Отчёт по лабораторной работе №14

дисциплина: Операционные системы

Латаева Гюзелия Андреевна

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:

gcc -c calculate.c

gcc -c main.c

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

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

– Запустите отладчик GDB, загрузив в него программу для отладки: gdb ./calcul

– Для запуска программы внутри отладчика введите команду run: run

– Для постраничного (по 9 строк) просмотра исходного код используйте команду list: list

– Для просмотра строк с 12 по 15 основного файла используйте list с параметрами: list 12,15

– Для просмотра определённых строк не основного файла используйте list с параметрами: list calculate.c:20,29

– Установите точку останова в файле calculate.c на строке номер 21:

list calculate.c:20,27

break 21

– Выведите информацию об имеющихся в проекте точка останова: info breakpoints

– Запустите программу внутри отладчика и убедитесь, что программа остановится в момент прохождения точки останова: run

5

backtrace

– Посмотрите, чему равно на этом этапе значение переменной Numeral, введя: print Numeral (на экран должно быть выведено число 5).

– Сравните с результатом вывода на экран после использования команды: display Numeral

– Уберите точки останова:

info breakpoints

delete 1

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

# 3 Выполнение лабораторной работы

1. В домашнем каталоге я создала подкаталог ~/work/os/lab\_prog и создала в нём файлы: calculate.h, calculate.c, main.c и заполнила их кодом из задания:(рис. ??), (рис. ??), (рис. ??)

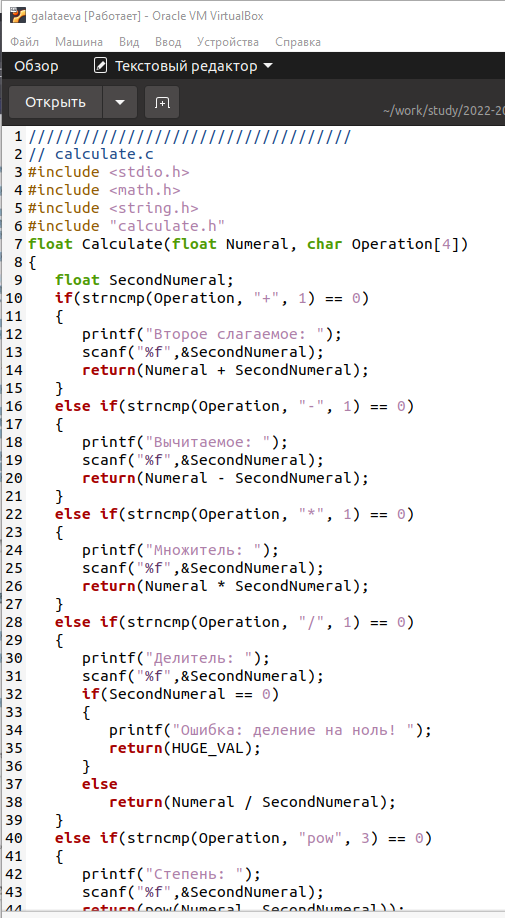


Рисунок 1

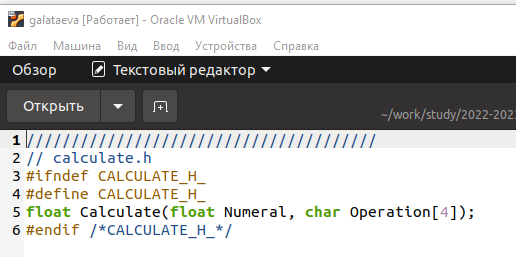


Рисунок 2

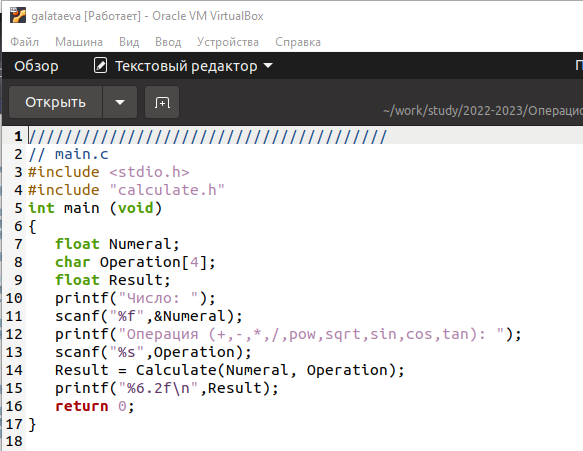


Рисунок 3

1. Выполнила компиляцию программы посредством gcc:

gcc -c main.c и gcc calculate.o main.o -o calcul -lm: (рис. ??)

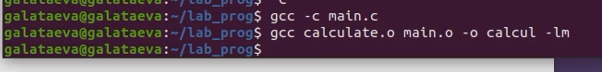


Рисунок 4

gcc -c calculate.c: (рис. ??)

Рисунок 5

Рисунок 5

1. Создала Makefile со следующим содержанием: (рис. ??)

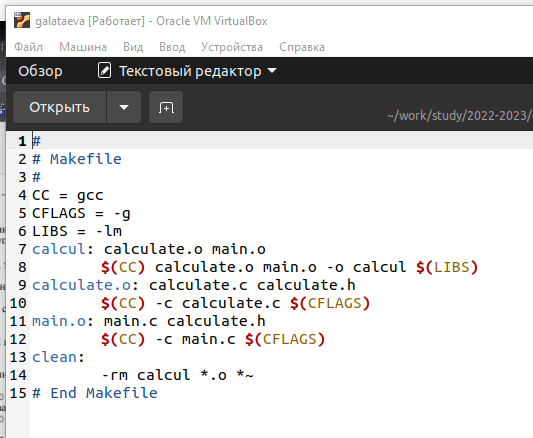


Рисунок 6

CC = gcc - создание переменной компилятора CC

CFLAGS = - создание переменной флагов компиляции CFLAGS

LIBS = -lm - создание переменной библиотек LIBS

calcul: calculate.o main.o. - описывается процесс создания исполняемого файла calcul, зависящего от файлов calculate.o и main.o

gcc calculate.o main.o -o calcul $(LIBS) - сборка файла calcul

calculate.o: calculate.c calculate.h - описывается процесс создания исполняемого файла calculate.o, зависящего от файлов calculate.c и calculate.h

gcc -c calculate.c $(CFLAGS): сборка файла calculate.o

main.o: main.c calculate.h: описывается процесс создания исполняемого файла main.o, зависящего от файлов main.c и calculate.h

gcc -c main.c $(CFLAGS): сборка файла main.o

clean: -rm calcul *.o* ~: очистка всех созданных файлов и объектов.

1. С помощью gdb выполнила отладку программы calcul:

– запустила отладчик GDB, загрузив в него программу для отладки: gdb ./calcul (рис. ??)

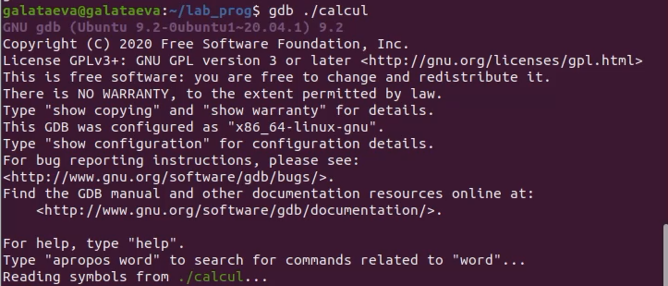


Рисунок 7

– для запуска программы внутри отладчика ввела команду run: (рис. ??)

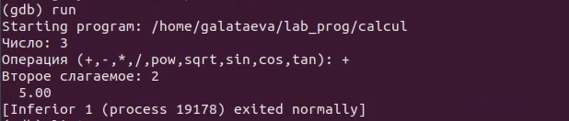


Рисунок 8

– использовала команду list для постраничного просмотра исходного кода: (рис. ??)

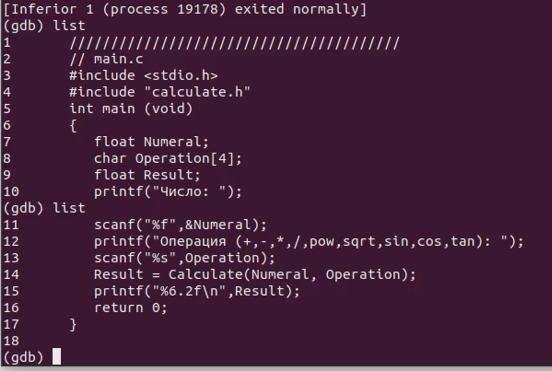


Рисунок 9

– для просмотра строк с 12 по 15 основного файла использовала list 12,15: (рис. ??)

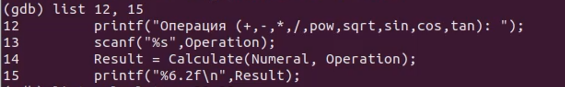


Рисунок 10

– для просмотра определённых строк не основного файла использовала list calculate.c:20,29: (рис. ??)

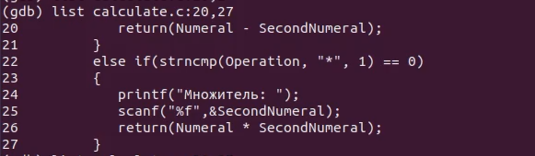


Рисунок 11

– установила точку останова в файле calculate.c на строке номер 21: (рис. ??)

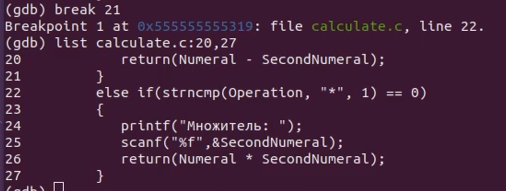


Рисунок 12

– вывела информацию об имеющихся в проекте точек останова: (рис. ??)

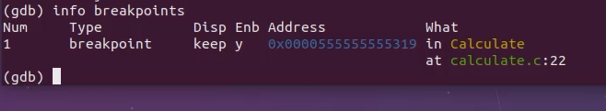


Рисунок 13

– запустила программу внутри отладчика. Программа остановилась в момент прохождения точки останова: (рис. ??)

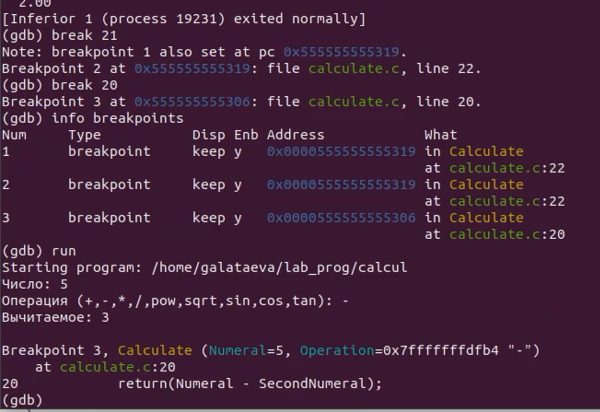


Рисунок 14

– значение переменной Numeral: (рис. ??)

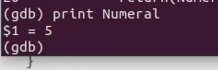


Рисунок 15

– сравнила с результатом вывода на экран после использования команды: (рис. ??)

Рисунок 16

Рисунок 16

– убрала точки останова: (рис. ??)

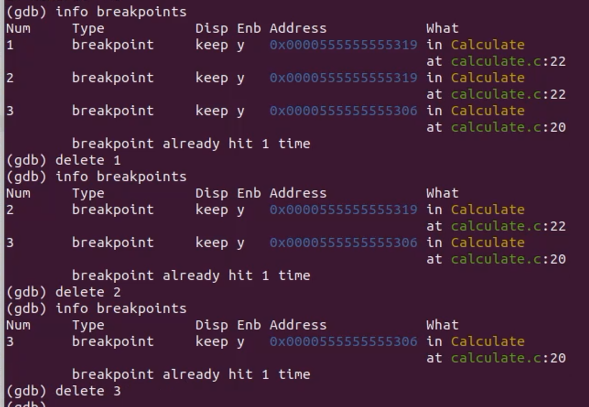


Рисунок 17

1. Использовала утилиту splint чтобы проанализировать коды файлов calculate.c и main.c: (рис. ??) (рис. ??)

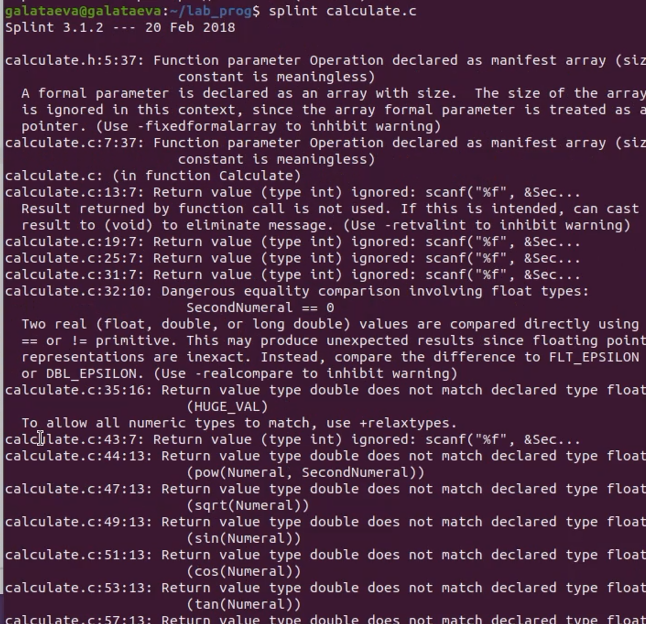


Рисунок 18

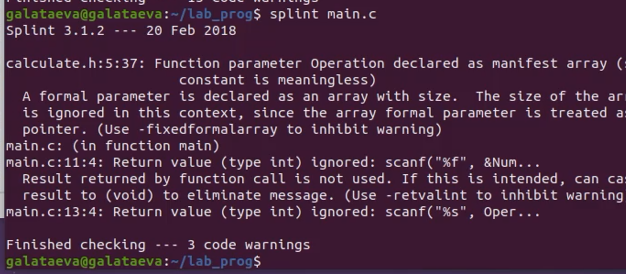


Рисунок 19

# 4 Выводы

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

Использовать команды man и info в терминале Linux.

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

1-Написать код на языке программирования

2-Компиляция исходного кода в исполняемый файл

3-Отладка программы

4-Создание Makefile и использование утилиты make

5-Доработка, тестирование

6- Документирование

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

Это часть имени файла, которая указывает на его тип или назначение. Например, programm.c суффикс “.c” указывает, что это файл кода на языке программирования C.

1. Каково основное назначение компилятора языка С в UNIX?

Преобразование исходного кода программы, написанной на языке C, в машинный код, который может быть исполнен процессором компьютера.

1. Для чего предназначена утилита make?

Для автоматизации процесса сборки приложения в UNIX.

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

См. пункт 3 выполнения отчета.

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

Возможность управлять выполнением программы, остановка ее на определенном месте, просмотр значений переменных и выполнения команд в контексте отладки.

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

backtrace - вывод на экран пути к текущей точке останова

break - установить точку останова

clear - удалить все точки останова в функции continue продолжить выполнение программы

delete - удалить точку останова

display - добавить выражение в список выражений, значения которых отображаются при достижении точки останова программы

finish - выполнить программу до момента выхода из функции

info breakpoints - вывести на экран список используемых точек останова

info watchpoints - вывести на экран список используемых контрольных выражений

list - вывести на экран исходный код (в качестве параметра может быть указано название файла и через двоеточие номера начальной и конечной строк)

next - выполнить программу пошагово, но без выполнения вызываемых в программе функций

print - вывести значение указываемого в качестве параметра выражения

run - запуск программы на выполнение

set - установить новое значение переменной

step - пошаговое выполнение программы watch установить контрольное выражение, при изменении значения которого программа будет остановлена

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

1-Компиляция программы с опцией -c

2-создание Makefile

3-запуск отладчика gdb, указав имя скомпилированного исполняемого файла

4-запуск run

5-посмотр исходного кода

6-установка точки останова на нужных строках кода с помощью команды break

7-запуск программы

8-исправление кода программы и повтор отладки после нахождения ошибки

1. Прокомментируйте реакцию компилятора на синтаксические ошибки в программе при его первом запуске.
2. Назовите основные средства, повышающие понимание исходного кода программы.

Комментарии, документация, отладчики, редакторы кода, программы анализа кода, руководства.

1. Каковы основные задачи, решаемые программой splint?

Это инструмент статического анализа кода на языке C, который помогает обнаруживать ошибки в коде, связанные с безопасностью и потенциальными уязвимостями.

Основные задачи: поиск ошибок в коде, анализ потока данных, проверка безопасности, проверка соответствия стандартам, улучшение качества кода.

# Список литературы

1. ya.ru