Отчёт по лабораторной работе №13

Операционные системы

Хрусталев Влад Николаевич

Содержание

Список иллюстраций

Список таблиц

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 Выполнение лабораторной работы

В домашнем каталоге создадим подкаталог ~/work/os/lab_prog. После чего создадим в нём файлы: calculate.h, calculate.c, main.c и выполним проверку. Перейдём в emacs (Рис. [2.1]).

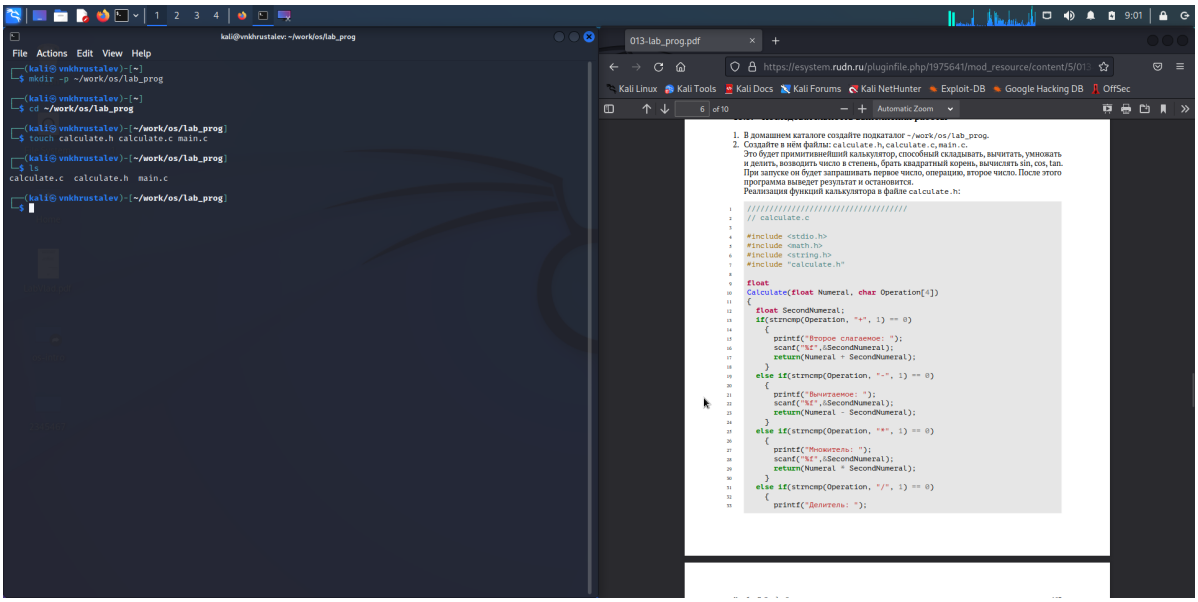

Рис. 2.1: Создание нового подкаталога и файлов в нём

В mousepad откроем созданный файл calculate.c и приступим к переносу в него скрипта из файла (Рис. [2.2]).

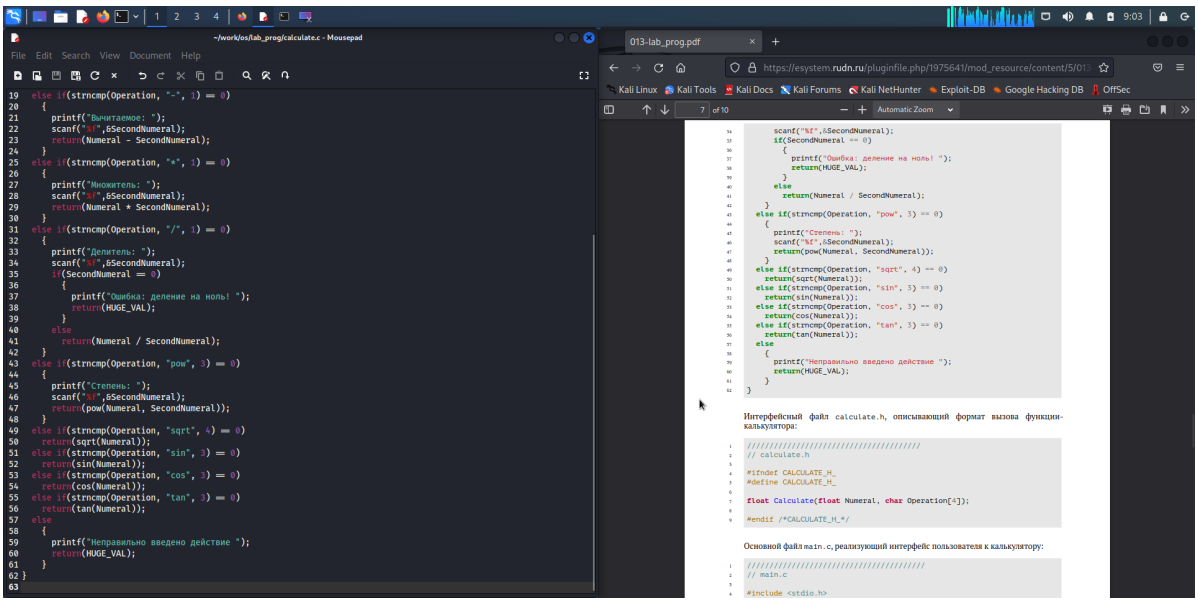

Рис. 2.2: Перенос скрипта для calculate.c

После того как мы перенесли и сохранили скрипт для первого файла, открываем файл calculate.h и также переносим в него скрипт, но уже для второго файла. Выполняем сохранение (Рис. [2.3]).

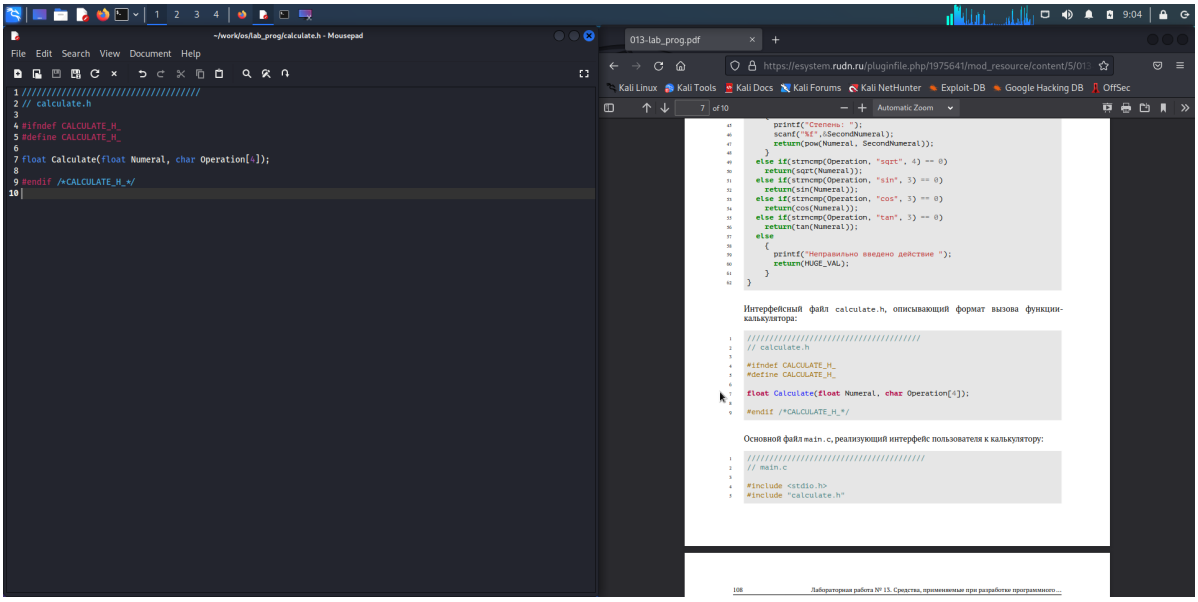

Рис. 2.3: Перенос скрипта для calculate.h

Теперь нам нужно перенести последний третий скрипт в файл main.c. После чего также выполняем сохранение и закрываем emacs (Рис. [2.4]).

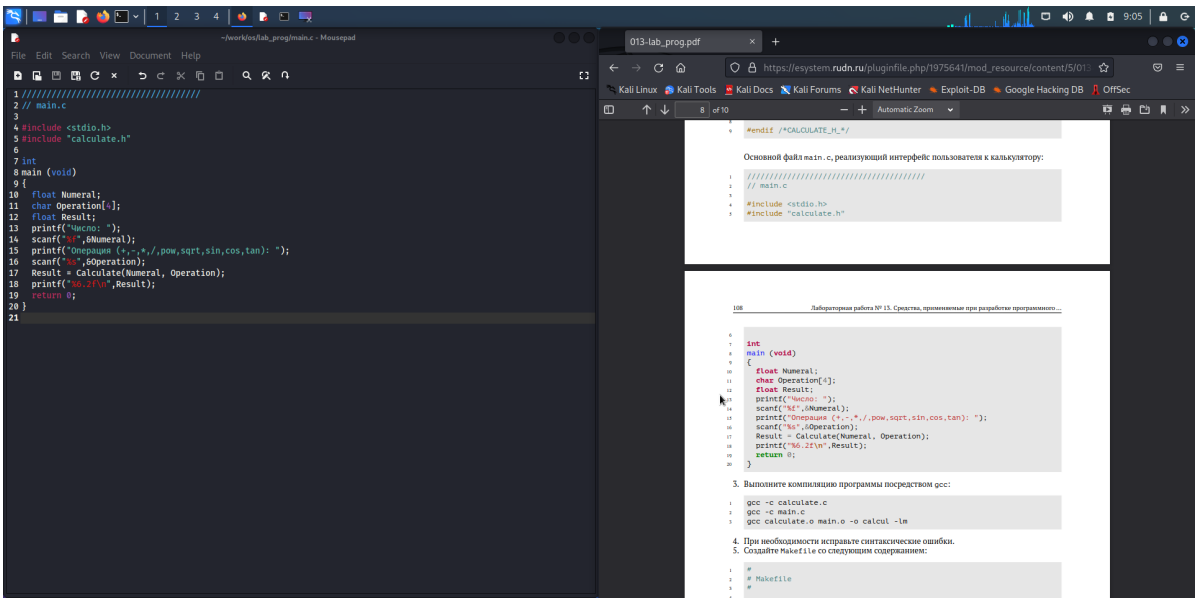

Рис. 2.4: Перенос скрипта для main.c

В терминале выполним компиляцию программы посредством gcc (Рис. [2.5]).

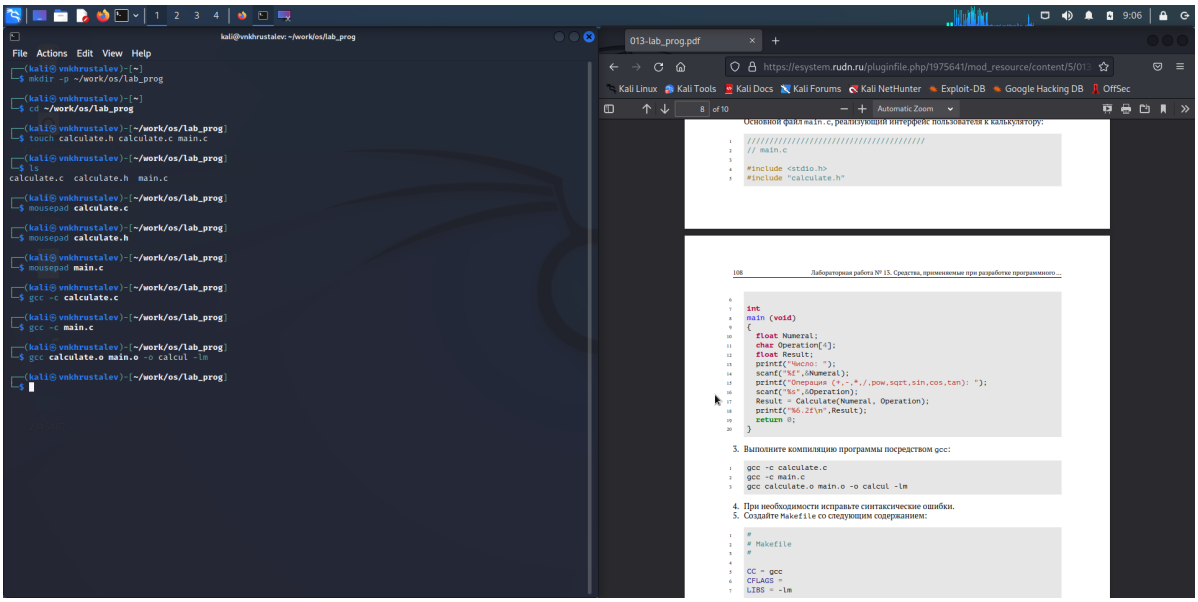

Рис. 2.5: Компиляция программы

Создадим Makefile и внесём туда небольшие изменения. В переменную CFLAGS добавил опцию -g, необходимую для компиляции объектных файлов и их использования в программе отладчика GDB. Сделал так, что утилита компиляции выбирается с помощью переменной CC (Рис. [2.6]).

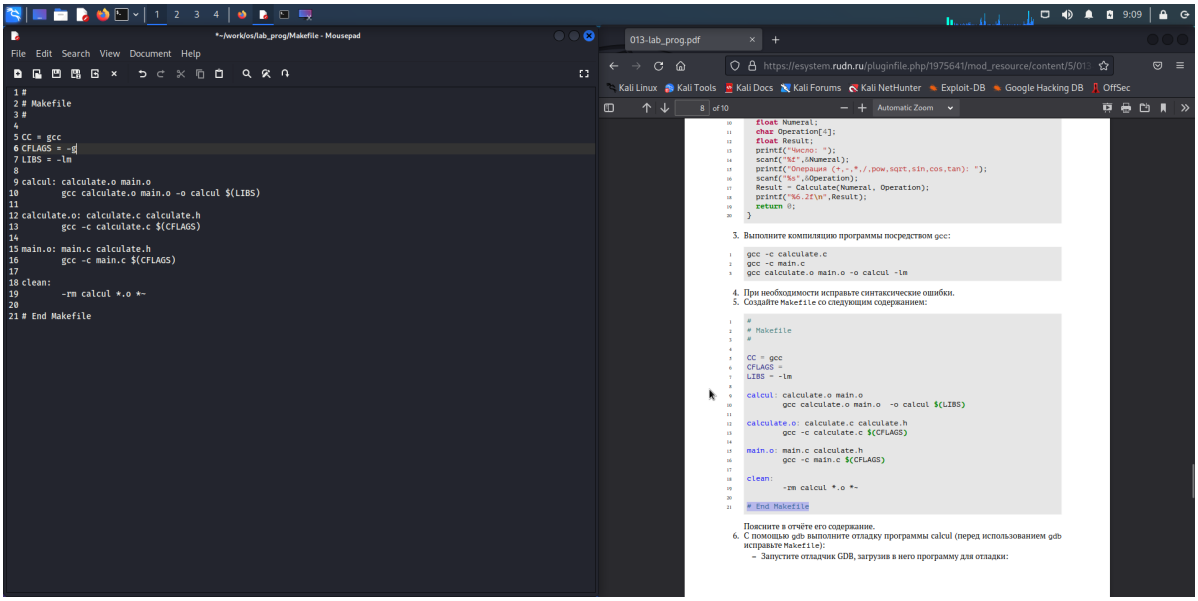

Рис. 2.6: Создание и изменение Makefile

С помощью gdb выполним отладку программы calcul. После чего запустим программу командой run (Рис. [2.7]).

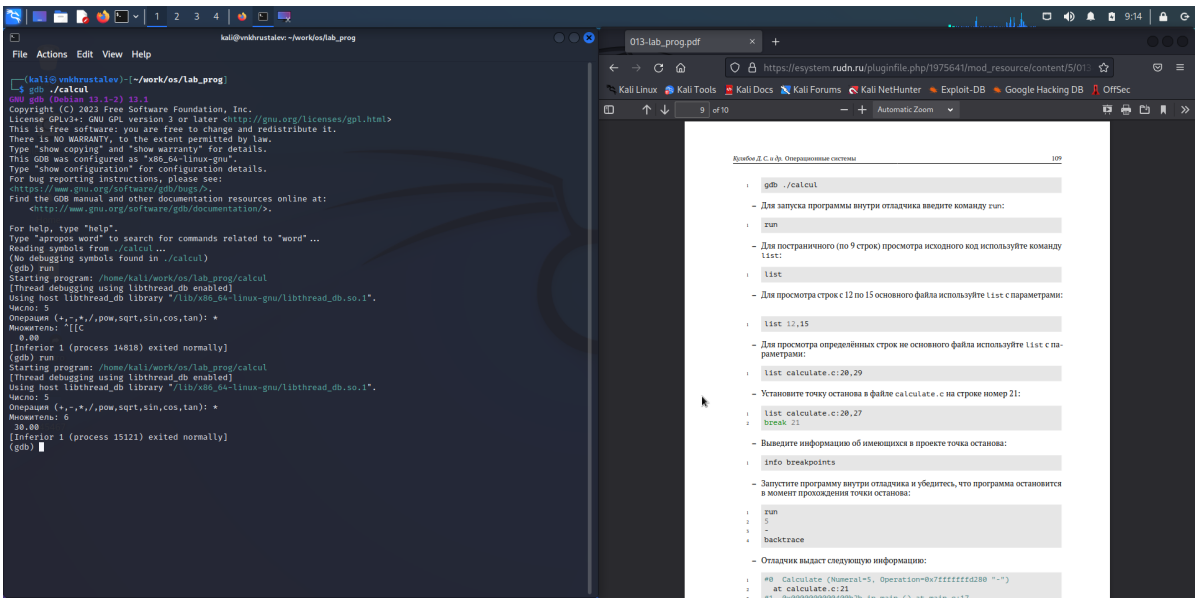

Рис. 2.7: Отладка программы calcul

Воспользовавшись утилитой splint проанализируем коды файлов calculate.c и main.c. С 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. [2.8]) и (Рис. [2.8]).

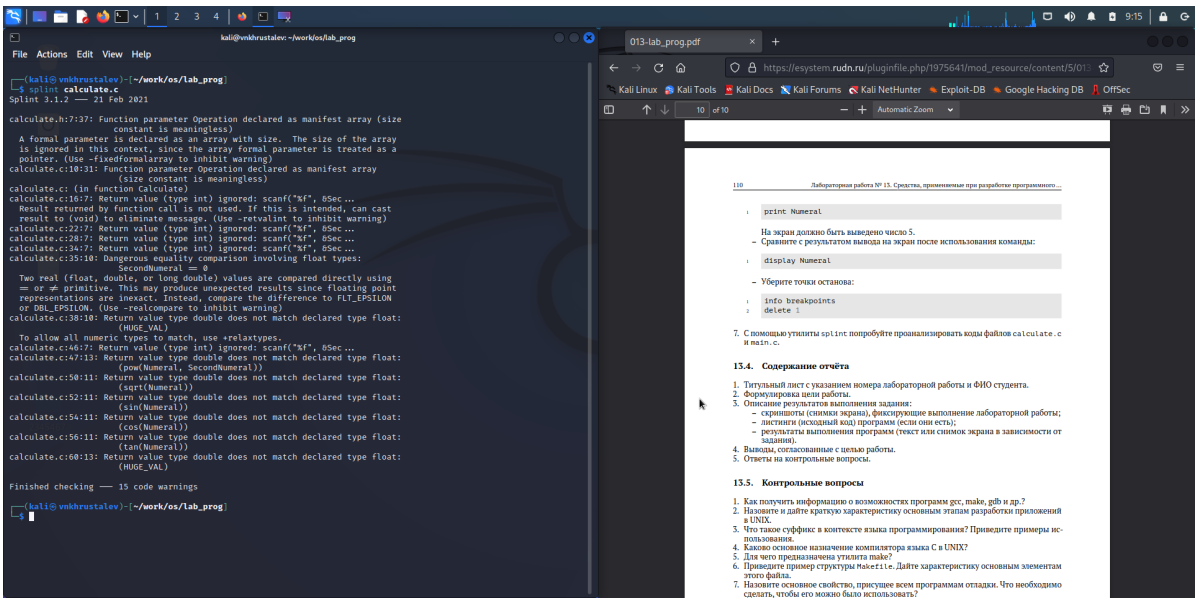

Рис. 2.8: Анализ файла calculate.c

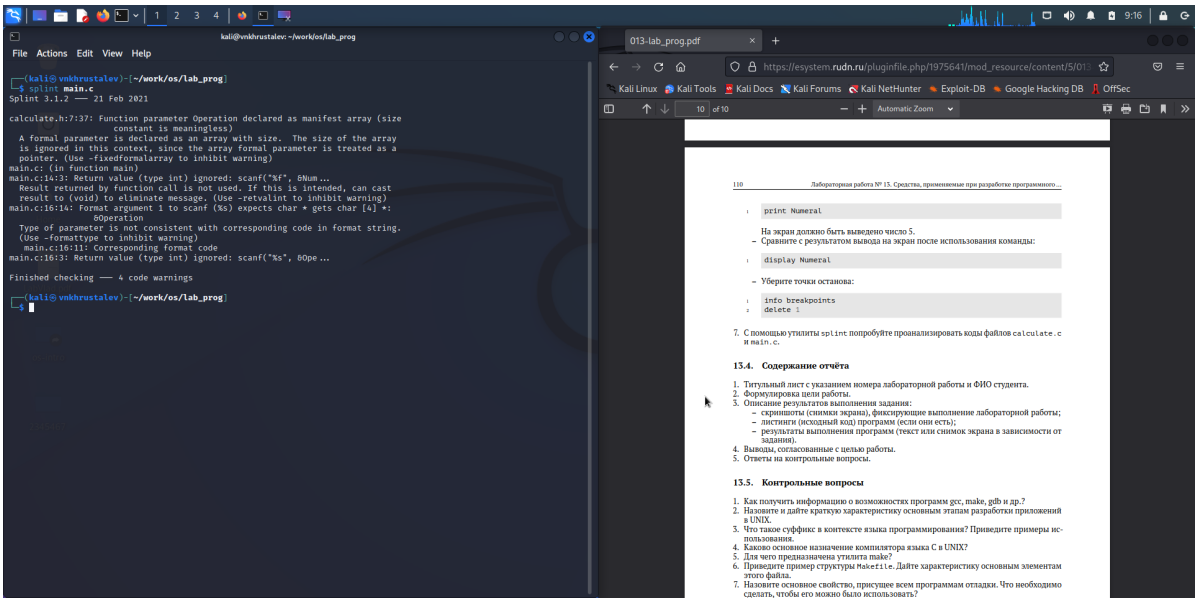

Рис. 2.9: Анализ файла main.c

3 Выводы

В ходе выполнения лабораторной работы мы приобрели простейшие навыки разработки, анализа, тестирования и отладки приложений в ОС типа UNIX/Linux на примере создания на языке программирования C калькулятора с простейшими функциями.

4 Контрольные вопросы

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

Чтобы получить информацию о возможностях программ gcc, make, gdb и др.нужно воспользоваться командой manили опцией -help(-h)для каждой команды.

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

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: окодирование по сути создание исходного текста программы (возможно в нескольких вариантах); анализ разработанного кода; осборка, компиляция и разработка исполняемого модуля; отестирование и отладка, сохранение произведённых изменений;
- документирование. Для создания исходного текста программы разработчик может воспользоваться любым удобным для него редактором текста: vi, vim, mceditor, emacs, geanyи др. После завершения написания исходного кода программы (возможно состоящей из нескольких файлов), необходимо её скомпилировать и получить исполняемый модуль.

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

Для имени входного файла суффикс определяет какая компиляция требуется. Суффиксы указывают на тип объекта. Файлы с расширением (суффиксом) .своспринимаются gcскак программы на языке C, файлы с расширением .ссили .C как файлы на языке C++, а файлы срасширением .осчитаются объектными.Например, в команде «gcc-cmain.c»:gccпо расширению (суффиксу) .сраспознает тип файла для компиляции и формирует объектный модуль файл с расширением .o. Если требуется получить исполняемый файл с определённым именем (например, hello), то требуется воспользоваться опцией oи в качестве параметра задать имя создаваемого файла: «gcc-ohellomaiВ ходе выполнения данной лабораторной работы я приобрелапростейшие навыки разработки, анализа, тестирования и отладки приложений в ОС типа UNIX/Linuxна примере создания на языке программирования C калькулятора с простейшими функциями.n.c».

4. Каково основное назначение компилятора языка C в UNIX?

Основное назначение компилятора языка Си в UNIX заключается в компиляции всей программы и получении исполняемого файла/модуля.

5. Для чего предназначена утилита make?

Для сборки разрабатываемого приложения и собственно компиляции полезно воспользоваться утилитой make. Она позволяет автоматизировать процесс преобразования файлов программы из одной формы в другую, отслеживает взаимосвязи между файлами.

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

Для работы с утилитой makeнеобходимо в корне рабочего каталога с Вашим проектом создать файл с названием makefileили Makefile, в котором будут описаны правила обработки файлов Вашего программного комплекса. В самом простом случае Makefile имеет следующий синтаксис: … : …<команда 1>…Сначала задаётся список целей, разделённых пробелами, за которым идёт двоеточие и список зависимостей. Затем в следующих строках указываются команды. Строки с командами обязательно должны начинаться с табуляции. В качестве цели в Makefileможет выступать имя файла или название какого-то действия. Зависимость задаёт исходные параметры (условия) для достижения указанной цели. Зависимость также может быть названием какого-то действия. Команды собственно действия, которые необходимо выполнить для достижения цели.Общий синтаксис Makefileимеет вид: target1 [target2…]:[[dependment1…][(tab)commands] [#commentary][(tab)commands] [#commentary]. Здесь знак # определяет начало комментария (содержимое от знака # и до конца строки не будет обрабатываться. Одинарное двоеточие указывает на то, что последовательность команд должна содержаться в одной строке. Для переноса можно в длинной строке команд можно использовать обратный слэш (). Двойное двоеточие указывает на то, что последовательность команд может содержаться в нескольких последовательных строках.Пример более сложного синтаксиса Makefile:## Makefile for abcd.c#CC = gccCFLAGS =# Compile abcd.c normalyabcd: abcd.c$(CC) -o abcd $(CFLAGS) abcd.cclean:-rm abcd .o ~# EndMakefileforabcd.c. В этом примере в начале файла заданы три переменные: CC и CFLAGS. Затем указаны цели, их зависимости и соответствующие команды. В командах происходит обращение к значениям переменных. Цель с именем cleanпроизводит очистку каталога от файлов, полученных в результате компиляции. Для её описания использованы регулярные выражения.

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

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 GNUдля ОС типа UNIXвходит отладчик GDB(GNUDebugger). Для использования GDB необходимо скомпилировать анализируемый код программы таким образом, чтобы отладочная информация содержалась в результирующем бинарном файле. Для этого следует воспользоваться опцией -gкомпилятора gcc: gcc-cfile.c-g. После этого для начала работы с gdbнеобходимо в командной строке ввести одноимённую команду, указав в качестве аргумента анализируемый бинарный файл: gdbfile.o

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

Основные команды отладчика gdb: 1. backtrace вывод на экран пути к текущей точке останова (по сутивывод названий всех функций); 2. break установить точку останова (в качестве параметра можетбыть указан номер строки или название функции); 3. clear удалить все точки останова в функции; 4. continue продолжить выполнение программы; 5. delete удалить точку останова; 6. display добавить выражение в список выражений, значения которых отображаются при достижении точки останова программы; 7. finish выполнить программу до момента выхода из функции; 8. info breakpoints вывести на экран список используемых точек останова; 9. info watchpoints вывести на экран список используемых контрольных выражений; 10. list вывести на экран исходный код (вВ ходе выполнения данной лабораторной работы я приобрелапростейшие навыки разработки, анализа, тестирования и отладки приложений в ОС типа UNIX/Linuxна примере создания на языке программирования С калькулятора с простейшими функциями. качестве параметра может быть указано название файла и через двоеточие номера начальнойи конечной строк); 11. next выполнить программу пошагово, но без выполнения вызываемых в программе функций; 12. print вывести значение указываемого в качестве параметра выражения; 13. run запуск программы

на выполнение; 14. set установить новое значение переменной; 15. step пошаговое выполнение программы; 16. watch установить контрольное выражение, при изменении значения которого программа будет остановлена. Для выхода из gdbможно воспользоваться командой quit (или её сокращённым вариантом q) или комбинацией клавиш Ctrl d. Более подробную информацию по работе с gdb можно получить с помощью команд gdb h и mangdb.

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

Схема отладки программы показана в 6 пункте лабораторной работы.

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

При первом запуске компилятор не выдал никаких ошибок, но в коде программы main.c допущена ошибка, которую компилятор мог пропустить (возможно, из-за версии 8.3.0-19): в строке scanf(“%s”, &Operation); нужно убрать знак &, потому что имя массивасимволов уже является указателемна первый элементэтого массива.

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

Система разработки приложений UNIX предоставляет различные средства, повышающие понимание исходного кода. К ним относятся: cscope исследование функций, содержащихся в программе, lint критическая проверка программ, написанных на языке Си.

12. Каковы основные задачи, решаемые программой splint?

Утилита splint анализирует программный код, проверяет корректность задания аргументов использованных в программе функций и типов возвращаемых

значений, обнаруживает синтаксические и семантические ошибки. В отличие от компилятора Санализатор splintгенерирует комментарии с описанием разбора кода программы и осуществляет общий контроль, обнаруживая такие ошибки, как одинаковые объекты, определённые в разных файлах, или объекты, чьи значения не используются в работт программы, переменные с некорректно заданными значениямии типами и многое другое.