Отчёт по лабораторной работе №12:

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

Касакьянц Владислав Сергеевич

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.
3. Выполните компиляцию программы посредством gcc.
4. Создайте Makefile и поясните о его содержании.
5. С помощью gdb выполните отладку программы calcul (перед использованием gdb исправьте Makefile)
6. С помощью утилиты splint попробуйте проанализировать коды файлов calculate.c и main.c.

# 3 Выполнение лабораторной работы

1. В домашнем каталоге создаю новый подкаталог ~/work/os/lab\_prog, перехожу в него и создаю 3 файла: calculate.h, calculate.c, main.c (рис. 1):

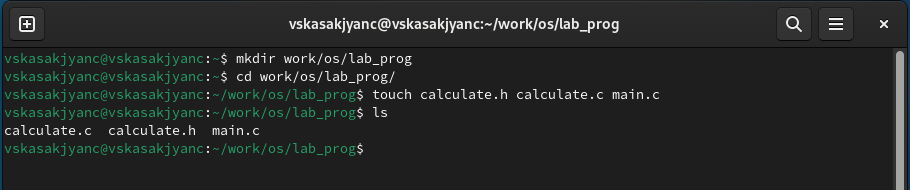


Рис. 1: Создание нового подкаталога и файлов

1. Запишем в файлы тексты программ, которые даны в лабораторной работе (рис. 2), (рис. 3), (рис. 4).

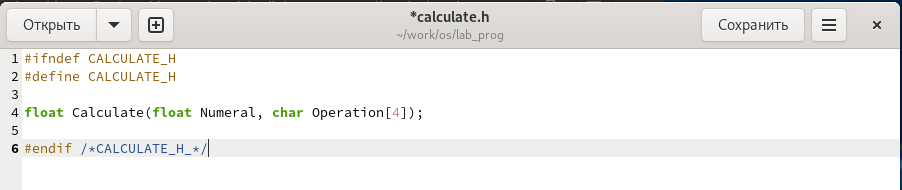


Рис. 2: calculate.h

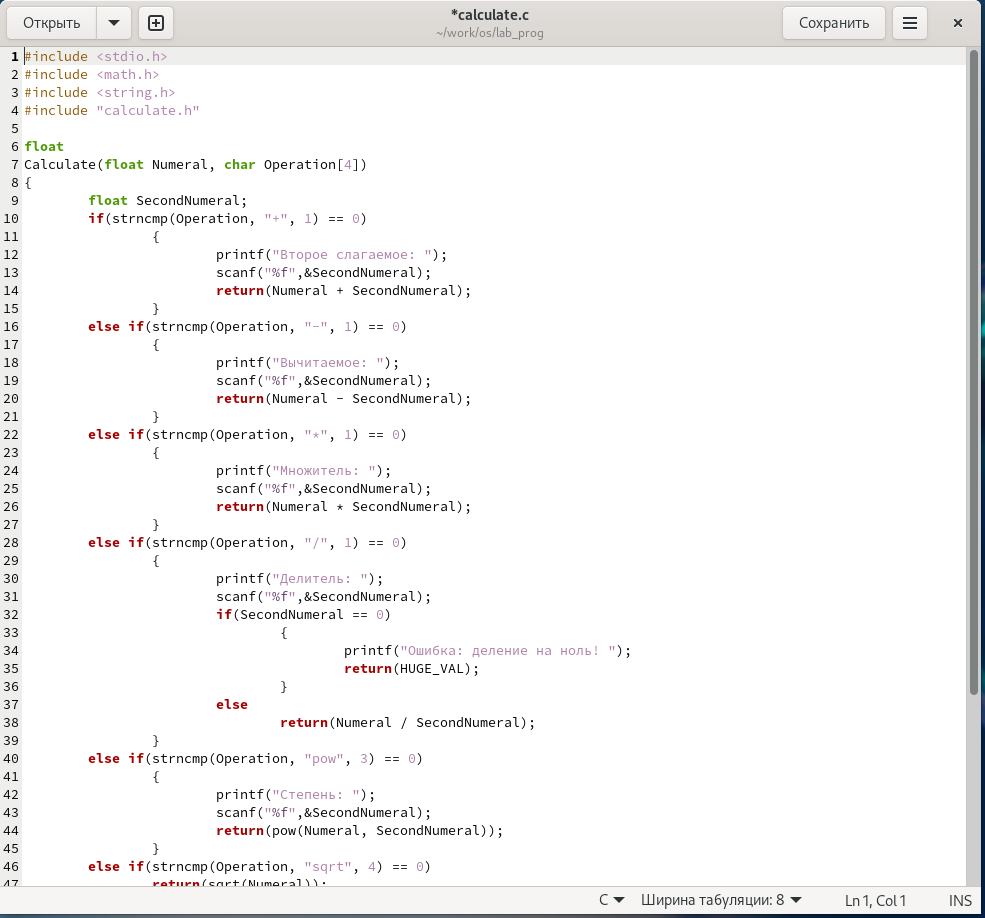


Рис. 3: calculate.c



Рис. 4: main.c

1. Выполним компиляцию программы посредством gcc (рис. 5):

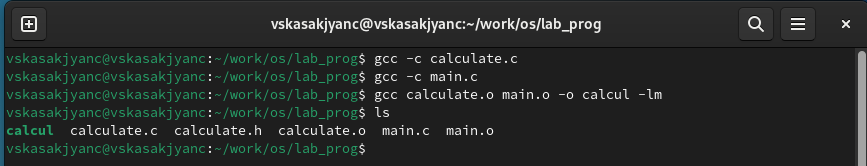


Рис. 5: компиляция программы посредством gcc

1. Создим Makefile (рис. 6):

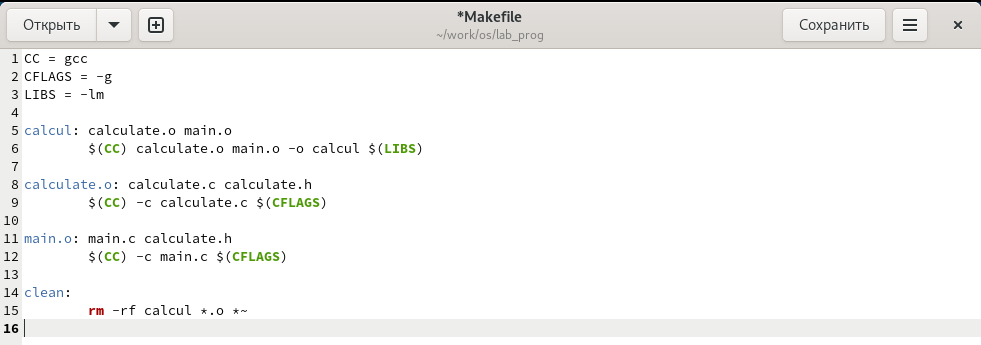


Рис. 6: Makefile

1. Запустим отладчик GDB, загрузив в него программу для отладки. Запустим программу и посчитаем некое выражение (рис. 7)

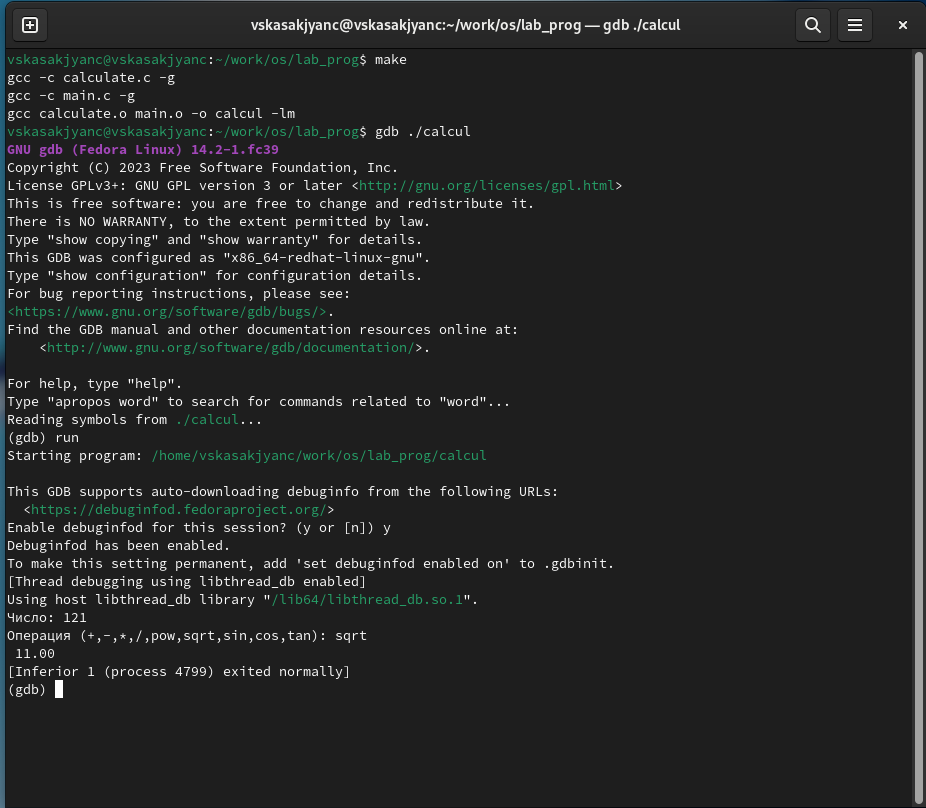


Рис. 7: Запуск отладчика

Для постраничного (по 9 строк) просмотра исходного код используем команду list, затем для просмотра строк с 12 по 15 основного файла используем list 12,15, просмотрим определённых строк не основного файла, используя list calculate.c:20,29, а также установим точку останова в файле calculate.c на строке номер 21, использовав list calculate.c:20,27 и break 21 (рис. 8):

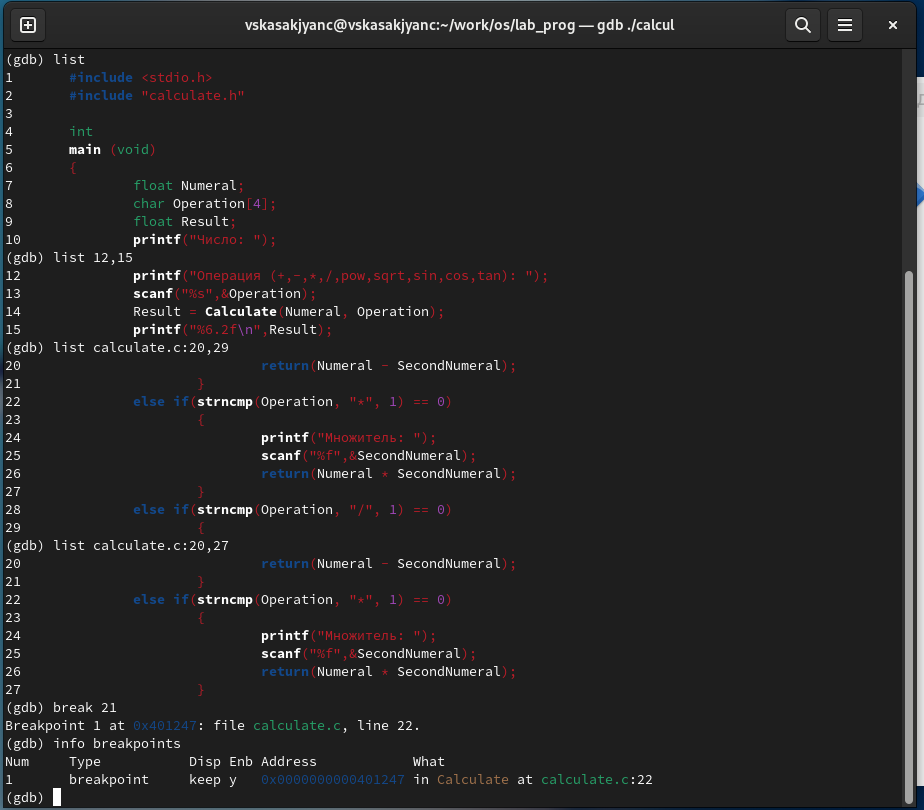


Рис. 8: Просмотр кода и точка остановки

Запустим программу внутри отладчика с помощью run и убедимся, что программа остановится в момент прохождения точки останова. С помощью команды backtrac покажим весь стек вызываемых функций от начала программы до текущего места. Посмотрим, чему равно на этом этапе значение переменной Numeral, введя print Numeral и сравним с результатом вывода на экран после использования команды, использовав display Numeral. Посмотрим, информацию про точку останова с помощью info breakpoints и удалим эту точку командой delete 1 (рис. 9):

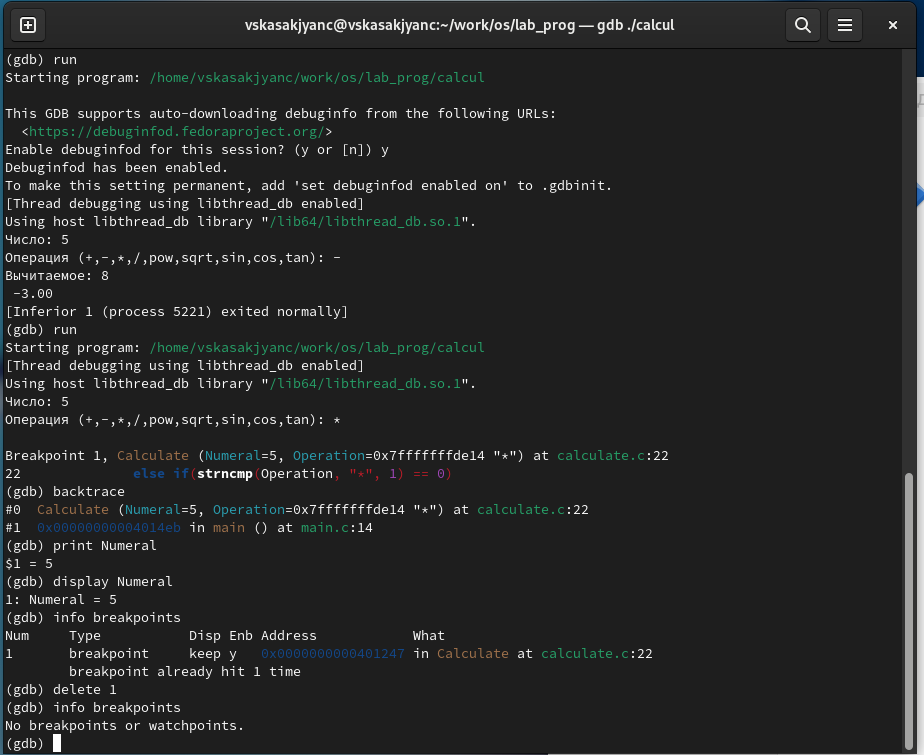


Рис. 9: Проверка остановки и удаление точки остановы

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

В файле main.c всего 3 предупреждения (рис. 10).

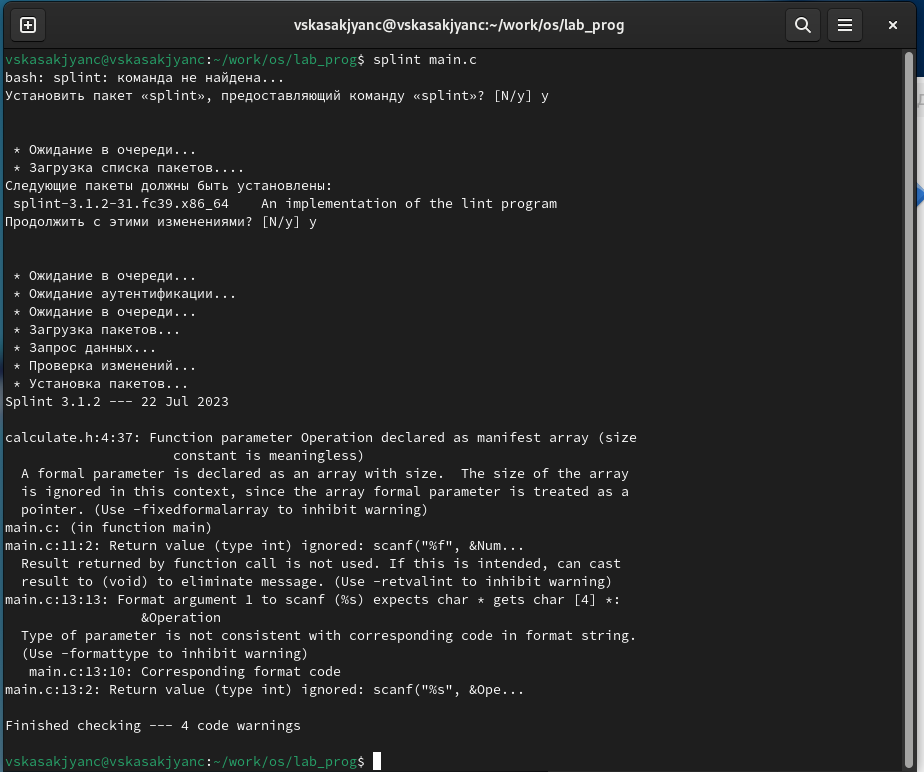


Рис. 10: Анализ кода файла main.c

А в файлу calculate.c всего 15 предупреждений (рис. 11).

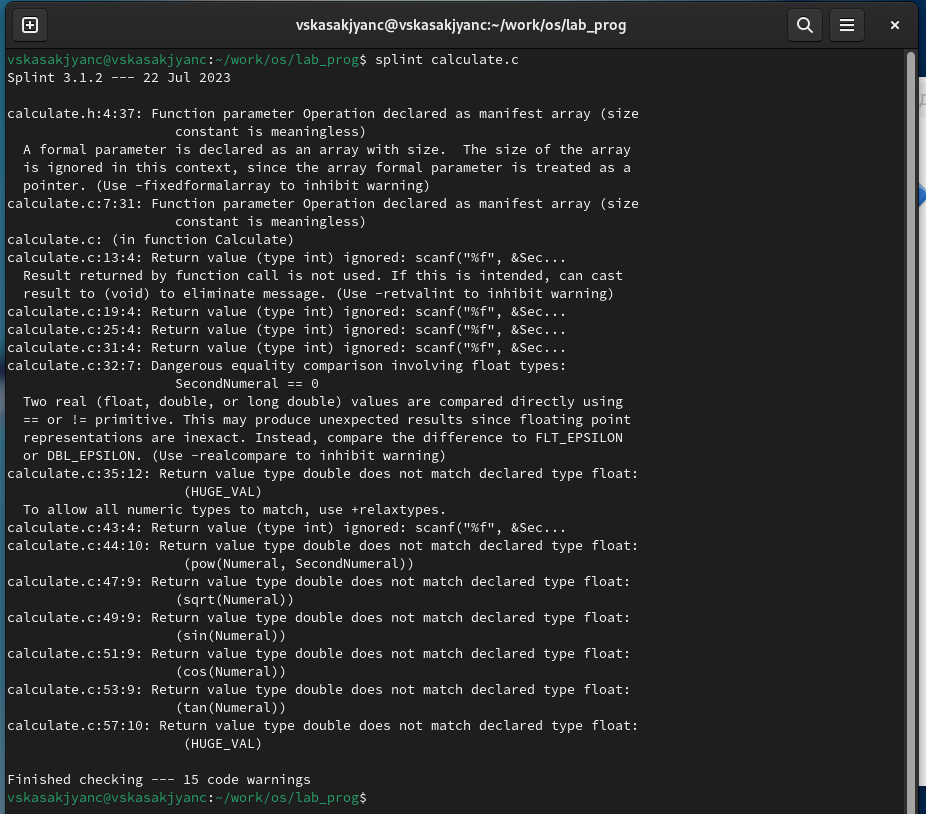


Рис. 11: Анализ кода файла calculate.c

# 4 Контрольные вопросы

1. Как получить информацию о возможностях программ gcc, make, gdb и др.?

Можно использовать название\_программы --help для общей помощи, man название\_программы для руководства пользователя или info название\_программы для более подробной информации.

1. Назовите и дайте краткую характеристику основным этапам разработки приложений в UNIX.

* **Дизайн**: Определение требований и архитектуры системы.
* **Кодирование**: Написание исходного кода приложения.
* **Компиляция**: Преобразование исходного кода в исполняемый файл.
* **Тестирование**: Проверка функциональности и поиск ошибок.
* **Отладка**: Исправление обнаруженных ошибок.
* **Установка**: Размещение программы в системе для использования.
* **Сопровождение**: Обновление и улучшение программы.

1. Что такое суффикс в контексте языка программирования? Приведите примеры использования.

Суффикс — это расширение файла, указывающее на тип содержимого. Например, .c для исходных файлов C, .h для заголовочных файлов C.

1. Каково основное назначение компилятора языка С в UNIX?

Компилятор C преобразует исходный код на языке C в машинный код, который может выполняться операционной системой UNIX.

1. Для чего предназначена утилита make?

make автоматизирует процесс компиляции и сборки программы, используя файл Makefile для определения зависимостей между файлами и правил сборки.

1. Приведите пример структуры Makefile. Дайте характеристику основным элементам этого файла.

**Пример структуры Makefile**:

all: program  
  
 program: main.o lib.o  
 gcc -o program main.o lib.o  
  
 main.o: main.c  
 gcc -c main.c  
  
 lib.o: lib.c  
 gcc -c lib.c  
  
 clean:  
 rm -f \*.o program

**Элементы Makefile**:

* **Цели**: all, program, main.o, lib.o, clean.
* **Зависимости**: Файлы, от которых зависит цель.
* **Правила**: Команды для создания цели из зависимостей.
* **Псевдоцели**: Цели, не связанные с файлами, например clean.

1. Назовите основное свойство, присущее всем программам отладки. Что необходимо сделать, чтобы его можно было использовать?

Возможность остановить выполнение программы, просмотреть и изменить значения переменных. Для использования требуется скомпилировать программу с опцией отладки (например, gcc -g).

1. Назовите и дайте основную характеристику основным командам отладчика gdb.

**Основные команды gdb**:

* run: Запуск программы.
* break: Установка точки останова.
* next: Выполнение следующей строки кода.
* continue: Продолжение выполнения до следующей точки останова.
* print: Вывод значения переменной.
* quit: Выход из gdb.

1. Опишите по шагам схему отладки программы, которую Вы использовали при выполнении лабораторной работы.

* Компиляция с опцией -g.
* Запуск gdb.
* Установка точек останова.
* Запуск программы с помощью run.
* Просмотр и изменение переменных.
* Продолжение выполнения и наблюдение за поведением программы.

1. Прокомментируйте реакцию компилятора на синтаксические ошибки в программе при его первом запуске.

Компилятор выдаст сообщения об ошибках, указывая местоположение и возможную причину ошибки.

1. Назовите основные средства, повышающие понимание исходного кода программы.

* Комментарии.
* Читаемые имена переменных и функций.
* Структурирование кода.
* Документация.

1. Каковы основные задачи, решаемые программой splint?

splint выполняет статический анализ кода на C для обнаружения ошибок программирования, уязвимостей безопасности и некачественного кода.

# 5 Выводы

В данной лабораторной работе 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