.

# Теоретическое введение

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.

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.

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

# Выполнение лабораторной работы

1. В домашнем каталоге создадим подкаталог ~/work/os/lab\_prog с помощью mkdir
2. Создадим в нём файлы: calculate.h, calculate.c, main.c с помощью touch

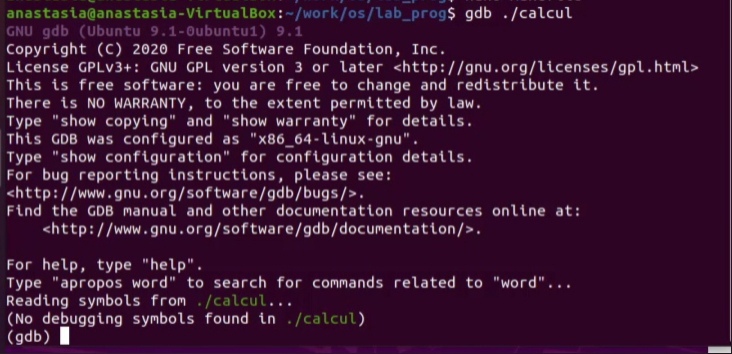
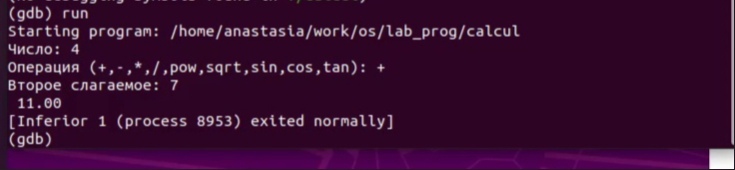
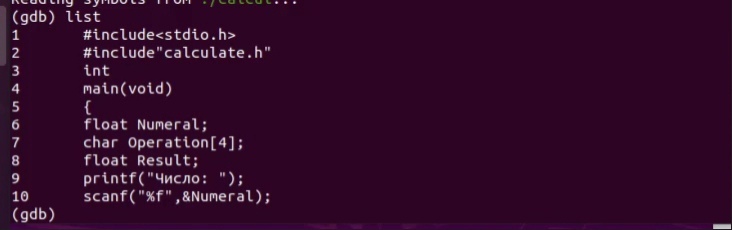
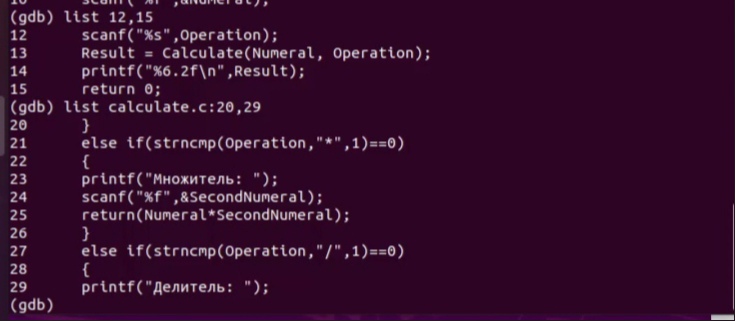
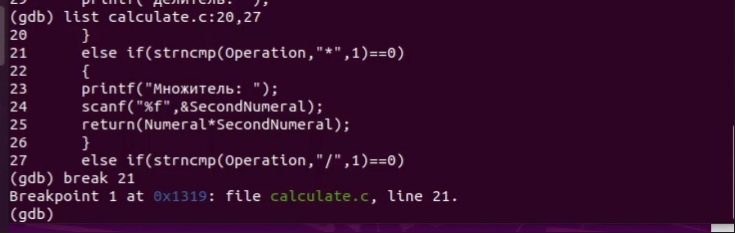
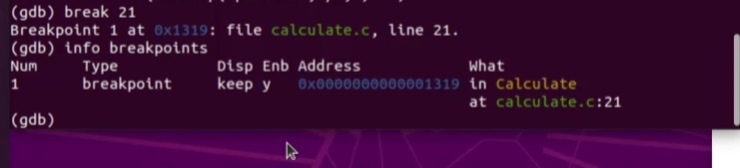
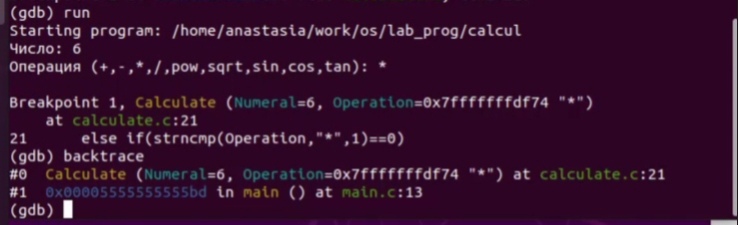
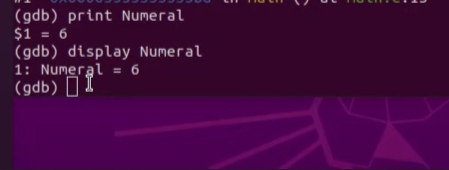
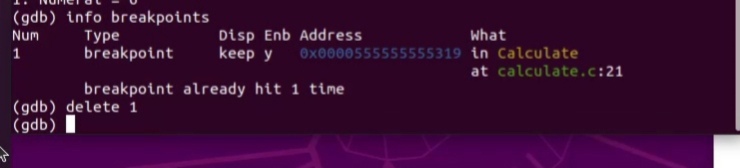
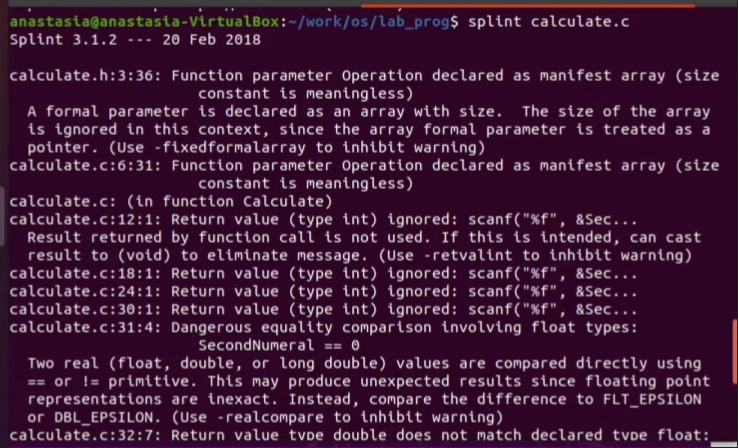
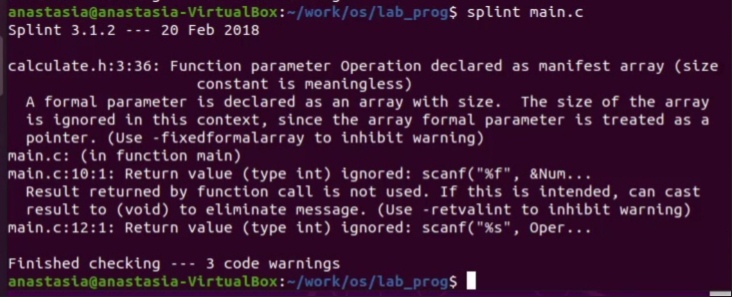
Запишем в эти файлы программы, данные нам. (рис1) (рис2) (рис3)

​ **рис 1** ​ **рис 2** ​ **рис 3**

1. Выполним компиляцию программы посредством gcc: gcc -c calculate.c gcc -c main.c gcc calculate.o main.o -o calcul -lm (рис4)

​ **рис 4**

1. Синтаксических ошибок не обнаружено
2. Создадим Makefile с данным нам кодом (рис 5) **рис 5**
3. С помощью gdb выполним отладку программы calcul:

* Запустим отладчик GDB, загрузив в него программу для отладки: gdb ./calcul (рис 6) 
* ​ **рис 6**
* Для запуска программы внутри отладчика введем команду run (рис 7)  **рис 7**
* Для постраничного просмотра исходного код используем команду list (рис 8)  **рис 8**
* Для просмотра строк с 12 по 15 основного файла используем list с параметрами: list 12,15 (рис 9)  **рис 9**
* Для просмотра определённых строк не основного файла используем list с параметрами: list calculate.c:20,29 (рис 9)
* Установим точку останова в файле calculate.c на строке номер 21: list calculate.c:20,27 break 21 (рис 10)  **рис 10**
* Выведем информацию об имеющихся в проекте точка останова: info breakpoints (рис 11)  **рис 11**
* Запустим программу внутри отладчика и убедимся, что программа остановится в момент прохождения точки останова: run 6 - backtrace (рис 12) **рис 12**
* Используем команду backtrace, которая покажет весь стек вызываемых функций от начала программы до текущего места. (рис 12)
* Посмотрим, чему равно на этом этапе значение переменной Numeral, введя: print Numeral. На экран должно быть выведено число 6. (рис 13)  **рис 13**
* Сравним с результатом вывода на экран после использования команды: display Numeral (рис 13)
* Уберем точки останова: info breakpoints delete(рис 14)  **рис 14**
  1. С помощью утилиты splint попробуйте проанализировать коды файлов calculate.c и main.c. (рис 15) (рис 16)  **рис 15**  **рис 16**

# Выводы

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

# Библиография

https://losst.ru/kak-polzovatsya-gdb