# Цель работы

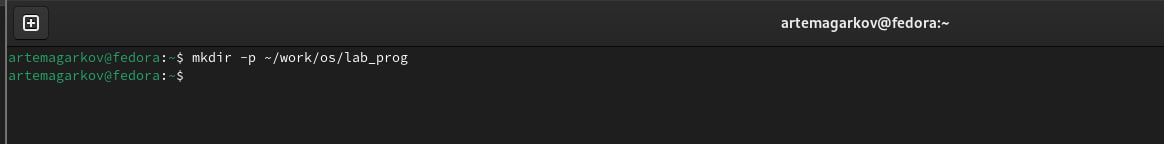
Приобрести простейшие навыки разработки, анализа, тестирования и отладки приложений в ОС типа UNIX/Linux на примере создания на языке программирования С калькулятора с простейшими функциями.

# Задание

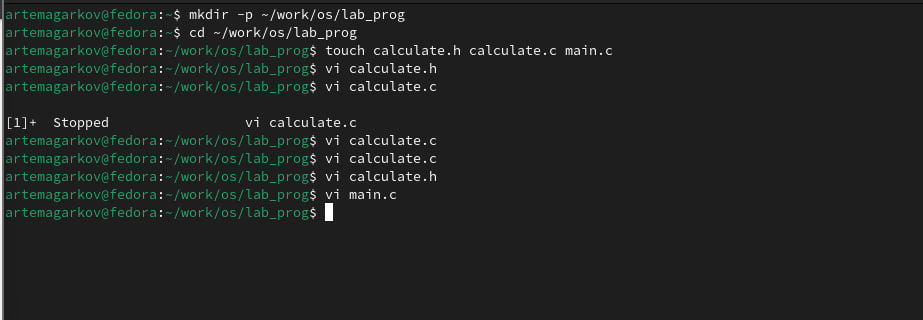
1. В домашнем каталоге создайте подкаталог ~/work/os/lab\_prog.
2. Создайте в нём файлы: calculate.h, calculate.c, main.c. Это будет примитивнейший калькулятор, способный складывать, вычитать, умножать и делить, возводить число в степень, брать квадратный корень, вычислять sin, cos, tan. При запуске он будет запрашивать первое число, операцию, второе число. После этого программа выведет результат и остановится. Реализация функций калькулятора в файле calculate.h:  
   Интерфейсный файл calculate.h, описывающий формат вызова функциикалькулятора:  
   Основной файл main.c, реализующий интерфейс пользователя к калькулятору:
3. Выполните компиляцию программы посредством gcc:
4. При необходимости исправьте синтаксические ошибки.
5. Создайте Makefile со следующим содержанием:
6. С помощью gdb выполните отладку программы calcul (перед использованием gdb исправьте Makefile): – Запустите отладчик GDB, загрузив в него программу для отладки:
7. С помощью утилиты splint попробуйте проанализировать коды файлов calculate.c и main.c.

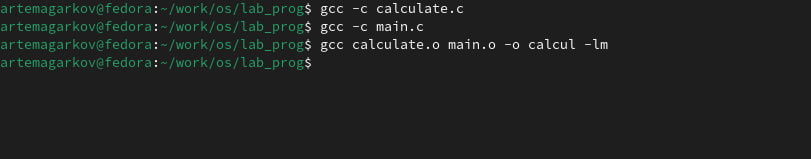
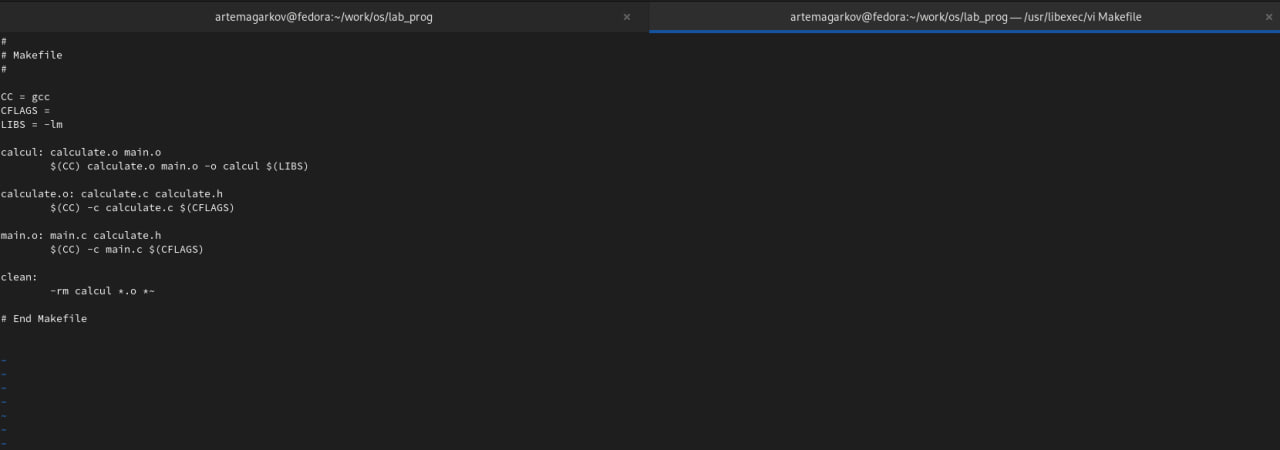
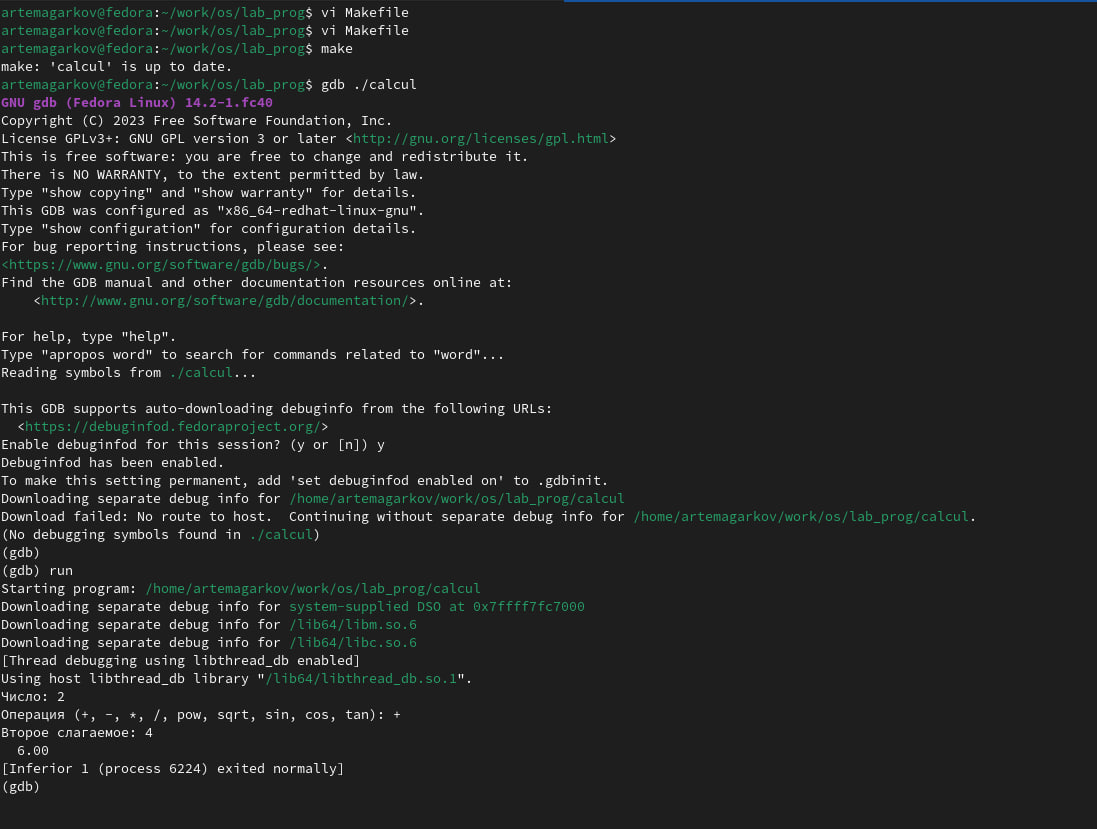
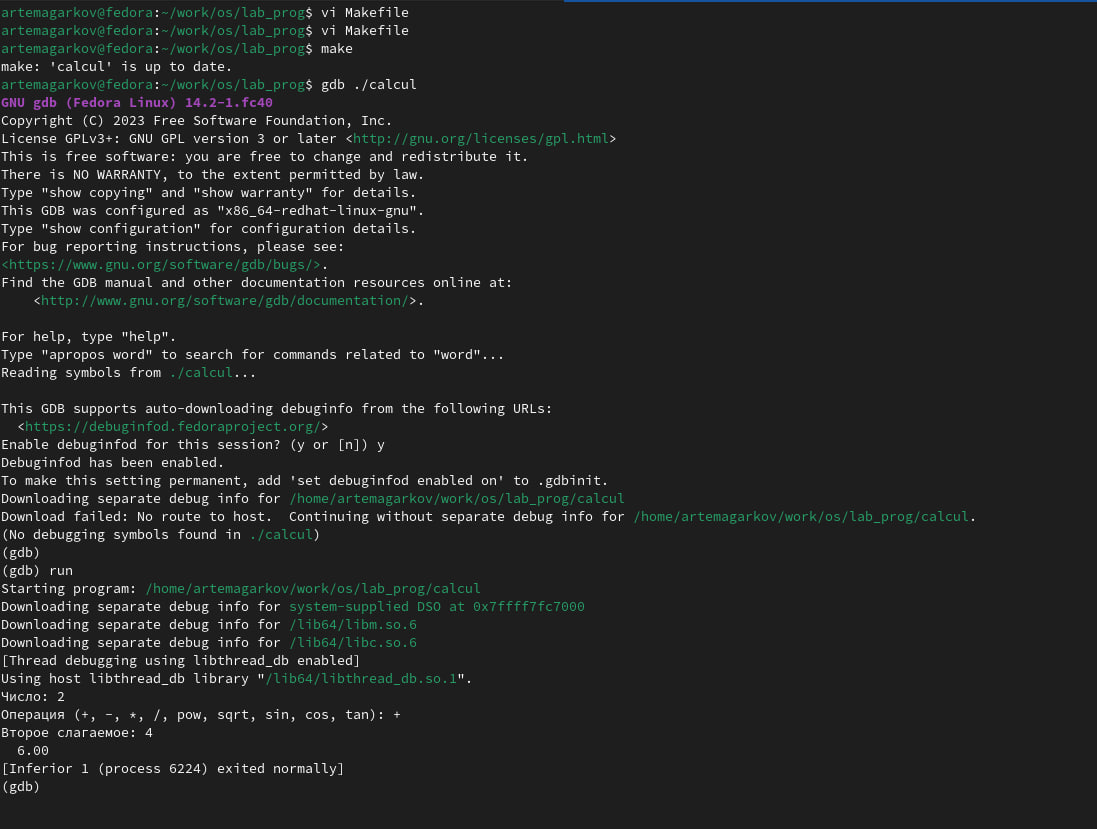
# Выполнение лабораторной работы

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



1. 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. Реализация функций калькулятора в файле calculate.h:  
   Интерфейсный файл calculate.h, описывающий формат вызова функциикалькулятора:  
   Основной файл main.c, реализующий интерфейс пользователя к калькулятору:



1. Выполните компиляцию программы посредством gcc:  
   
2. Создайте Makefile со следующим содержанием.  
   
3. С помощью gdb выполните отладку программы calcul (перед использованием gdb исправьте Makefile): – Запустите отладчик GDB, загрузив в него программу для отладки:  
    
4. С помощью утилиты splint попробуйте проанализировать коды файлов calculate.c и main.c.  
   

# 13.5. Контрольные вопросы и ответы

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

* Информацию о возможностях программ gcc, make, gdb и других утилит можно получить из их документации, которая часто включена в состав системы UNIX. Кроме того, можно использовать команды --help или -h, чтобы получить краткую справку о доступных опциях и ключах.

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

* Написание исходного кода на языке программирования (например, C, C++, Python).
* Компиляция исходного кода в исполняемый файл с помощью компилятора (например, gcc).
* Создание Makefile для автоматизации сборки программы.
* Отладка программы с использованием отладчика (например, gdb).
* Тестирование программы на различных входных данных.
* Улучшение и оптимизация кода.

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

* В контексте языка программирования суффикс - это часть имени файла, которая указывает на тип или формат файла. Например, в файлах исходного кода на C обычно используются суффиксы .c, .h, а в исполняемых файлах - суффикс .exe (для Windows) или отсутствует (для UNIX).

1. Каково основное назначение компилятора языка С в UNIX?

* Основное назначение компилятора языка С в UNIX - это преобразование исходного кода на языке C в машинный код, который может быть выполнен на конкретной архитектуре процессора.

1. Для чего предназначена утилита make?

* Утилита make предназначена для автоматизации процесса сборки программы из исходных файлов. Она читает файл Makefile, содержащий инструкции о том, какие файлы нужно компилировать и как их компилировать, и выполняет соответствующие действия.

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

* target: dependencies command1 command2 ...

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

* Основное свойство, присущее всем программам отладки, - это возможность пошагового выполнения программы и анализа её состояния во время выполнения. Для использования этой возможности необходимо скомпилировать программу с отладочной информацией и запустить её в отладчике.

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

* run: запуск программы.
* break: установка точки останова.
* list: просмотр исходного кода.
* print: вывод значения переменной.
* next: выполнение следующей строки.
* step: выполнение следующей строки с заходом в функции.
* continue: продолжение выполнения программы до следующей точки останова.

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

* Схема отладки программы включает в себя компиляцию программы с отладочной информацией, запуск программы в отладчике, установку точек останова, пошаговое выполнение программы, анализ переменных и структуры программы для выявления ошибок.

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

* При первом запуске компилятора gcc программа выводит сообщения об ошибках и предупреждениях в коде программы, если они есть. Это могут быть синтаксические ошибки, неправильное использование функций или другие проблемы, которые необходимо исправить.

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

* Основные средства, повышающие понимание исходного кода программы, включают в себя комментарии в коде, разбиение кода на логические блоки, использование осмысленных имен переменных и функций, а также документацию к программе.

1. Каковы основные задачи, решаемые программой splint?12.

* Программа splint предназначена для анализа исходного кода на языке программирования C и выявления потенциальных проблем с его безопасностью и надежностью. Основные задачи, решаемые программой splint, включают обнаружение утечек памяти, неинициализированных переменных, ошибок в использовании указателей, а также других типов ошибок, которые могут привести к непредсказуемому поведению программы или уязвимостям безопасности.

# Выводы

В результате выполнения лабораторной работы были приобретены простейшие навыки разработки, анализа, тестирования и отладки приложений в операционных системах типа UNIX/Linux. Процесс создания калькулятора на языке программирования C включал в себя создание файлов с кодом (calculate.c, main.c, calculate.h), компиляцию и сборку программы с использованием утилит gcc и make, а также отладку с помощью отладчика gdb. В ходе работы были освоены основные этапы разработки приложений в UNIX, включая написание и отладку кода, использование инструментов сборки и отладки, а также анализ кода с использованием инструментов статического анализа, таких как splint. Полученные навыки позволяют создавать простые программы и эффективно их отлаживать и анализировать.