**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**Кафедра прикладной информатики и теории вероятностей**

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 14**

*дисциплина: Операционные системы*

Студент: Мухтарова Камила Айратовна.

Группа: НПИбд-02-20

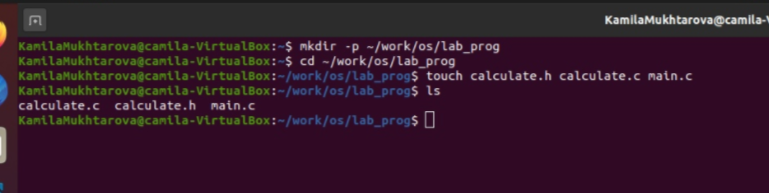
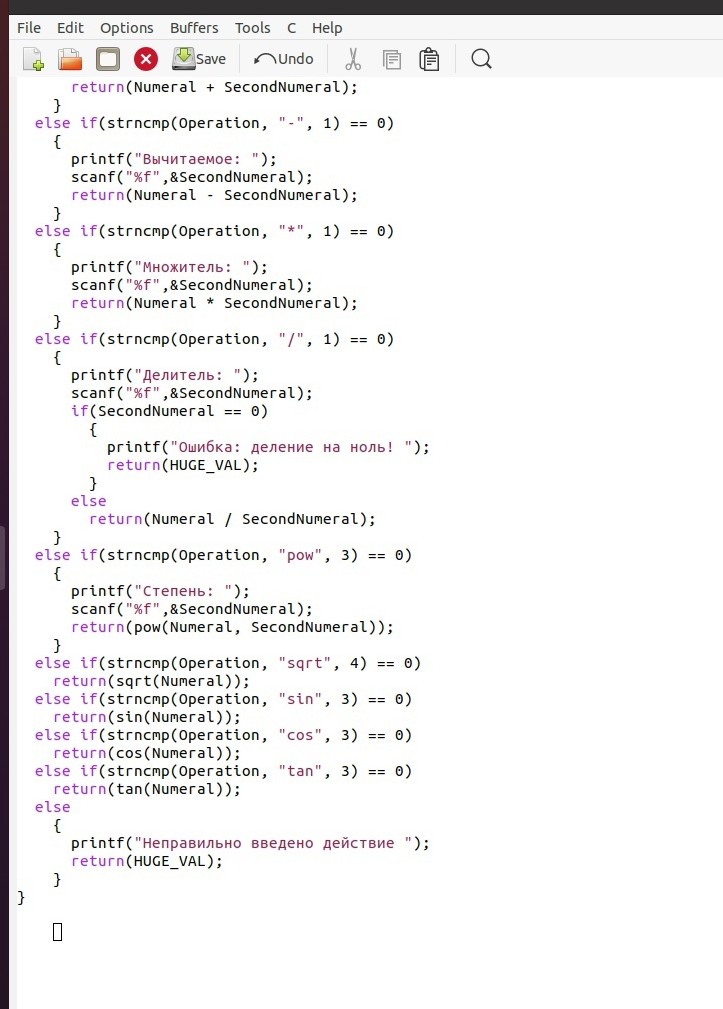
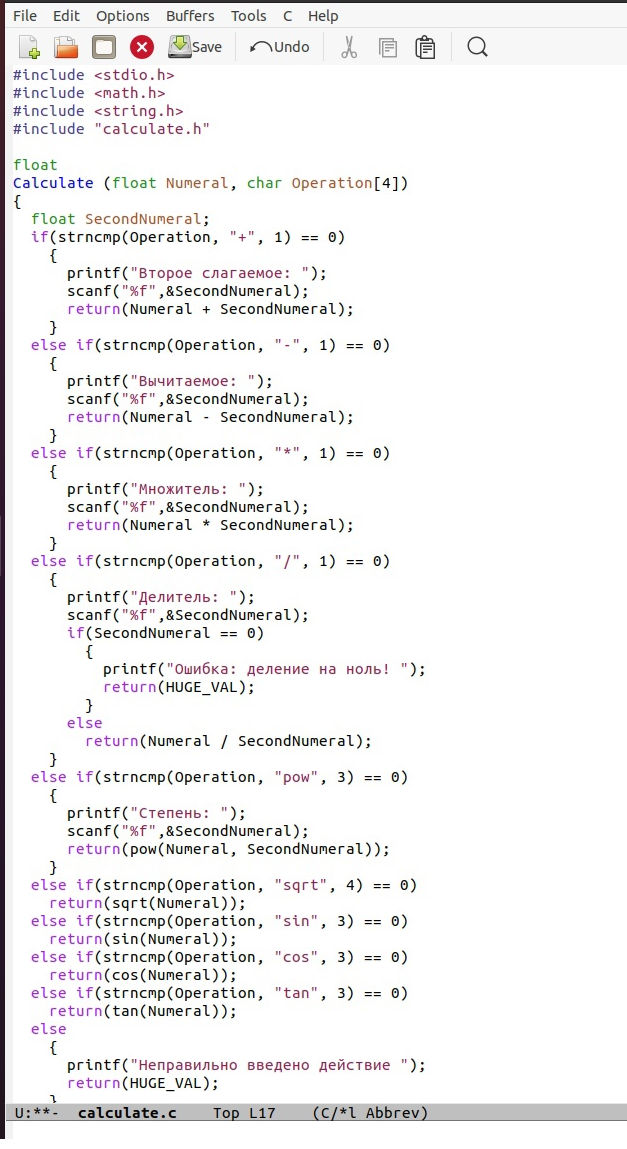
**МОСКВА**

2021 год

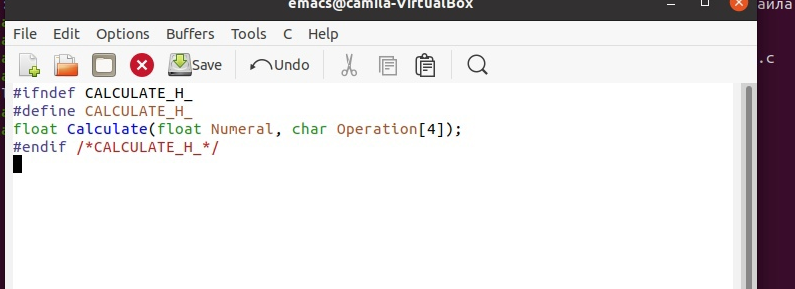
Цель работы: Приобрести простейшие навыки разработки, анализа, тестирования и отладки приложений в ОС типа UNIX/Linux на примере создания на языке программирования С калькулятора с простейшими функциями.

Ход работы:

* В домашнем каталоге создадим подкаталог ~/work/os/lab\_prog.. Создадим 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.

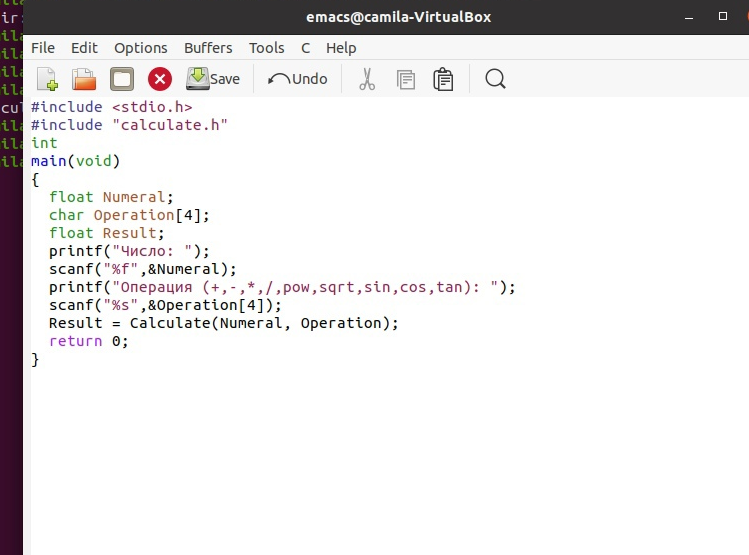
  
  
(рис. 1) - создание  
  
  
Реализация функций калькулятора в файле calculate.с  
  
  
(рис. 2, 3) - скрипт

Интерфейсный файл calculate.h, описывающий формат вызова функции калькулятора



(рис.4) - скрипт

Основной файл main.c, реализующий интерфейс пользователя к калькулятору

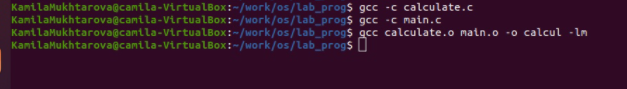
  
  
(рис.5) - скрипт

* Выполним компиляцию программы посредством gcc:

gcc -c calculate.c

gcc -c main.c

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



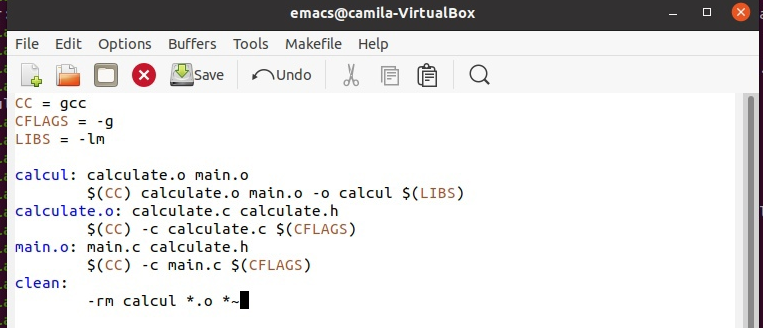
(рис. 6) - компиляция

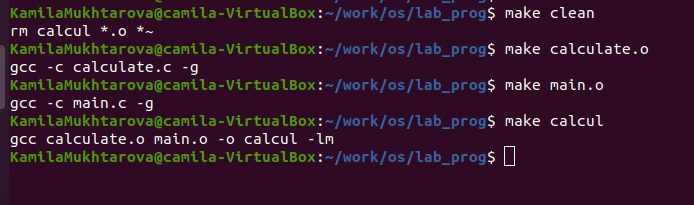
В ходе компиляции программы никаких ошибок выявлено не было.

* Создадим Makefile со следующим содержанием:

# # Makefile # CC = gcc CFLAGS = LIBS = -lm calcul: calculate.o main.o gcc calculate.o main.o -o calcul $(LIBS) calculate.o: calculate.c calculate.h gcc -c calculate.c $(CFLAGS) main.o: main.c calculate.h gcc -c main.c $(CFLAGS) clean: -rm calcul \*.o \*~ # End Makefile

Данный файл необходим для автоматической компиляции файлов calculate.c (цель calculate.o), main.c (цель main.o), а также их объединения в один исполняемый файл calcul (цель calcul). Цель clean нужна для автоматического удаления файлов. Переменная CC отвечает за утилиту для компиляции. Переменная CFLAGS отвечает за опции в данной утилите. Переменная LIBS отвечает за опции для объединения объектных файлов в один исполняемый файл.  
  
В переменную CFLAGS добавила опцию -g, необходимую для компиляции объектных файлов и их использования в программе отладчика GDB. Сделала так, что утилита компиляции выбирается с помощью переменной CC. После этого я удалила исполняемые и объектные файлы из каталога с помощью команды «make clean» . Выполнила компиляцию файлов, используя команды «make calculate.o», «make main.o», «male calcul»

  
  
(рис. 7) - Makefile

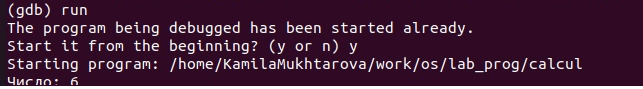


(рис. 7) - компиляция

* С помощью gdb выполним отладку программы calcul

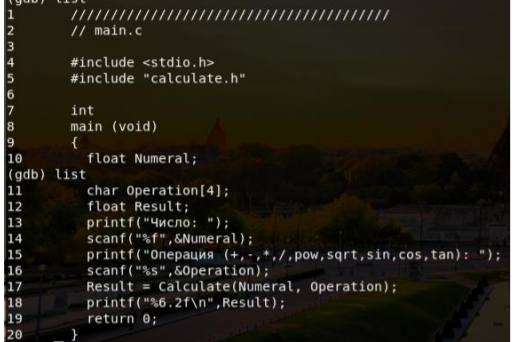
– Запустим отладчик GDB, загрузив в него программу для отладки: gdb ./calcul

Для запуска программы внутри отладчика ввела команду «run»



(рис.8) - запуск программы

Для постраничного (по 10 строк) просмотра исходного кода использовала команду «list»



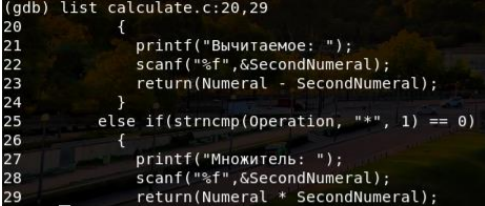
(рис. 9) - просмотр по 10 строк

Для просмотра строк с 12 по 15 основного файла использовала команду «list 12,15»



(рис. 10) - просмотр с 12 по 15 строк

Для просмотра определенных строк не основного файла использовала команду «list calculate.c:20,29»



(рис. 11) - определённые строки

Установила точку останова в файле calculate.c на строке номер 21, используя команды «list calculate.c:20,27» и «break 21»



(рис. 12) - точка останова

Вывела информацию об имеющихся в проекте точках останова с помощью команды «info breakpoints»



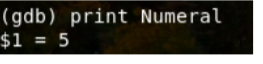
(рис.13) - информация о точке останова

Запустила программу внутри отладчика и убедилась, что программа остановилась в момент прохождения точки останова. Использовала команды «run», «5», «−» и «backtrace»



(рис. 14) - запуск программы

Посмотрела, чему равно на этом этапе значение переменной Numeral, введя команду «print Numeral»



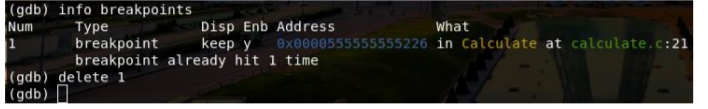
(рис. 15) - просмотр значения

Сравнила с результатом вывода на экран после использования команды «display Numeral». Значения совпадают



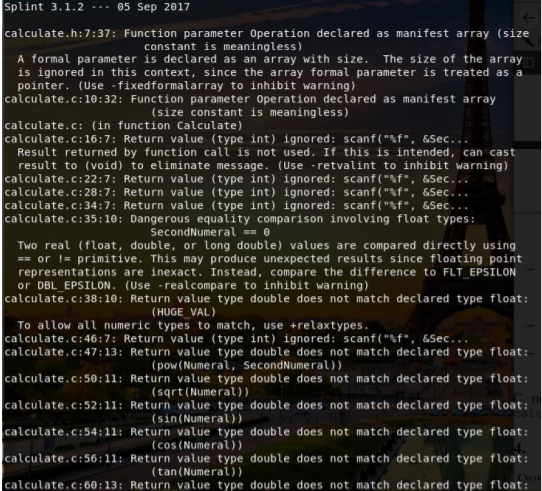
(рис. 16) - сравнение

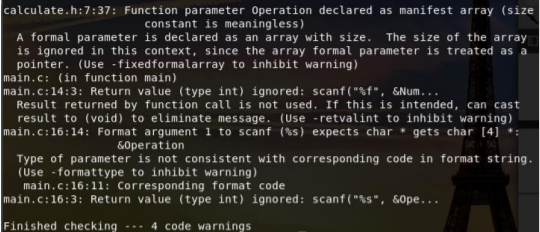
Убрала точки останова с помощью команд «info breakpoints» и «delete 1»



(рис. 17) - удаление точки останова

Далее воспользовалась командами «splint calculate.c» и «splint main.c». C помощью утилиты splint выяснилось, что в файлах calculate.c и main.c присутствует функция чтения scanf, возвращающая целое число (тип int), но эти числа не используются и нигде не сохранятся. Утилита вывела предупреждение о том, что в файле calculate.c происходит сравнение вещественного числа с нулем. Также возвращаемые значения (тип double) в функциях pow, sqrt, sin, cos и tan записываются в переменную типа float, что свидетельствует о потери данных.





(рис. 18, 19) - утилита splint

Вывод: В ходе выполнения данной лабораторной работы я приобрела простейшие навыки разработки, анализа, тестирования и отладки приложений в ОС типа UNIX/Linux на примере создания на языке программирования С калькулятора с простейшими функциями