Отчёт по лабораторной работе №13

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

Надежда Александровна Рогожина

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

1. В домашнем каталоге создайте подкаталог ~/work/os/lab\_prog.
2. Создайте в нём файлы: calculate.h, calculate.c, main.c.

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.

1. Выполните компиляцию программы посредством gcc:

gcc -c -g calculate.c  
gcc -c -g main.c  
gcc calculate.o main.o -o calcul -lm

1. При необходимости исправьте синтаксические ошибки.
2. Создайте Makefile.
3. С помощью gdb выполните отладку программы calcul (перед использованием gdb исправьте Makefile):

* Запустите отладчик GDB, загрузив в него программу для отладки:
* Для запуска программы внутри отладчика введите команду run
* Для постраничного (по 9 строк) просмотра исходного код используйте команду list
* Для просмотра строк с 12 по 15 основного файла используйте list с параметрами
* Для просмотра определённых строк не основного файла используйте list с параметрами
* Установите точку остановки в файле calculate.c на строке номер 21
* Выведите информацию об имеющихся в проекте точка остановки
* Запустите программу внутри отладчика и убедитесь, что программа остановится в момент прохождения точки остановки
* Посмотрите, чему равно на этом этапе значение переменной Numeral
* Сравните с результатом вывода на экран после использования команды
* Уберите точки остановки

1. С помощью утилиты splint попробуйте проанализировать коды файлов calculate.c и main.c.

# 3 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.

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.

Например, в табл. [1](#tbl:std-dir) приведено описание некоторых опций gcc.

Table 1: Некоторые опции компиляции в gcc

| Опция | Описание |
| --- | --- |
| -с | Компиляция без компоновки — создаются объектные файлы file.o |
| -o file-name | Задать имя file-name создаваемому файлу |
| -g | Поместить в файл (объектный или исполняемый) отладочную информацию для отладчика gdb |
| -MM | Вывести зависимости от заголовочных файлов C и/или C++ программ в формате, подходящем для утилиты make; при этом объектные или исполняемые файлы не будет созданы |
| -Wall | Вывод на экран сообщений об ошибках, возникших во время компиляции |

# 4 Выполнение лабораторной работы

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

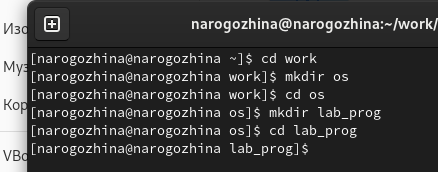


Figure 1: Создание подкаталога

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

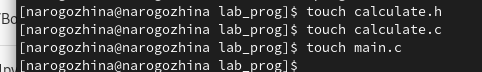


Figure 2: Создание файлов

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 (рис. [3](#fig:003), [4](#fig:004), [5](#fig:005)).

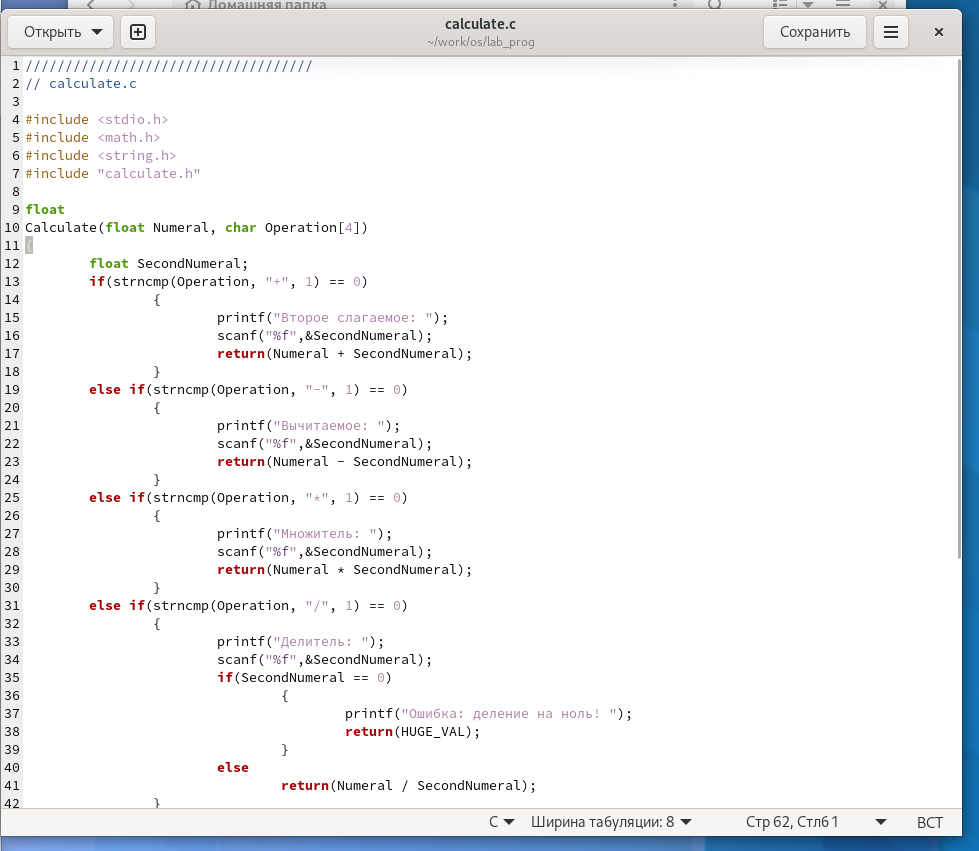


Figure 3: calculate.c

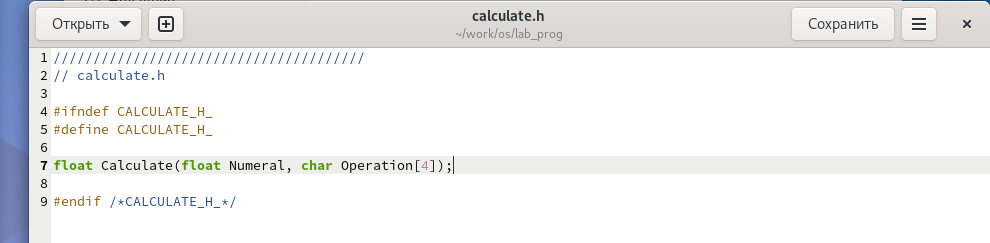


Figure 4: calculate.h



Figure 5: main.c

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

gcc -c -g calculate.c  
gcc -c -g main.c  
gcc calculate.o main.o -o calcul -lm

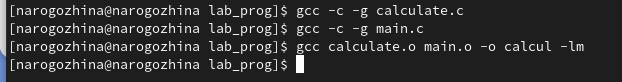


Figure 6: Компиляция программы

1. При необходимости исправьте синтаксические ошибки.

Ошибок не было :)

1. Создайте Makefile (рис. [7](#fig:007)).

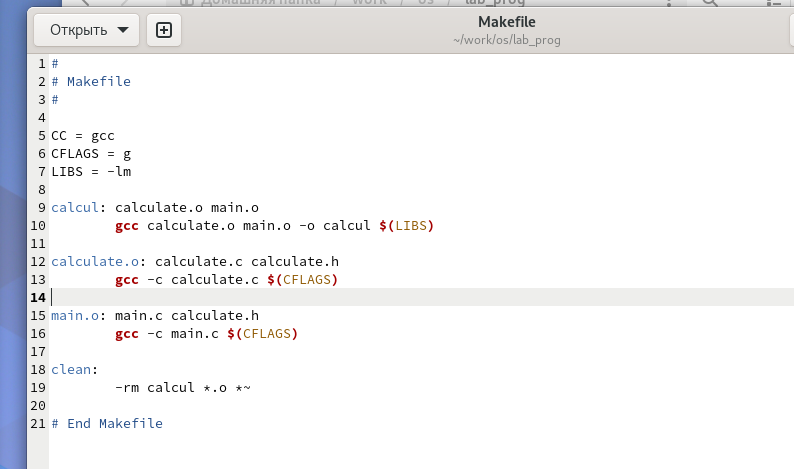


Figure 7: Создание Makefile

В данном Makefile целями являются:

* calcul - зависит от: calculate.o, main.o
* calculate.o - зависит от: calculate.c, calculate.h
* main.o - зависит от: main.c, calculate.h

На строчках 10, 13, 16 прописаны команды, которые относятся к 9,12 и 15 строчкам соответственно.

1. С помощью gdb выполните отладку программы calcul (перед использованием gdb исправьте Makefile):

Первоначально я запустила Makefile (рис. [8](#fig:008)), но так как обновления не требовалось, приступила к выполнению дальнейшего задания.

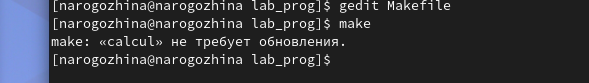


Figure 8: make

* Запустите отладчик GDB, загрузив в него программу для отладки. Для запуска программы внутри отладчика введите команду run (рис. [9](#fig:009)):

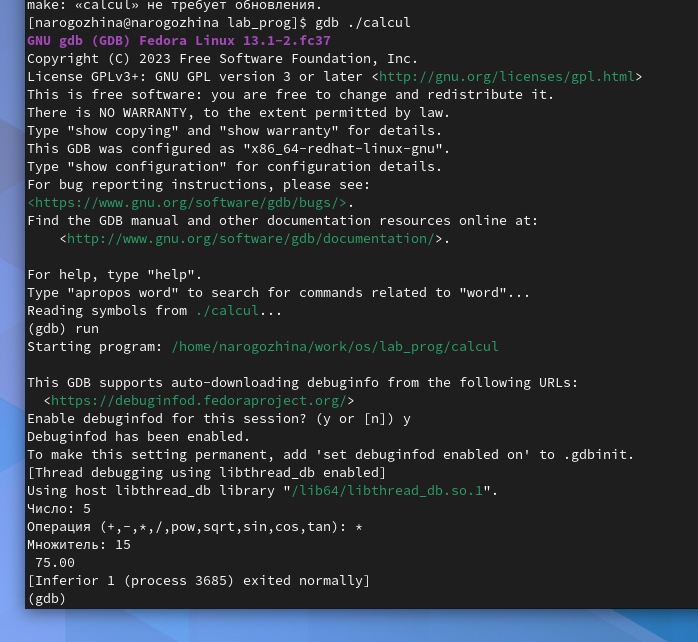


Figure 9: Запуск gdb

* Для постраничного (по 9 строк) просмотра исходного код используйте команду list (рис. [10](#fig:010)):

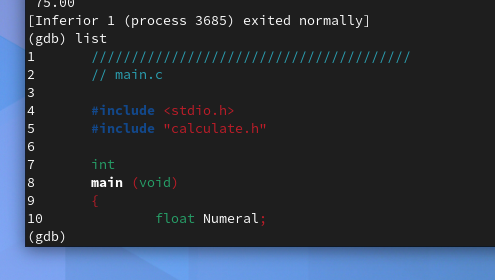


Figure 10: list

* Для просмотра строк с 12 по 15 основного файла используйте list с параметрами (рис. [11](#fig:011)):

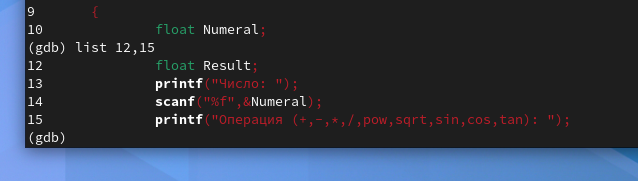


Figure 11: list 12,15

* Для просмотра определённых строк не основного файла используйте list с параметрами (рис. [12](#fig:012)):

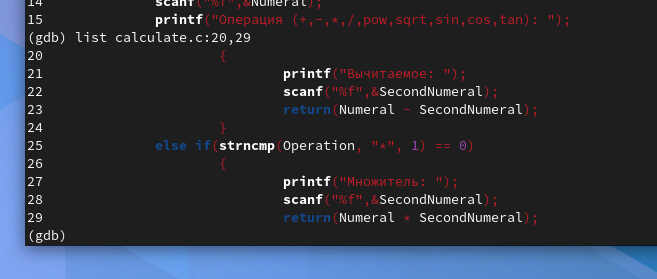


Figure 12: list calculate.c:20,29

* Установите точку остановки в файле calculate.c на строке номер 21 (рис. [13](#fig:013)):

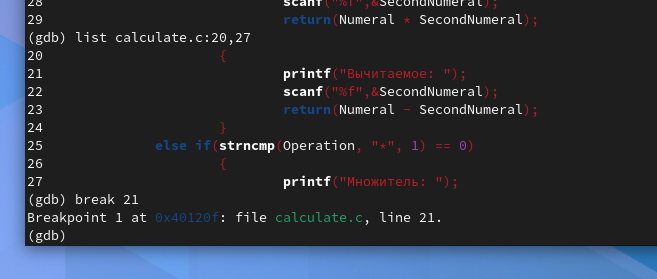


Figure 13: Точка остановки

* Выведите информацию об имеющихся в проекте точках остановки (рис. [14](#fig:014)):

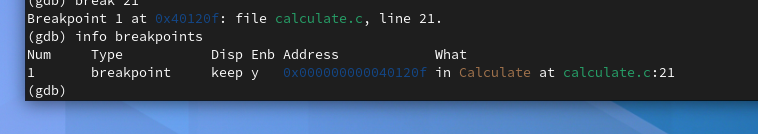


Figure 14: Точки остановки

* Запустите программу внутри отладчика и убедитесь, что программа остановится в момент прохождения точки остановки (рис. [15](#fig:015)):

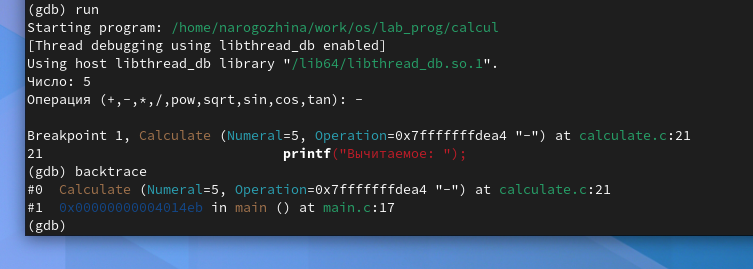


Figure 15: run + backtrace

* Посмотрите, чему равно на этом этапе значение переменной Numeral (рис. [16](#fig:016)):

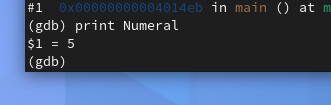


Figure 16: print Numeral

* Сравните с результатом вывода на экран после использования команды (рис. [17](#fig:017)):

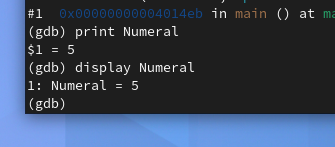


Figure 17: display Numeral

Результаты вывода обеих команд одинаковы.

* Уберите точки остановки (рис. [18](#fig:018)):

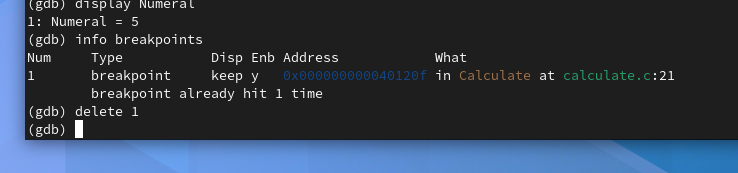


Figure 18: Удаление точек остановки

После я решила проверить удаление точек остановки и запустила программу снова (рис. [19](#fig:019)):

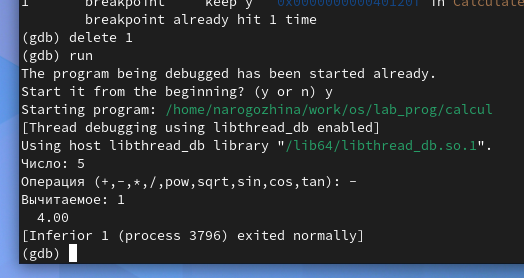


Figure 19: run

1. С помощью утилиты splint попробуйте проанализировать коды файлов calculate.c и main.c (рис. [20](#fig:020), [21](#fig:021)):

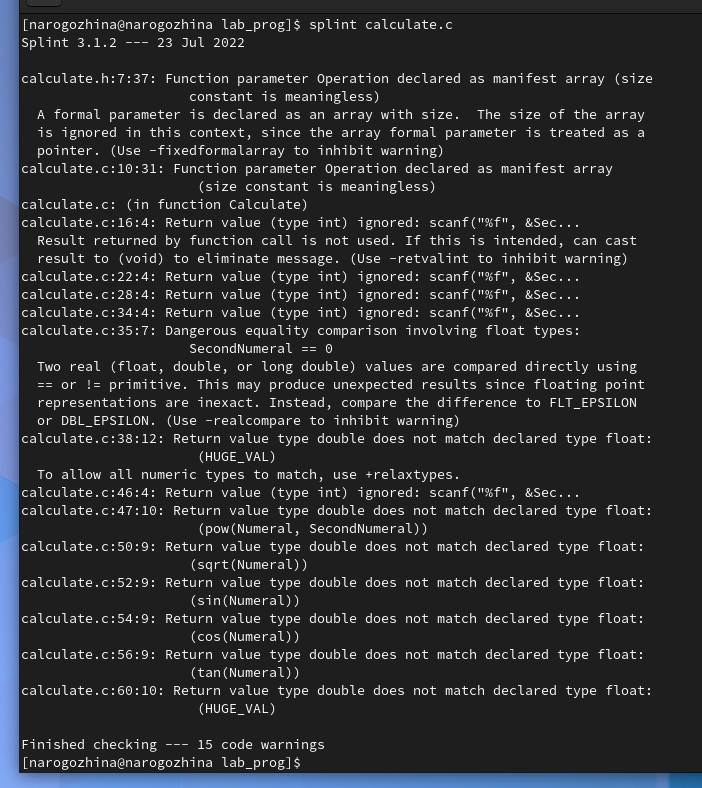


Figure 20: splint calculate.c

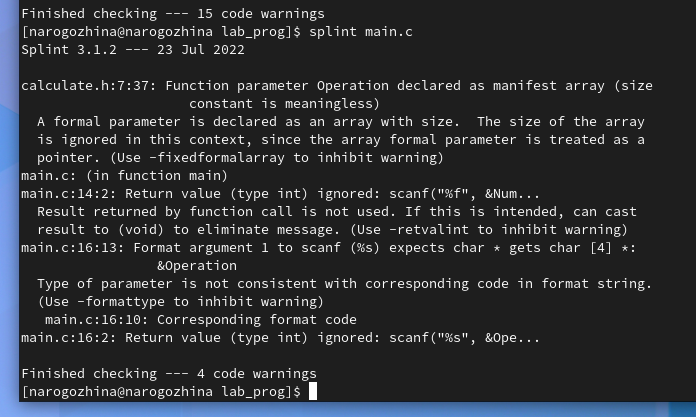


Figure 21: splint main.c

# 5 Выводы

В ходе лабораторной работы 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.

# Список литературы

1. [Руководство по выполнению лабораторной работы №13](https://esystem.rudn.ru/pluginfile.php/1975785/mod_resource/content/5/013-lab_prog.pdf)
2. [Руководство пользования gdb](http://dmilvdv.narod.ru/Translate/LDD3/ldd_debuggers_related_tools.html)
3. [man splint](https://linux.die.net/man/1/splint)