Отчёт по лабораторной работе

Лабораторная работа № 13

Мурзаев Замир Зейнадинович

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Приобрести простейшие навыки разработки, анализа, тестирования и отладки при- ложений в ОС типа UNIX/Linux на примере создания на языке программирования C 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.
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):

3 Выполнение лабораторной работы

В домашнем каталоге создаем подкаталог (рис. 3.1).

```
zzmurzaev@dk5n51 ~ $ cd work/os
zzmurzaev@dk5n51 ~/work/os $ mkdir lab_prog
zzmurzaev@dk5n51 ~/work/os $
```

Рис. 3.1: Создание подкаталога

Создаем в нем файлы (рис. 3.2).

```
zzmurzaev@dk5n51 ~/work/os $ touch calculate.h
zzmurzaev@dk5n51 ~/work/os $ touch calculate.c
zzmurzaev@dk5n51 ~/work/os $ touch main.c
```

Рис. 3.2: Создание файлов

Заполняем calculate.c (рис. 3.3).

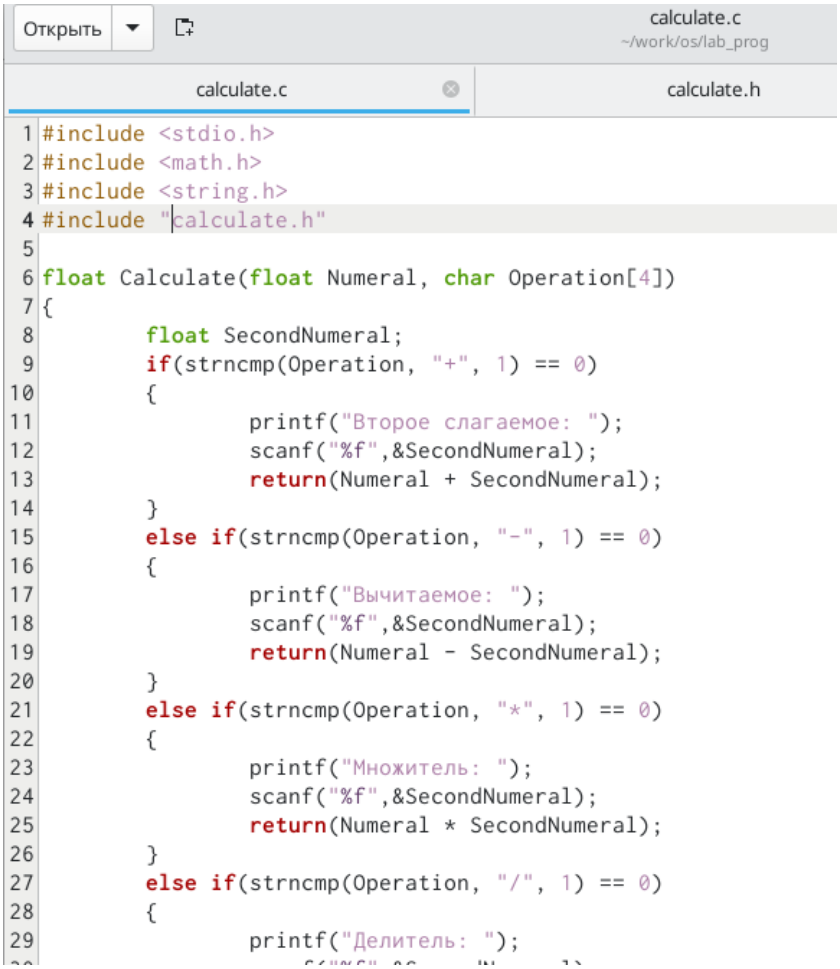

Рис. 3.3: Программа на C

Заполняем calculate.h (рис. 3.4).

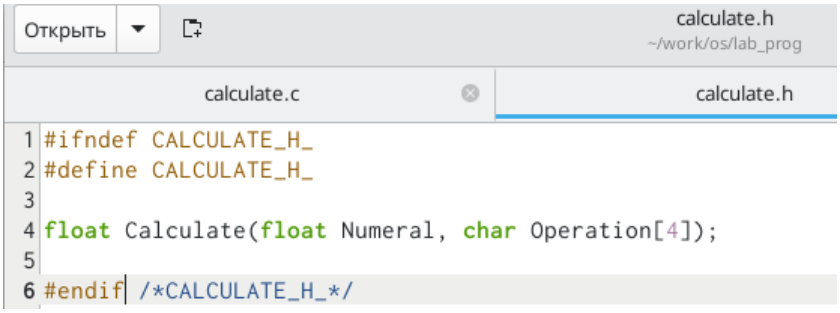

Рис. 3.4: Программа на C

Заполняем main.c (рис. 3.5).

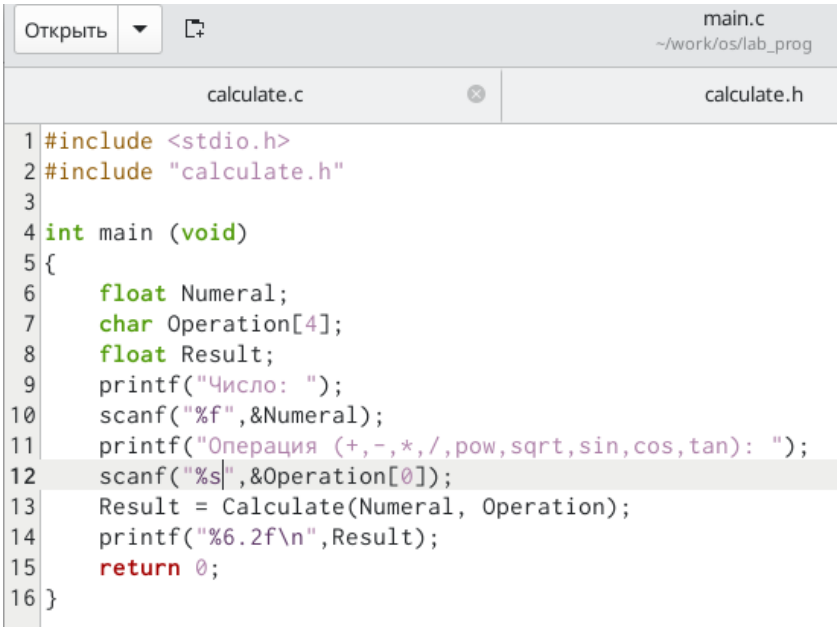

Рис. 3.5: Программа на C

Компилируем файлы (рис. 3.6).

```
zzmurzaev@dk5n51 ~/work/os/lab_prog $ gcc -c main.c
zzmurzaev@dk5n51 ~/work/os/lab_prog $ gcc calculate.o main.o -o calcul -lm
```

Рис. 3.6: gcc

Создаем Makefile (рис. 3.7).

В содержании файла указаны флаги компиляции, тип компилятора и файлы, которые должен собрать сборщик.

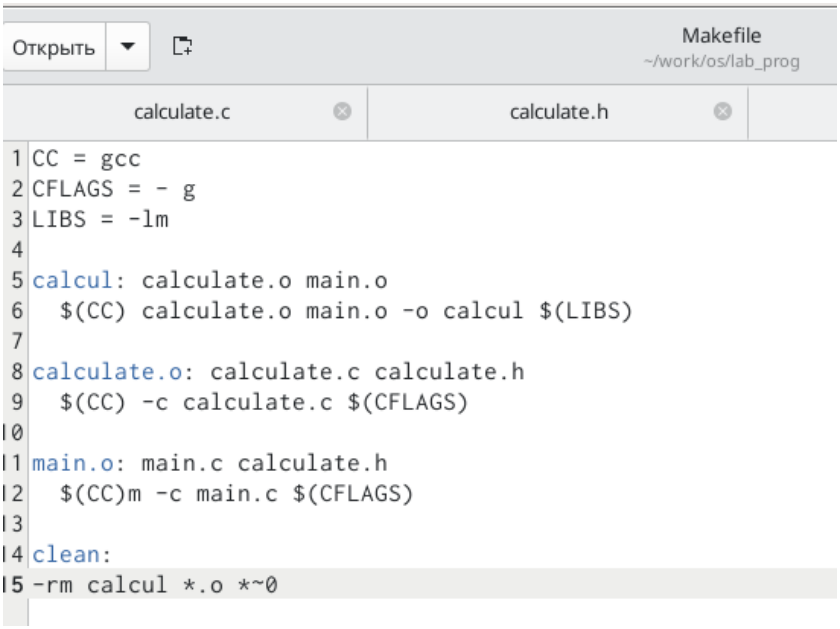

Рис. 3.7: Makefile

Проверяем работу калькулятора (рис. **??**).

```
zzmurzaev@dk5n51 ~/work/os/lab_prog $ ./calcul
Число: 3
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): -
Вычитаемое: 2
  1.00
```

Выводы

Приобретены простейшие навыки разработки, анализа, тестирования и отладки при- ложений в ОС типа UNIX/Linux на примере создания на языке программирования C калькулятора с простейшими функциями.

4 Ответы на вопросы

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

Ответ: при помощи программы man.

Назовите и дайте краткую характеристику основным этапам разработки приложений в U

Ответ: 1. Выбор названия 2. Выбор языка программирования 3. Попытка выполнить работу всю зараз 4. Отрицание 5. Гнев 6. Торг 7. Депрессия 8. Принятие

Что такое суффикс в контексте языка программирования? Приведите примеры использов

Ответ: финальная часть названия программы, обычно отделяемая точкой.

Каково основное назначение компилятора языка C в UNIX?

Ответ: компилятор языка C в UNIX в основном компилирует программы языка C в UNIX, написанные на языке C в UNIX.

Для чего предназначена утилита make?

Ответ: При разработке большой программы, состоящей из нескольких исходных файлов заголовков, приходится постоянно следить за файлами, которые требуют перекомпиляции после внесения изменений. Программа make освобождает пользователя от такой рутинной работы и служит для документирования взаимосвязей между файлами. Описание взаимосвязей и соответствующих действий хранится в так называемом make-файле, который по умолчанию имеет имя makefile или Makefile.

Приведите пример структуры Makefile. Дайте характеристику основным элементам этог

Ответ: В общем случае make-файл содержит последовательность записей (строк), определяющих зависимости между файлами. Первая строка записи представляет собой список целевых (зависимых) файлов, разделенных пробелами, за которыми следует двоеточие и список файлов, от которых зависят целевые. Текст, следующий за точкой с запятой, и все последующие строки, начинающиеся с литеры табуляции, являются командами ОС UNIX, которые необходимо выполнить для обновления целевого файла. Таким образом, спецификация взаимосвязей имеет формат: target1 [target2…]: [:] [dependment1…] [(tab)commands] [#commentary] [(tab)commands] [#commentary], где # — специфицирует начало комментария, так как содержимое строки, начиная с # и до конца строки, не будет обрабатываться командой make; : — последовательность команд ОС UNIX должна содержаться в одной строке make-файла (файла описаний), есть возможность переноса команд (), но она считается как одна строка; :: — последовательность команд ОС UNIX может содержаться в нескольких последовательных строках файла описаний. Приведённый выше make-файл для программы abcd.c включает два способа компиляции и построения исполняемого модуля. Первый способ предусматривает обычную компиляцию с построением исполняемого модуля с именем abcd. Второй способ позволяет включать в исполняемый модуль testabcd возможность выполнить процесс отладки на уровне исходного текста. Пример можно найти в задании 5.

Назовите основное свойство, присущее всем программам отладки. Что необходимо сдел

Ответ: свойство - анализ кода; для анализа необходимо скомпилировать программу.

Назовите и дайте основную характеристику основным командам отладчика gdb.

Ответ: см. ответ к вопросу 6.

Опишите по шагам схему отладки программы, которую вы использовали при выполнении

Ответ: 1. Вначале я запустил gdb 2. Затем я его закрыл

Прокомментируйте реакцию компилятора на синтаксические ошибки в программе при его

Ответ: когда я увидел реакцию компилятора на синтаксические ошибки в программе при его первом запуске,я был возмущён, поражён, обескуражен, ошеломлён, фрустрирован и изумлён. Но использовал совершенно другие выражения.

Назовите основные средства, повышающие понимание исходного кода программы.

Ответ: Если вы работаете с исходным кодом, который не вами разрабатывался, то назначение различных конструкций может быть не совсем понятным. Система разработки приложений UNIX предоставляет различные средства, повышающие понимание исходного кода. К ним относятся: – cscope - исследование функций, содержащихся в программе; – splint — критическая проверка программ, написанных на языке Си.

Каковы основные задачи, решаемые программой splint?

Ответ: анализ кода. # Список литературы{.unnumbered}