РУДН. Операционные системы

Отчёт по лабораторной работе №12

Косинов Никита Андреевич, НПМбв-02-20

Содержание

# 1 Цель работы

Познакомиться с основными принципами написания, отладки, тестирования приложения в **ОС Linux** на примере разработки приложения “Простейший калькулятор” на языке **С**.

# 2 Ход работы

Лабораторная работа выполнена в терминале **OC Linux**, командной оболочке **bash**, отладчике **gdb** и хостинге хранения проектов **Github**.

Действия по лабораторной работе представлены в виде последовательных шагов.

По завершении отчёта, вся рабочая папка отправляется на репозиторий на *github*.

# 3 Написание программы “Простейший калькулятор”

Перед началом работы создадим новый рабочий каталог **lab12** и перейдём внутрь. Также не забываем синхронизироваться с нашим **Git**.

1. В первую очередь перейдём в рабочую директорию и создадим необходимые файлы.

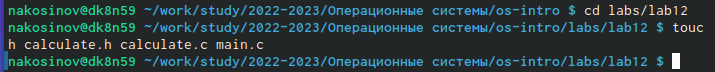


Рис. 1: Создание файлов

1. Напишем необходимый код для нашего приложения.

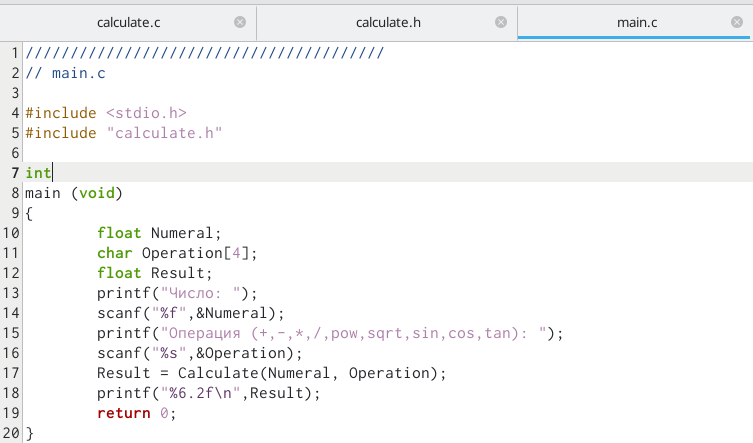


Рис. 2: Код основного файла

1. Сконфигурируем написанные программы с помощью компилятора **gcc**.

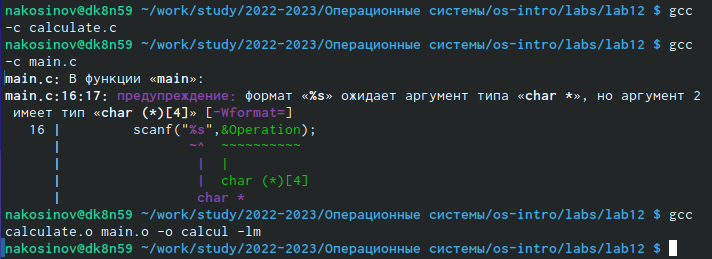


Рис. 3: Компиляция файлов

1. Увидим, что после компиляции появились соответствующие исполняемые файлы.

Исполняемые файлы

Рис. 4: Исполняемые файлы

1. Создадим конфигурирующий *Makefile*.

Создание файла make

Рис. 5: Создание файла make

1. Пропишем в *Makefile* следующий код:

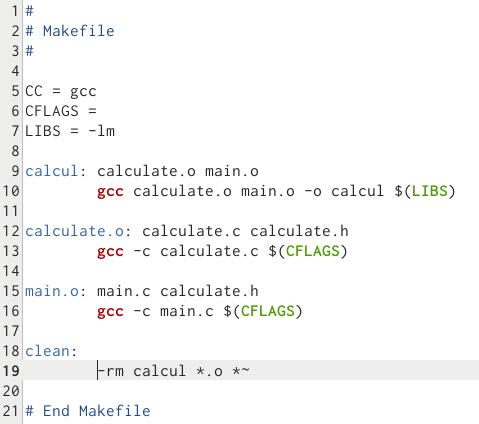


Рис. 6: Код файла make

Здесь синим подсвечены цели, после двоеточия указаны зависимости для этих целей, далее идут команды.

1. Запустим отладчик **gdb**

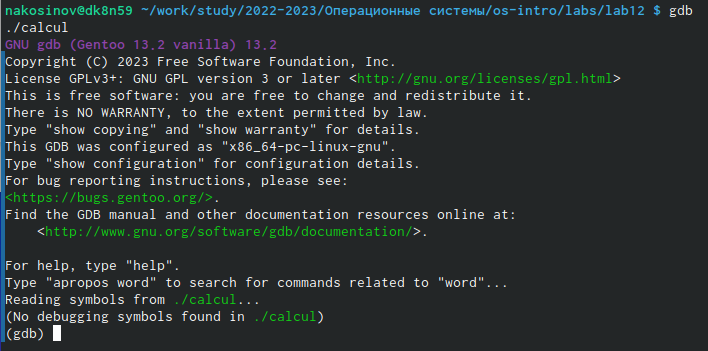


Рис. 7: Отладка приложения

1. Запустим программу внутри отладчика для проверки корректности её работы.

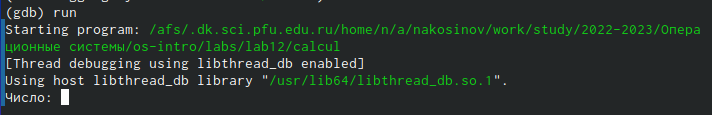


Рис. 8: Запуск программы внутри отладчика

1. Запускаем программу несколько раз, чтобы проверить работоспособность каждой функции нашего калькулятора. Запустить пришлось 9 раз, по одному на каждую команду, причём для операции деления дважды, чтобы проверить обработку исключения “деление на ноль”.

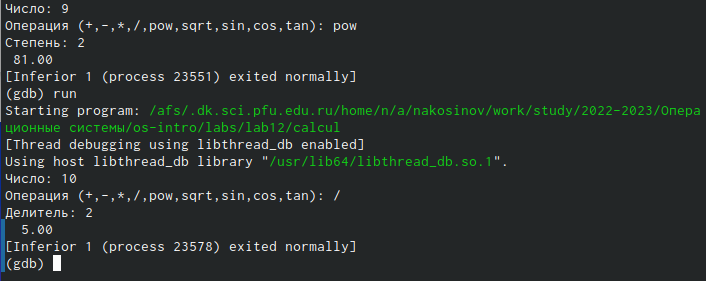


Рис. 9: Проверка двух операций

1. Поставим точку останова командой *break* на 21-ю строку файла *calculate.c*, соответствующую моменту ввода вычитаемого. Выбираем ответ *y* (yes).

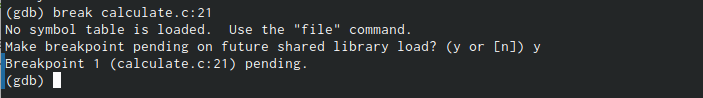


Рис. 10: Установка точки останова

1. Проверим наличие точки останова, выведя информацию о всех таких командой *info breakpoints*.

|  |
| --- |
| Информация о точках останова |

Рис. 11: Информация о точках останова

1. Удалим созданную точку остановы командой *delete*. Снова выведем информацию о точках, чтобы убедиться в корректности выполненного удаления.

|  |
| --- |
| Удаление точки остановы |

Рис. 12: Удаление точки остановы

1. Выход из отладчика производится командой *quit(q)*.

|  |
| --- |
| Выход из отладчика gdb |

Рис. 13: Выход из отладчика gdb

1. Проанализируем код программы, используя утилиту *splint*.

|  |
| --- |
| Анализ кода программы утилитой splint |

Рис. 14: Анализ кода программы утилитой splint

1. Видим, что эта утилита нашла 15 ошибок. Координаты ошибки даются после названия файла, далее идёт описание ошибки. Например, та, что идёт первой относится к связанному файлу *calculate.h*: 7 строка, 37 символ, ошибка связана с перемнной *Operation[4]*. **Splint** утверждает, что бессмысленно указывать размер константы.

# 4 Выводы

Мы познакомились с отладчиком приложений **gdb** и более требовательным **splint**, протестировали написанное приложение “Простейший калькулятор”, научились ставить точки остановы.