Лабораторная работа №13

Дисциплина: Операционные системы

Кондратьев Арсений Вячеславович

25.09.2022

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 Теоретическое введение

Процесс разработки программного обеспечения обычно разделяется на следующие этапы:  
– планирование, включающее сбор и анализ требований к функционалу и другим характеристикам разрабатываемого приложения;  
– проектирование, включающее в себя разработку базовых алгоритмов и спецификаций, определение языка программирования;  
– непосредственная разработка приложения:  
– кодирование — по сути создание исходного текста программы (возможно в нескольких вариантах);  
– анализ разработанного кода;  
– сборка, компиляция и разработка исполняемого модуля;  
– тестирование и отладка, сохранение произведённых изменений;  
– документирование.

# 3 Выполнение лабораторной работы

1. Создал подкаталог /lab\_prog в котором создал файлы: calculate.h, calculate.c, main.c и написал в них реализацию простейшего калькулятора.(рис.1)

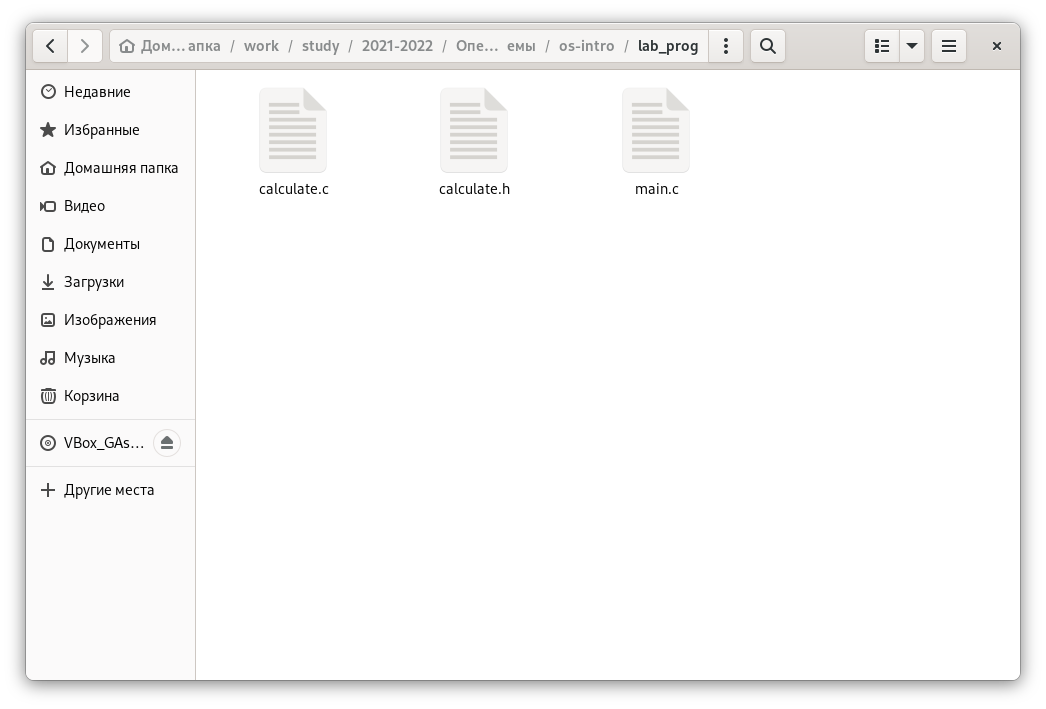


Figure 1: Рис. 1

1. Выполнил компиляцию программы посредством gcc(рис.2)

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

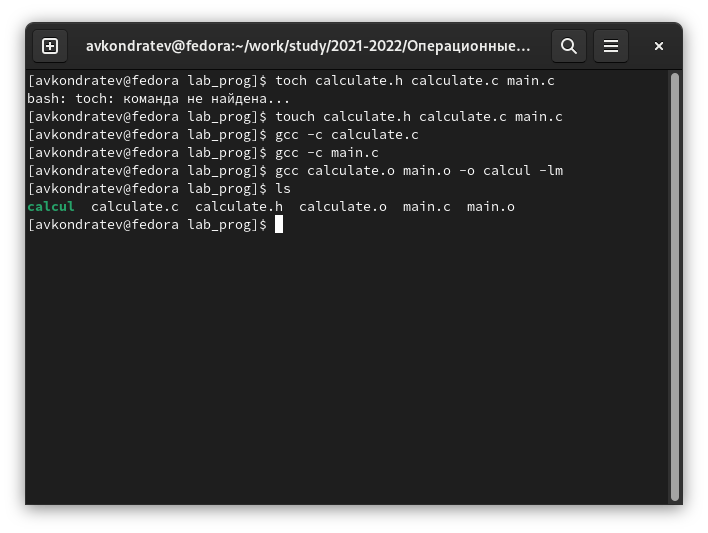


Figure 2: Рис. 2

1. Создал Makefile в котором дописал опцию -g в СFLAGS для корректной компиляции объектных файлов и использовал переменную СС в которую помещен компилятор. Он необходим для автоматической компиляции calculate.c, main.c и объединения в один исполняемый файл calcul. Сlean нужна для автоматического удаления файлов. Переменная CFLAGS отвечает за опции. Переменная LIBS отвечает за опции для объединения.(рис.3)

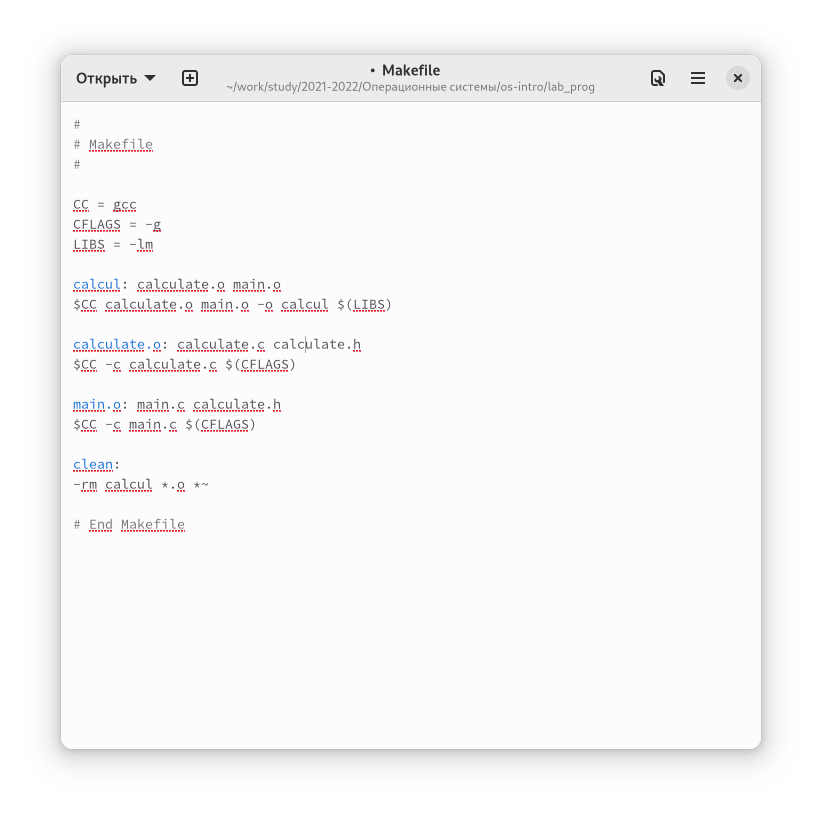


Figure 3: Рис. 3

1. Запустил отладчик GDB, загрузив в него программу для отладки(рис.4)

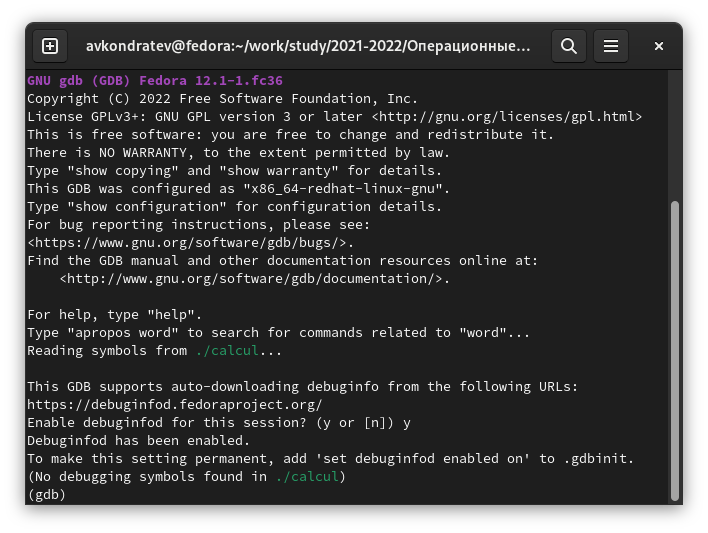


Figure 4: Рис. 4

1. Запустил программу внутри отладчика(рис.5)

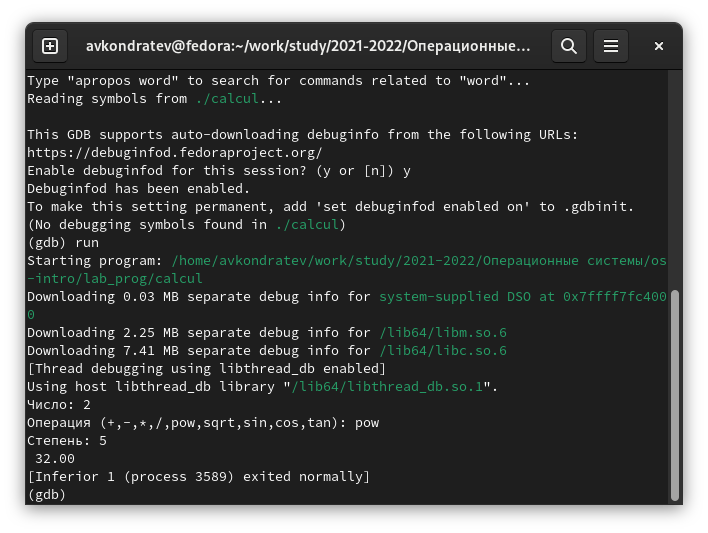


Figure 5: Рис. 5

1. Использовал команду list для постраничного просмотра кода, затем просмотрел код с 12 по 15 строку, затем просмотрел определенные строки не основного файла(рис.6)

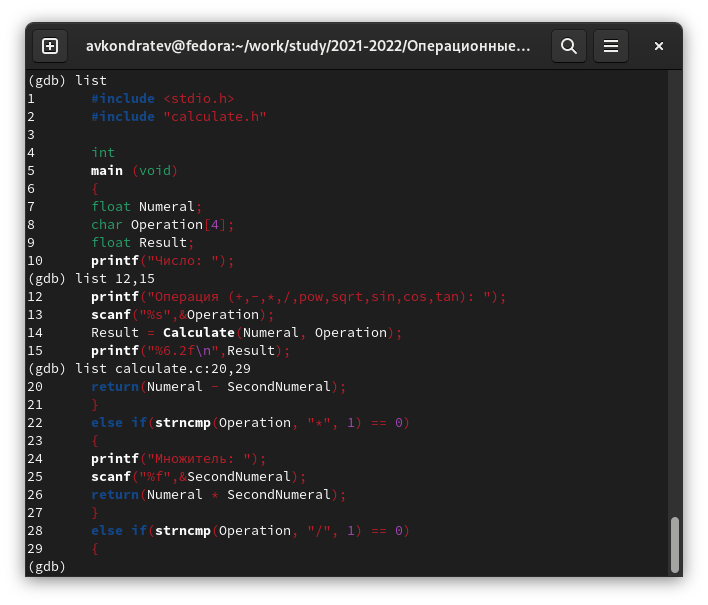


Figure 6: Рис. 6

1. Установил точку останова в файле calculate.c на строке номер 21 и вывел информацию о точках останова с помощью info breakpoints(рис.7)

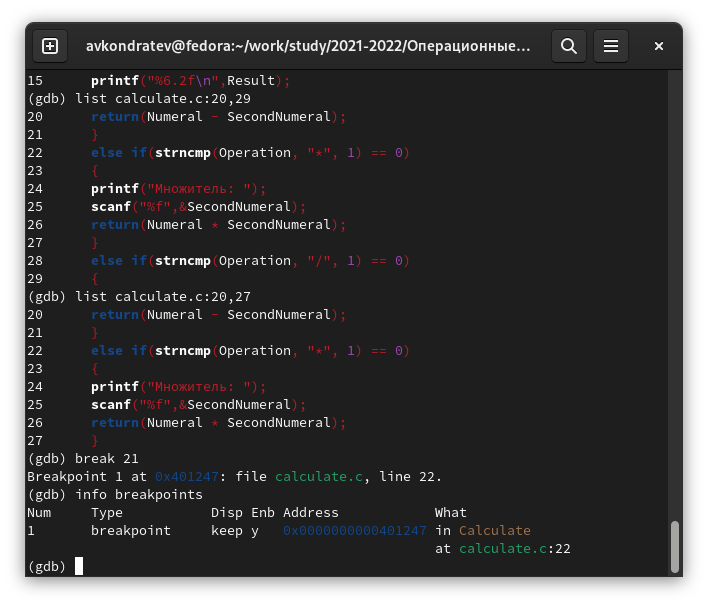


Figure 7: Рис. 7

1. Убедился, что программа останавливается в момент точки останова, с помощью print Numeral выяснил что эта переменная равна 5 на тот момент и сравнил результат с командой display Numeral(рис.8)

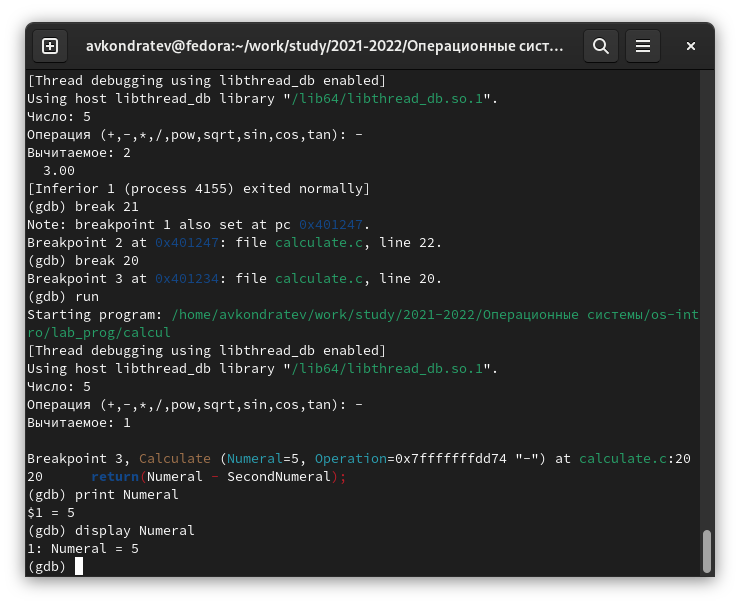


Figure 8: Рис. 8

1. Убрал точки останова(рис.9)

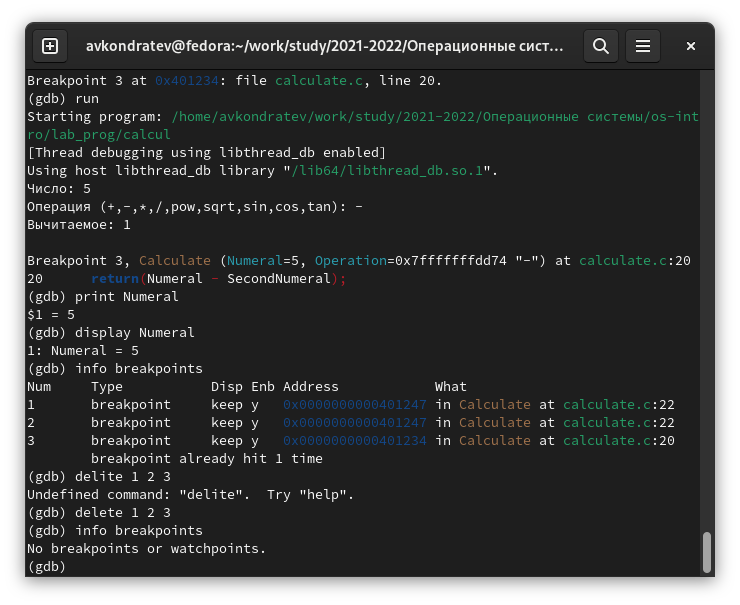


Figure 9: Рис. 9

1. С помощью утилиты splint проанализировал коды файлов calculate.c и main.c(рис.10)

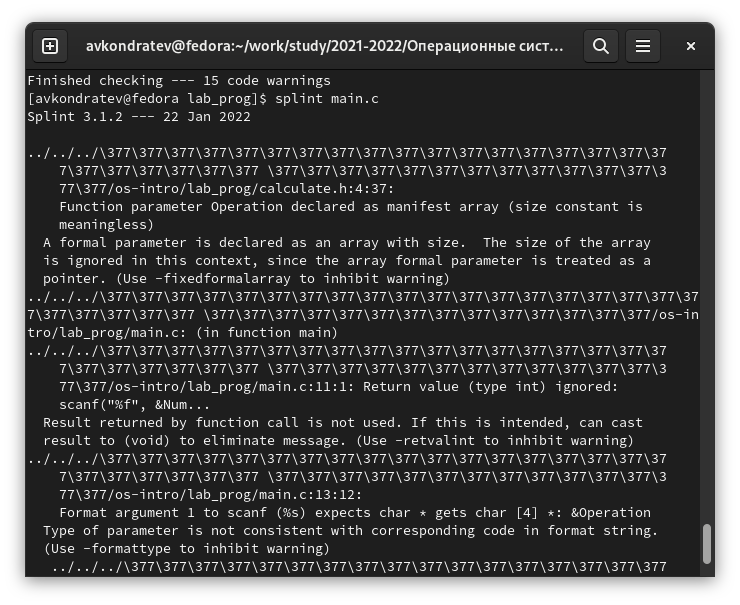


Figure 10: Рис. 10

# 4 Выводы

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

# 5 Контрольные вопросы

1. С помощью справочной команды man
2. – планирование, включающее сбор и анализ требований к функционалу и другим характеристикам разрабатываемого приложения;  
   – проектирование, включающее в себя разработку базовых алгоритмов и спецификаций, определение языка программирования;  
   – непосредственная разработка приложения:  
   – кодирование — по сути создание исходного текста программы (возможно в нескольких вариантах);  
   – анализ разработанного кода;  
   – сборка, компиляция и разработка исполняемого модуля;  
   – тестирование и отладка, сохранение произведённых изменений;  
   – документирование.
3. Суффикс - это расширение  
   gcc -c main.c  
   Таким образом, gcc по расширению (суффиксу) .c распознает тип файла для компиляции и формирует объектный модуль — файл с расширением .o
4. Для запуска программ, написанных на других языках программирования
5. Для исполнения мэйкфайлов
6. Он состоит из списка зависимостей и команд, которые нужно выполнить
7. 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
8. – backtrace – выводит весь путь к текущей точке останова, то есть названия всех функций, начиная от main(); иными словами, выводит весь стек функций;  
   – break – устанавливает точку останова; параметром может быть номер строки или название функции;  
   – clear – удаляет все точки останова на текущем уровне стека (то есть в текущей функции);  
   – continue – продолжает выполнение программы от текущей точки до конца;  
   – delete – удаляет точку останова или контрольное выражение;  
   – display – добавляет выражение в список выражений, значения кото- рых отображаются каждый раз при остановке программы;  
   – finish – выполняет программу до выхода из текущей функции; отоб- ражает возвращаемое значение,если такое имеется;  
   – info breakpoints – выводит список всех имеющихся точек останова;  
   – info watchpoints – выводит список всех имеющихся контрольных выражений;  
   – list – выводит исходный код; в качестве параметра передаются название файла исходного кода, затем, через двоеточие, номер начальной и конечной строки;  
   – next – пошаговое выполнение программы, но, в отличие от команды step, не выполняет пошагово вызываемые функции;  
   – print – выводит значение какого-либо выражения (выражение пере- даётся в качестве параметра);  
   – run – запускает программу на выполнение;  
   – set – устанавливает новое значение переменной  
   – step – пошаговое выполнение программы;  
   – watch – устанавливает контрольное выражение
9. Просмотрел код программы, поставил точку останова, запустил программу, на точке останова узнал значение переменной, закончил программу
10. Синтаксические ошибки отсутствовали
11. cscope − исследование функций, содержащихся в программе, lint − критическая проверка программ, написанных на языке Си
12. Splint анализирует программный код, проверяет корректность задания аргументов использованных в программе функций и типов возвращаемых значений.