# Шаблон отчёта по лабораторной работе N.13

Кристина Эспиноса

# Содержание

**1 Цель работы** **5**

**2 Выполнение лабораторной работы** **6**

**3 Контрольные вопросы** **9**

# Список иллюстраций

2.1 . . . . . 7

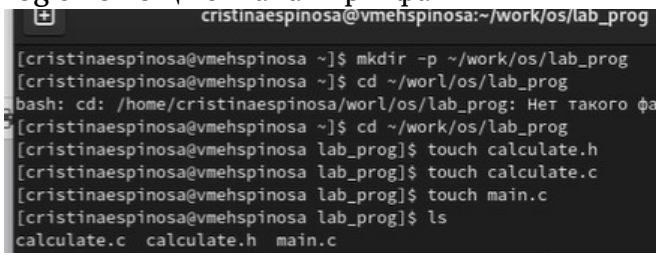
# Список таблиц

# 1 Цель работы

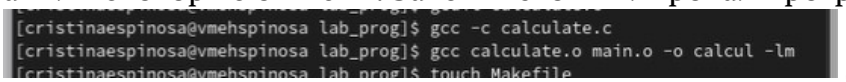
Приобрести простейшие навыки разработки, анализа, тестирования и отладки при- ложений в ОС типа UNIX/Linux на примере создания на языке программирования C калькулятора с простейшими функциями.

# 2 Выполнение лабораторной работы

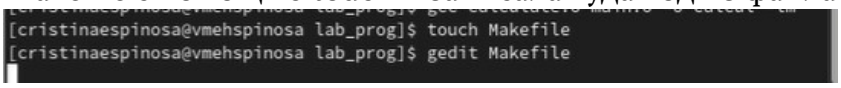
Первым шагом я создала каталог ~/work/os/lab_prog с помощью makdir -p и файлы calculate.h, calculate.c, main.c. с помощью touch.




Далее я писал программы в эти файлы. Я взял листинги из лабораторного файла, исправил некоторые ошибки. Затем я скомпилировал программу с помощью gcc.




Далее я создала Makefile с помощью touch и записала туда код из файла Лабораторной работы.




После я запустила отладчик GDB командой gdb ./calcul

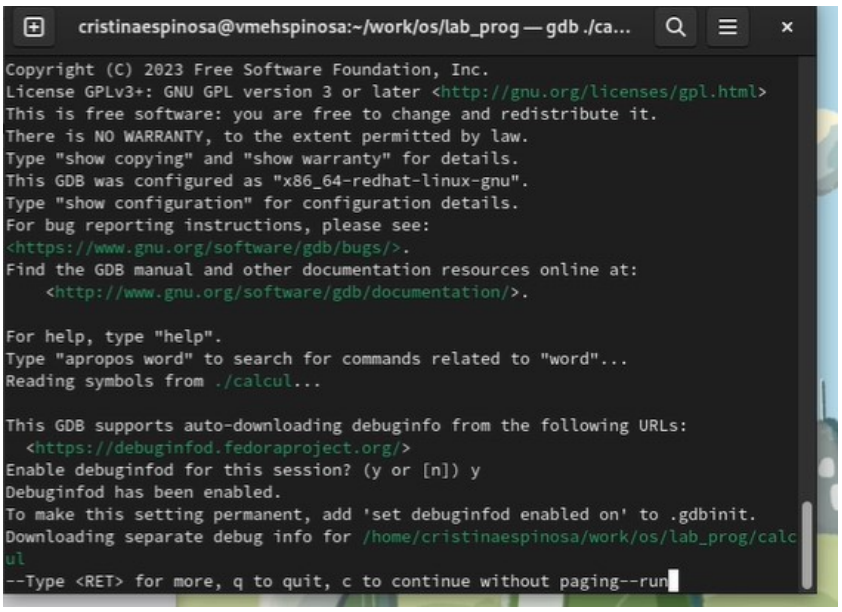
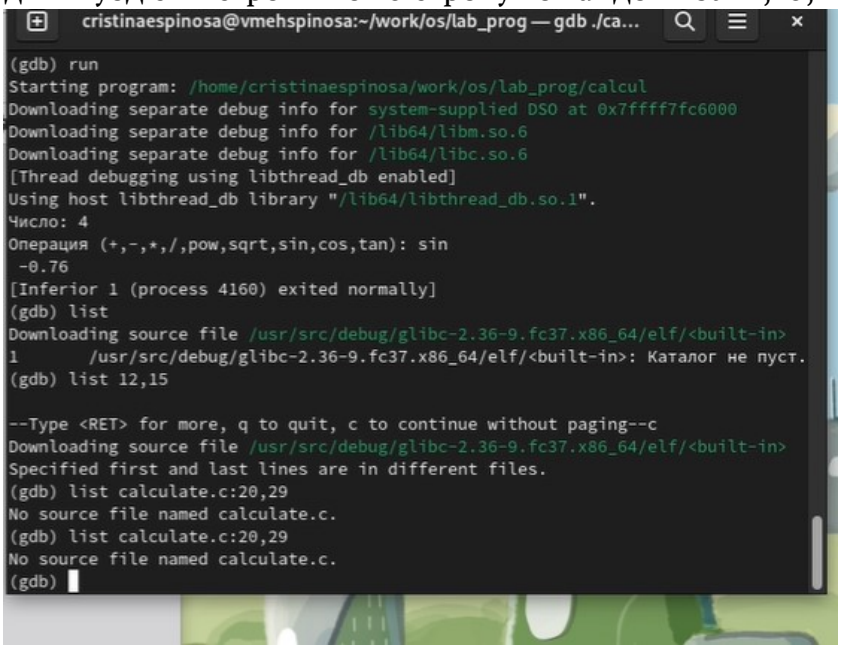
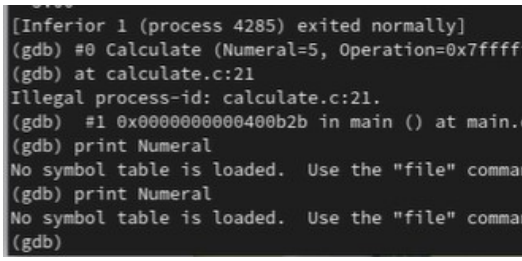



Рис. 2.1:

Постранично просмотрела код, используя list, а после хотела посмотреть исходный уод с 12 строки по 15 строку командой list 12,15, но нечего не показалы.




Последним шакгом с помощью утилиты splint проанализировала коды файлов

```
[cristinaespinosa@vmehspinosa lab_prog]$ splint main.c
Splint 3.1.2 --- 23 Jul 2022

calculate.h:5:37: Function parameter Operation declared as manifest array (size
                    constant is meaningless)
  A formal parameter is declared as an array with size.  The size of the array
  is ignored in this context, since the array formal parameter is treated as a
  pointer. (Use -fixedformalarray to inhibit warning)
main.c: (in function main)
main.c:11:2: Return value (type int) ignored: scanf("%f", &Num...
  Result returned by function call is not used. If this is intended, can cast
  result to (void) to eliminate message. (Use -retvalint to inhibit warning)
main.c:13:13: Format argument 1 to scanf (%s) expects char * gets char [4] *:
                &Operation
  Type of parameter is not consistent with corresponding code in format string.
  (Use -formattype to inhibit warning)
   main.c:13:10: Corresponding format code
main.c:13:2: Return value (type int) ignored: scanf("%s", &Ope...

Finished checking --- 4 code warnings
[cristinaespinosa@vmehspinosa lab_prog]$
```

calculate.c и main.c.

```
                    SecondNumeral == 0
  Two real (float, double, or long double) values are compared directly using
  == or != primitive. This may produce unexpected results since floating point
  representations are inexact. Instead, compare the difference to FLT_EPSILON
  or DBL_EPSILON. (Use -realcompare to inhibit warning)
calculate.c:38:8: Return value type double does not match declared type float:
                    (HUGE_VAL)
  To allow all numeric types to match, use +relaxtypes.
calculate.c:46:2: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:47:8: Return value type double does not match declared type float:
                    (pow(Numeral, SecondNumeral))
calculate.c:50:8: Return value type double does not match declared type float:
                    (sqrt(Numeral))
calculate.c:52:8: Return value type double does not match declared type float:
                    (sin(Numeral))
calculate.c:54:8: Return value type double does not match declared type float:
                    (cos(Numeral))
calculate.c:56:8: Return value type double does not match declared type float:
                    (tan(Numeral))
calculate.c:60:8: Return value type double does not match declared type float:
                    (HUGE_VAL)

Finished checking --- 15 code warnings
[cristinaespinosa@vmehspinosa lab_prog]$
```

# Выводы

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.

# 3 Контрольные вопросы

1. Как получить более полную информацию о программах: gcc, make, gdb и др.?

Дополнительную информацию о этих программах можно получить с помощью функций info и man.

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

Unix поддерживает следующие основные этапы разработки приложений:

создание исходного кода программы;

представляется в виде файла;

сохранение различных вариантов исходного текста;

анализ исходного текста; (необходимо отслеживать изменения исходного кода, а такж

компиляция исходного текста и построение исполняемого модуля;

тестирование и отладка;

проверка кода на наличие ошибок

сохранение всех изменений, выполняемых при тестировании и отладке.

3. Что такое суффиксы и префиксы? Основное их назначение. Приведите примеры их использования.

Использование суффикса “.c” для имени файла с программой на языке Си отражает удобное и полезное соглашение, принятое в ОС UNIX. Для любого имени входного файла суффикс определяет какая компиляция требуется. Суффиксы и префиксы указывают тип объекта. Одно из полезных свойств компилятора Си — его способность по суффиксам определять типы файлов. По суффиксу .c компилятор распознает, что файл abcd.c должен компилироваться, а по суффиксу .o, что файл abcd.o является объектным модулем и для получения исполняемой программы необходимо выполнить редактирование связей. Простейший пример командной строки для компиляции программы abcd.c и построения исполняемого модуля abcd имеет вид: gcc -o abcd abcd.c. Некоторые проекты предпочитают показывать префиксы в начале текста изменений для старых (old) и новых (new) файлов. Опция – prefix может быть использована для установки такого префикса. Плюс к этому команда bzr diff -p1 выводит префиксы в форме которая подходит для команды patch -p1.

4. Основное назначение компилятора с языка Си в UNIX?

Основное назначение компилятора с языка Си заключается в компиляции всей программы в целом и получении исполняемого модуля.

5. Для чего предназначена утилика make?

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.

6. Приведите структуру make-файла. Дайте характеристику основным элементам этого файла.

makefile для программы abcd.c мог бы иметь вид:

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

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.

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

Пошаговая отладка программ заключается в том, что выполняется один оператор программы и, затем контролируются те переменные, на которые должен был воздействовать данный оператор. Если в программе имеются уже отлаженные подпрограммы, то подпрограмму можно рассматривать, как один оператор программы и воспользоваться вторым способом отладки программ. Если в программе существует достаточно большой участок программы, уже отлаженный ранее, то его можно выполнить, не контролируя переменные, на которые он воздействует. Использование точек останова позволяет пропускать уже отлаженную часть программы. Точка останова устанавливается в местах, где необходимо проверить содержимое переменных или просто проконтролировать, передаётся ли управление данному оператору. Практически во всех отладчиках поддерживается это свойство (а также выполнение программы до курсора и выход из подпрограммы). Затем отладка программы продолжается в пошаговом режиме с контролем локальных и глобальных переменных, а также внутренних регистров микроконтроллера и напряжений на выводах этой микросхемы.

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

```
backtrace – выводит весь путь к текущей точке останова, то есть названия всех фун
```

```
break - устанавливает точку останова; параметром может быть номер строки или назв

clear - удаляет все точки останова на текущем уровне стека (то есть в текущей фун

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 к

finish - выполняет программу до выхода из текущей функции; отображает возвращаемо

info breakpoints - выводит список всех имеющихся точек останова; - info watchpoir

splist - выводит исходный код; в качестве параметра передаются название файла исх

print - выводит значение какого-либо выражения (выражение передаётся в качестве п

run - запускает программу на выполнение;

set - устанавливает новое значение переменной

step - пошаговое выполнение программы;

watch - устанавливает контрольное выражение, программа остановится, как только зн
```

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

Выполнили компиляцию программы

Увидели ошибки в программе (если они есть)

Открыли редактор и исправили программу
Загрузили программу в отладчик gdb
run — отладчик выполнил программу, мы ввели требуемые значения.
программа завершена, gdb не видит ошибок.

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

Ошибки были, не работал в gdb команды list, и не смог найти файл calculate.c.

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

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.

12. Каковы основные задачи, решаемые программой slint?

Проверка корректности задания аргументов всех использованных в программе функций,

Поиск фрагментов исходного текста, корректных с точки зрения синтаксиса языка Си,

Общая оценка мобильности пользовательской программы.