**Лабораторная работа 14**

по дисциплине «Операционные системы»

Выполнил:

Студент группы НПВбм-01-19

Студенческий билет №: 1032197651

Потапов Александр Сергеевич

Руководитель:

Валиева Татьяна Рефатовна

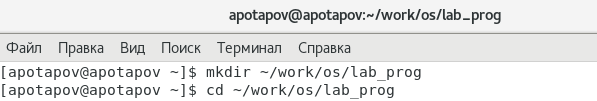
Москва 2023

# Цель работы

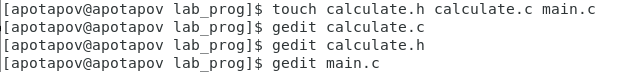
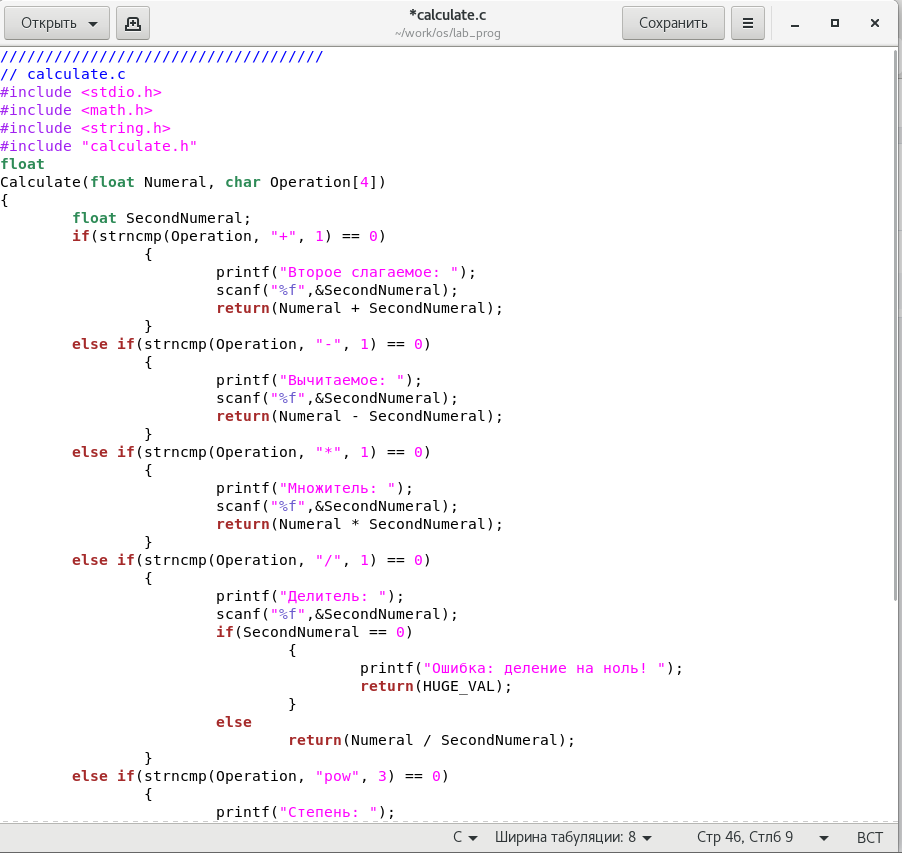
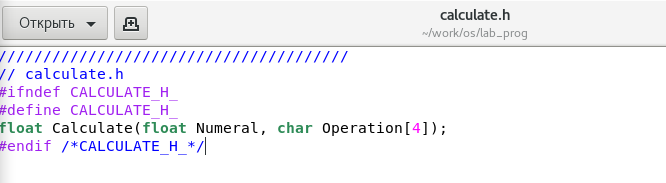
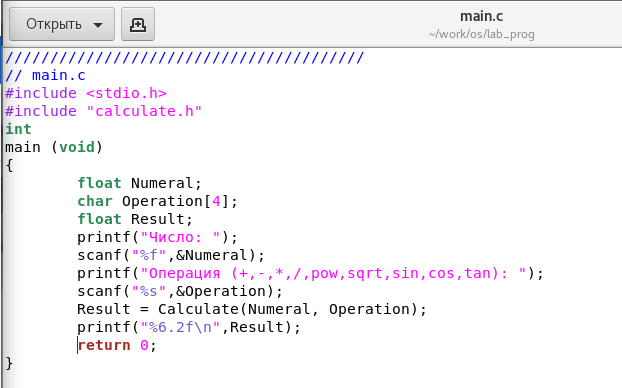
Приобрести простейшие навыки разработки, анализа, тестирования и отладки приложений в ОС типа UNIX/Linux на примере создания на языке программирования С калькулятора с простейшими функциями.

# Начало работы

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

* 
* Рисунок 1

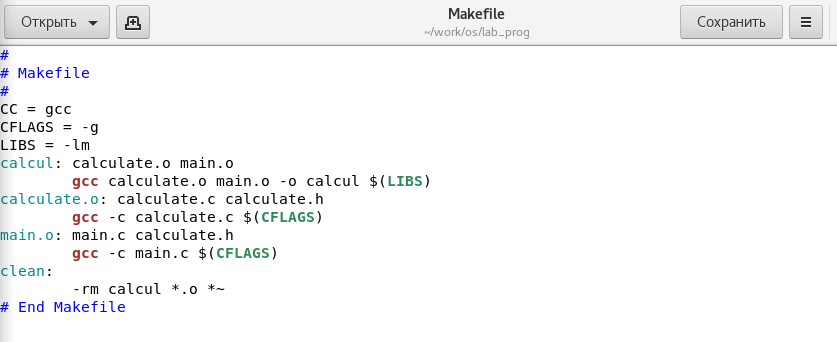
1. Создадим в нём файлы: calculate.h, calculate.c, main.c.

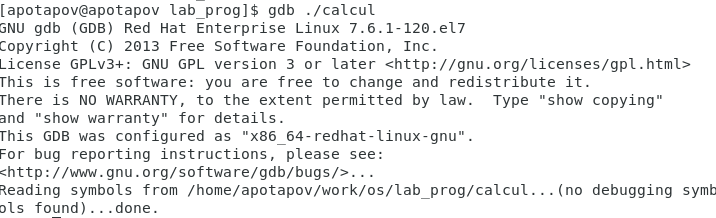
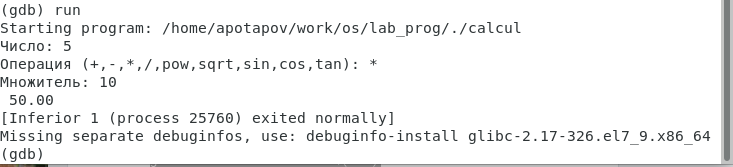
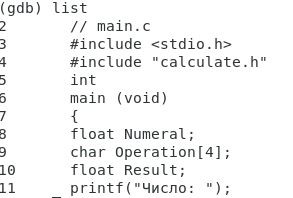
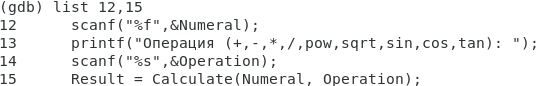
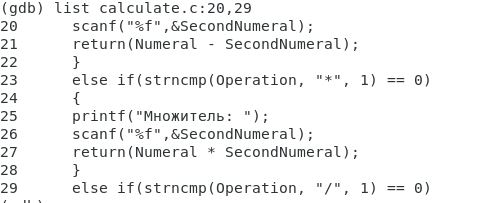
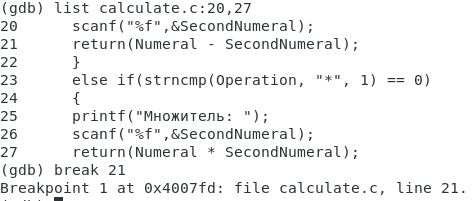
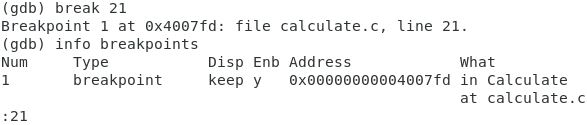
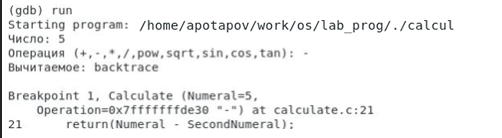
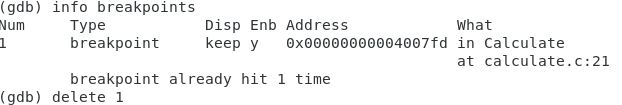
* 
* Рисунок 2
* Реализация функций калькулятора в файле calculate.с:
* 
* Рисунок 3
* Интерфейсный файл calculate.h, описывающий формат вызова функциикалькулятора:
* 
* Рисунок 4
* Основной файл main.c, реализующий интерфейс пользователя к калькулятору:
* 
* Рисунок 5

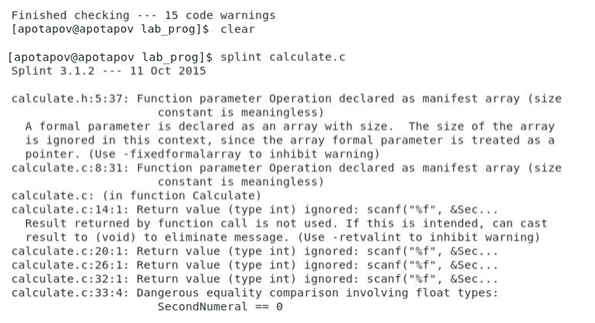
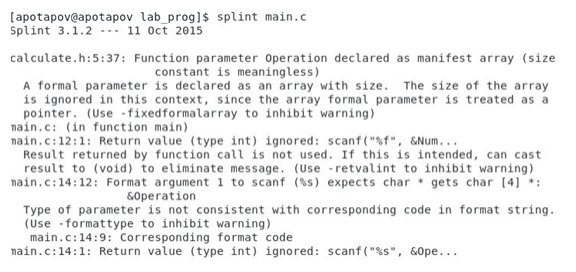
1. Выполним компиляцию программы посредством gcc.

* 
* Рисунок 6

1. При необходимости исправим синтаксические ошибки.
2. Создадим Makefile со следующим содержанием:

* 
* Рисунок 7

1. С помощью gdb выполните отладку программы calcul (перед использованием gdb исправьте Makefile):
   * Запустите отладчик GDB, загрузив в него программу для отладки:
   * gdb ./calcul
   * 
   * Рисунок 8
   * Для запуска программы внутри отладчика введите команду run:
   * run
   * 
   * Рисунок 9
   * Для постраничного (по 9 строк) просмотра исходного код используйте команду list:
   * list
   * 
   * Рисунок 10
   * Для просмотра строк с 12 по 15 основного файла используйте list с параметрами:
   * list 12,15
   * 
   * Рисунок 11
   * Для просмотра определённых строк не основного файла используйте list с параметрами:
   * list calculate.c:20,29
   * 
   * Рисунок 12
   * Установите точку останова в файле calculate.c на строке номер 21:
   * list calculate.c:20,27
   * break 21
   * 
   * Рисунок 13
   * Выведите информацию об имеющихся в проекте точка останова:
   * info breakpoints
   * 
   * Рисунок 14
   * Запустите программу внутри отладчика и убедитесь, что программа остановится в момент прохождения точки останова:
   * run
   * 5
   * -
   * backtrace
   * 
   * Рисунок 15
   * Отладчик выдаст следующую информацию:
   * #0 Calculate (Numeral=5, Operation=0x7fffffffd280 "-")
   * at calculate.c:21
   * #1 0x0000000000400b2b in main () at main.c:17
   * а команда backtrace покажет весь стек вызываемых функций от начала программы до текущего места.
   * Посмотрите, чему равно на этом этапе значение переменной Numeral, введя:
   * print Numeral
   * fig:
   * Рисунок 16
   * На экран должно быть выведено число 5.
   * Сравните с результатом вывода на экран после использования команды:
   * display Numeral
   * fig:
   * Рисунок 17
   * Уберите точки останова:
   * info breakpoints
   * delete 1
   * 
   * Рисунок 18
2. С помощью утилиты splint попробуем проанализировать коды файлов calculate.c и main.c.

* 
* Рисунок 19
* 
* Рисунок 20

# Вывод

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.

# Контрольные вопросы

1. Для этого есть команда man и предлагающиеся к ней файлы.
2. Кодировка, Компиляция, Тест.
3. Это расширения файлов.
4. Программа gcc, которая интерпретирует к определенному языку программирования аргументы командной строки и определяет запуск нужного компилятора для нужного файла
5. Для компиляции группы файлов. Собрания из них программы, и последующего удаления.
6. program: main.o lib.o

* cc -o program main.o lib.o
* main.o lib.o: defines.h
* В имени второй цели указаны два файла и для этой же цели не указана команда компиляции. Кроме того, нигде явно не указана зависимость объектных 14 файлов от «\*.c»-файлов. Дело в том, что программа make имеет предопределённые правила для получения файлов с определёнными расширениями. Так, для цели-объектного файла (расширение «.o») при обнаружении соответствующего файла с расширением «.c» будет вызван компилятор «сс -с» с указанием в параметрах этого «.c»-файла и всех файлов-зависимостей.

1. Программы для отладки нужны для нахождения ошибок в программе. Для их использования надо скомпилировать программу таким образом, чтобы отладочная информация содержалась в конечном бинарном файле.
   * 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 которых отображаются каждый раз при остановке программы;
   * finish – выполняет программу до выхода из текущей функции; отображает 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 передаётся в качестве параметра);
   * run – запускает программу на выполнение;
   * set – устанавливает новое значение переменной step – пошаговое выполнение программы;
   * watch – устанавливает контрольное выражение, программа остановится, как только значение контрольного выражения изменится;
   * gdb –silent ./calcul
   * run 12. list
   * backtrace
   * breakpoints
   * print Numeral
   * Splint
2. Консоль выводит ошибку с номером строки и ошибочным сегментом, но при этом есть возможность выполнить программу сразу.
   * Правильный синтаксис
   * Наличие комментариев
   * Разбиение большой сложной программы на несколько сегментов попроще.
3. Split – разбиение файла на меньшие, определённого размера. Может разбивать текстовые файлы по строкам и любые – по байтам. По умолчанию читает со стандартного ввода и создает файлы с именами вида xaa, xab и т.д. По умолчанию разбиение идёт по 1000 строк в файле.