**Лабораторная работа №11**

## Аксенова Анастасия

**Содержание**

1. [Цель работы](#_bookmark0) 5
2. [Задание](#_bookmark1) 6
3. [Выполнение лабораторной работы](#_bookmark2) 7
4. [Библиография](#_bookmark17) 18
5. [Выводы](#_bookmark18) 19

**List of Tables**

**List of Figures**

[3.1 Создание файла](#_bookmark3) . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

[3.2 Скрипт №1](#_bookmark4) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

* 1. [Предоставление прав доступа](#_bookmark5) . . . . . . . . . . . . . . . . . . . . 8
  2. [Проверка работы программы](#_bookmark6) . . . . . . . . . . . . . . . . . . . . 8 [3.5 Создание файлов](#_bookmark7) . . . . . . . . . . . . . . . . . . . . . . . . . . . 9 [3.6 Работа в файле chslo.c](#_bookmark8) 10
  3. [Работа в файле chslo.sh](#_bookmark9) 11
  4. [Проверка скрипта №2](#_bookmark10) 11
  5. [Создание файлов](#_bookmark11) 12
  6. [Скрипт №3](#_bookmark12) 12
  7. [Проверка работы скрипта №3](#_bookmark13) 13
  8. [Создание файлов](#_bookmark14) 13
  9. [Скрипт №4](#_bookmark15) 14
  10. [Проверка скрипта №4](#_bookmark16) 15

# Цель работы

Цель данной лабораторной работы — Изучить основы программирования в обо- лочке ОС UNIX. Научится писать более сложные командные файлы с использова- нием логических управляющих конструкций и циклов.

# Задание

* 1. Сделать отчёт по лабораторной работе №12 в формате Markdown.
  2. Научится писать более сложные командные файлы с использованием логи- ческих управляющих конструкций и циклов.

# Выполнение лабораторной работы

1. Используя команды getopts grep, написала командный файл, который анализи- рует командную строку с ключами: 1.-iinputfile—прочитатьданные из указанного файла; 2. -ooutputfile—вывести данные в указанный файл; 3. -p шаблон —указать шаблон для поиска; 4. -C—различать большие и малые буквы; 5. -n—выдавать номера строк,а затем ищет в указанном файле нужные строки, определяемые ключом –p. Для данной задачи я создала файл prog1.sh (Рисунки [3.1](#_bookmark3) ) и написала соответствующие скрипты. (алгоритм действий представлен на рис. [3.2](#_bookmark4) ).

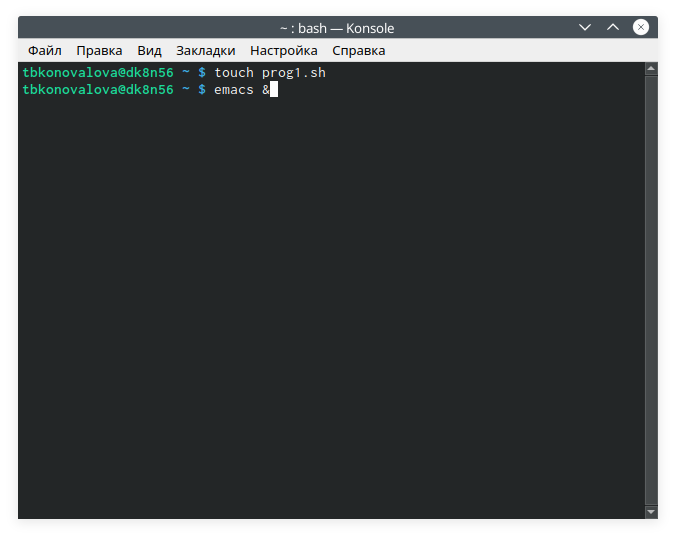


Figure 3.1: Создание файла

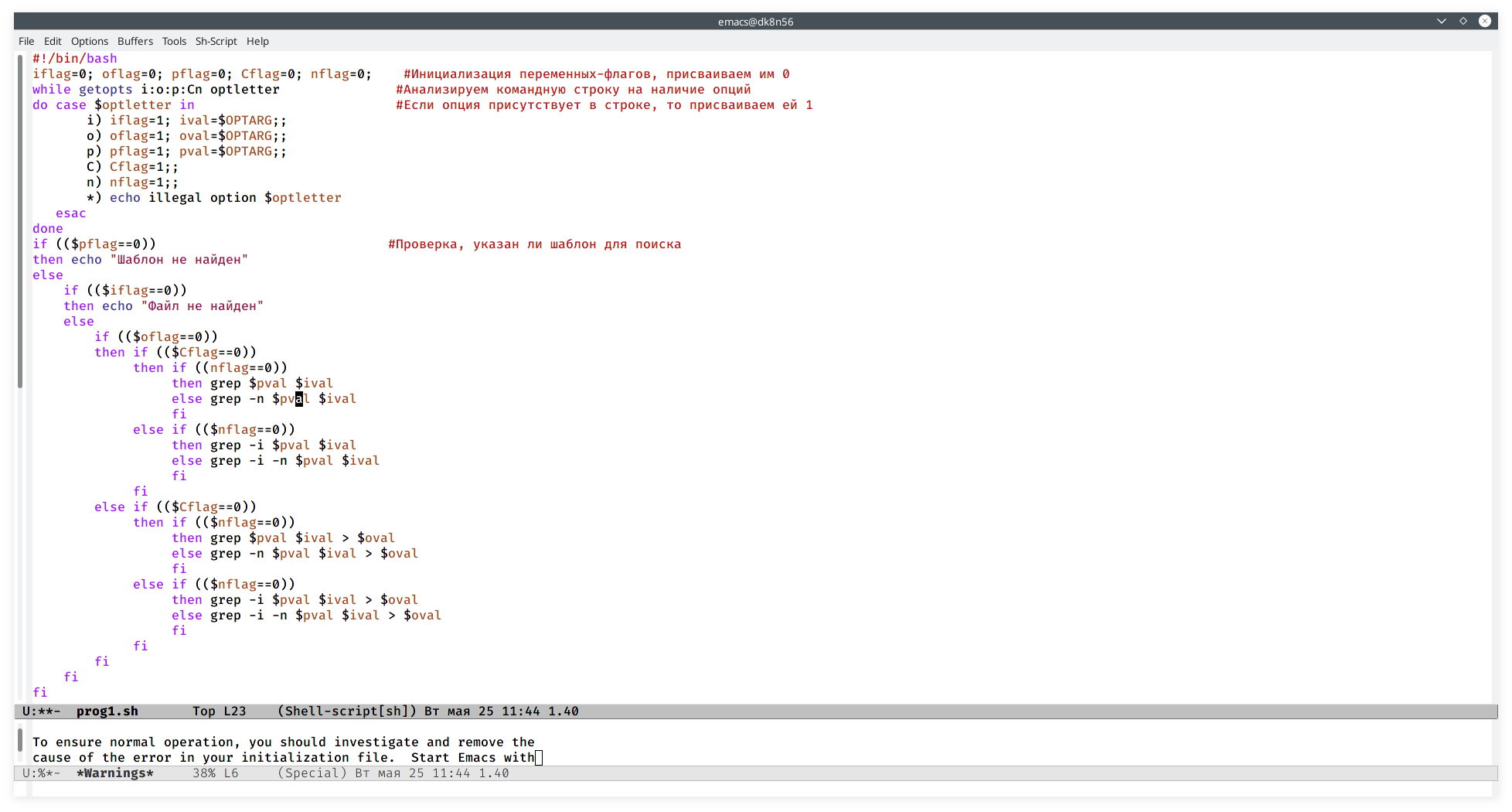


Figure 3.2: Скрипт №1

Проверила работу написанного скрипта, используя различные опции (напри- мер, команда «./prog.sh–Ia1.txt–oa2.txt–pcapital–C-n»), предварительно добавив право на исполнение файла (команда «chmod+xprog1.sh») и создав 2 файла, кото- рые необходимы для выполнения программы: a1.txt и a2.txt (алгоритм действий представлен на рис. [3.3](#_bookmark5) , [3.4](#_bookmark6) ). Скрипт работает корректно.

Figure 3.3: Предоставление прав доступа

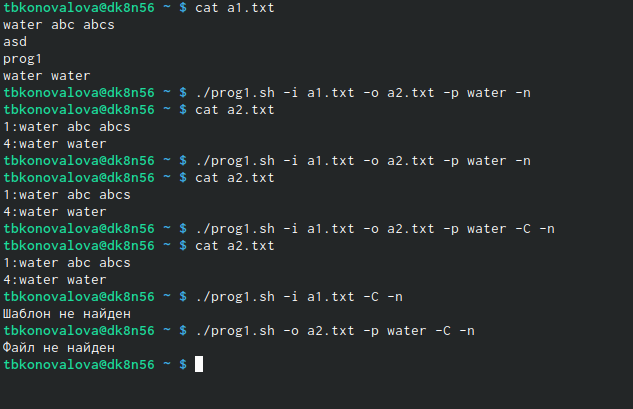


Figure 3.4: Проверка работы программы

* 1. Написала на языке Си программу, которая вводит число и определяет, явля- ется ли оно больше нуля, меньше нуля или равно нулю. Затем программа завер- шается с помощью функции exit(n), передавая информацию в о коде завершения в оболочку. Командный файл должен вызывать эту программу и, проанализиро- вав с помощью команды $?, выдать сообщение о том, какое число было введено. Для данной задачи я создала 2 файла: chslo.c и chislo.sh (Рисунок [3.5](#_bookmark7) ) и написала соответствующие скрипты. (команды «touch prog2.sh» и «emacs &») (Скриншоты [3.6](#_bookmark8) , [3.7](#_bookmark9) ).

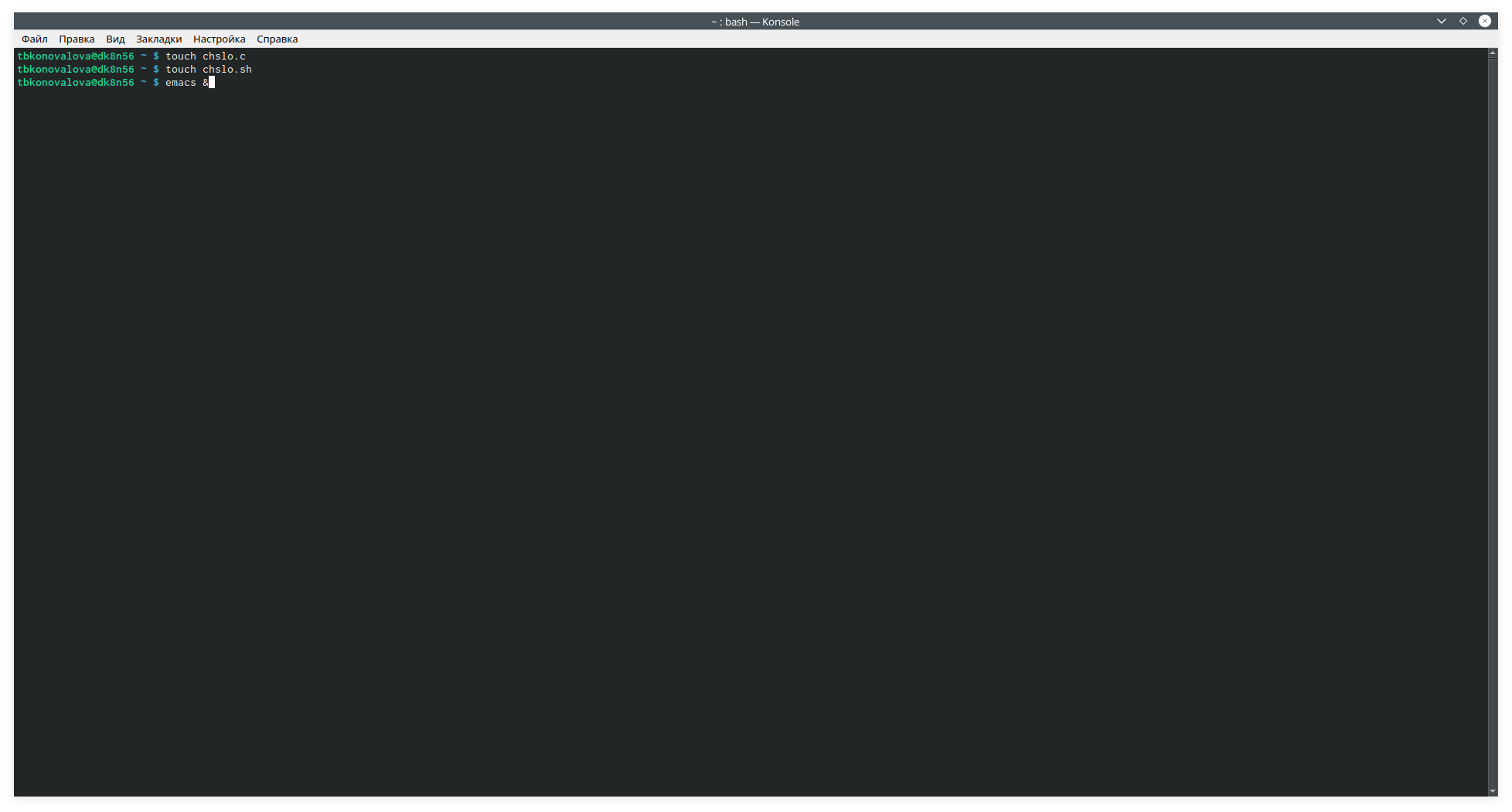


Figure 3.5: Создание файлов

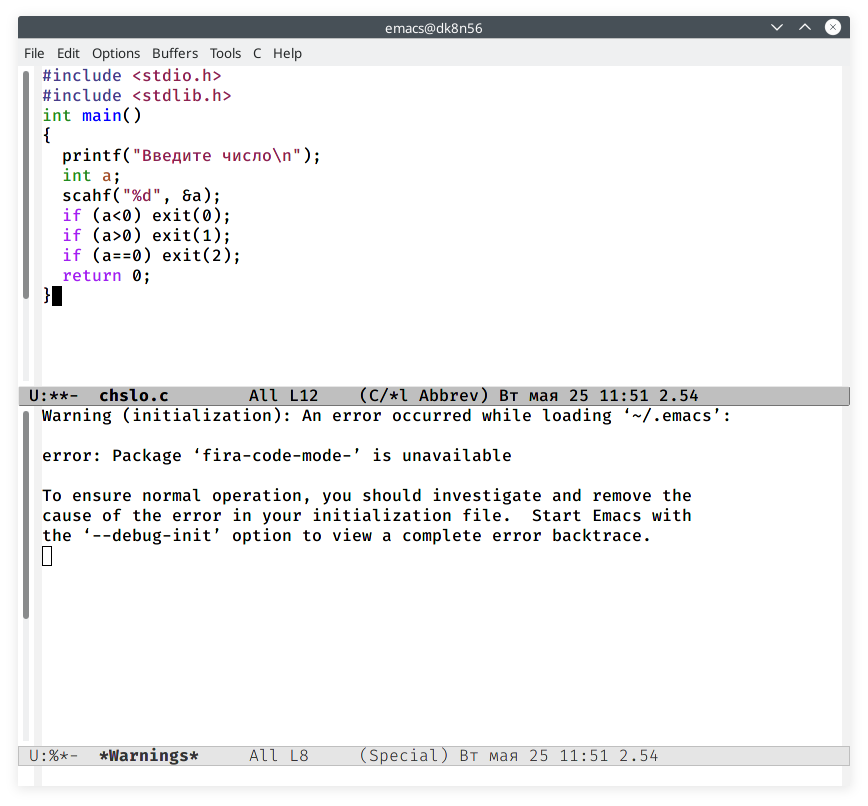


Figure 3.6: Работа в файле chslo.c

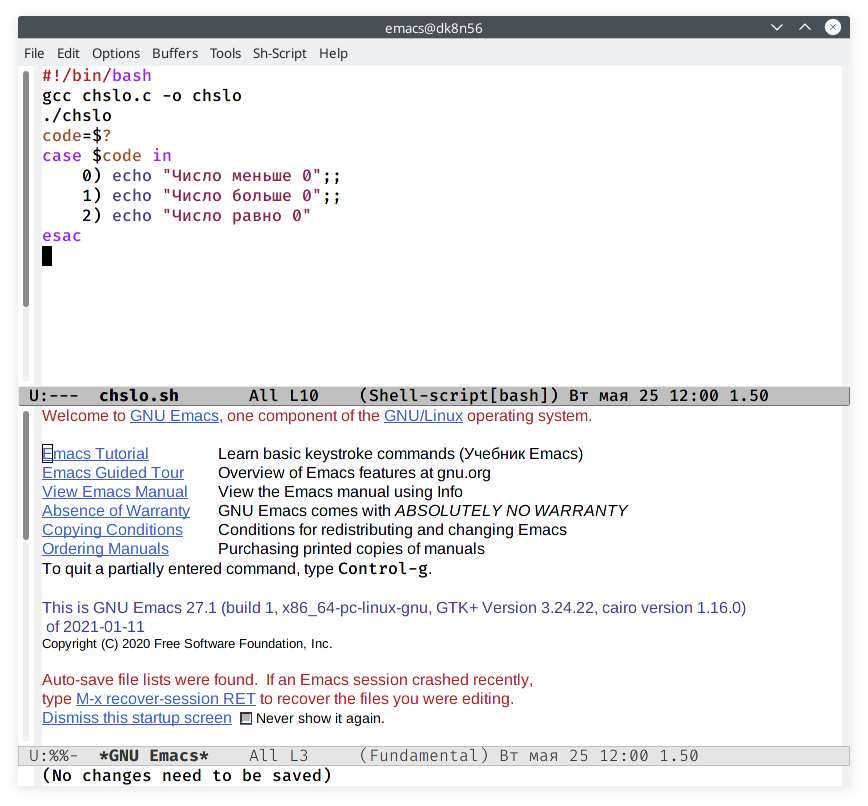


Figure 3.7: Работа в файле chslo.sh

Проверила работу написанных скриптов (команда «./chislo.sh»), предваритель- но добавив право на исполнение файла (команда «chmod+x chislo.sh») (Рисунок

[3.8](#_bookmark10) ). Скрипты работают корректно.

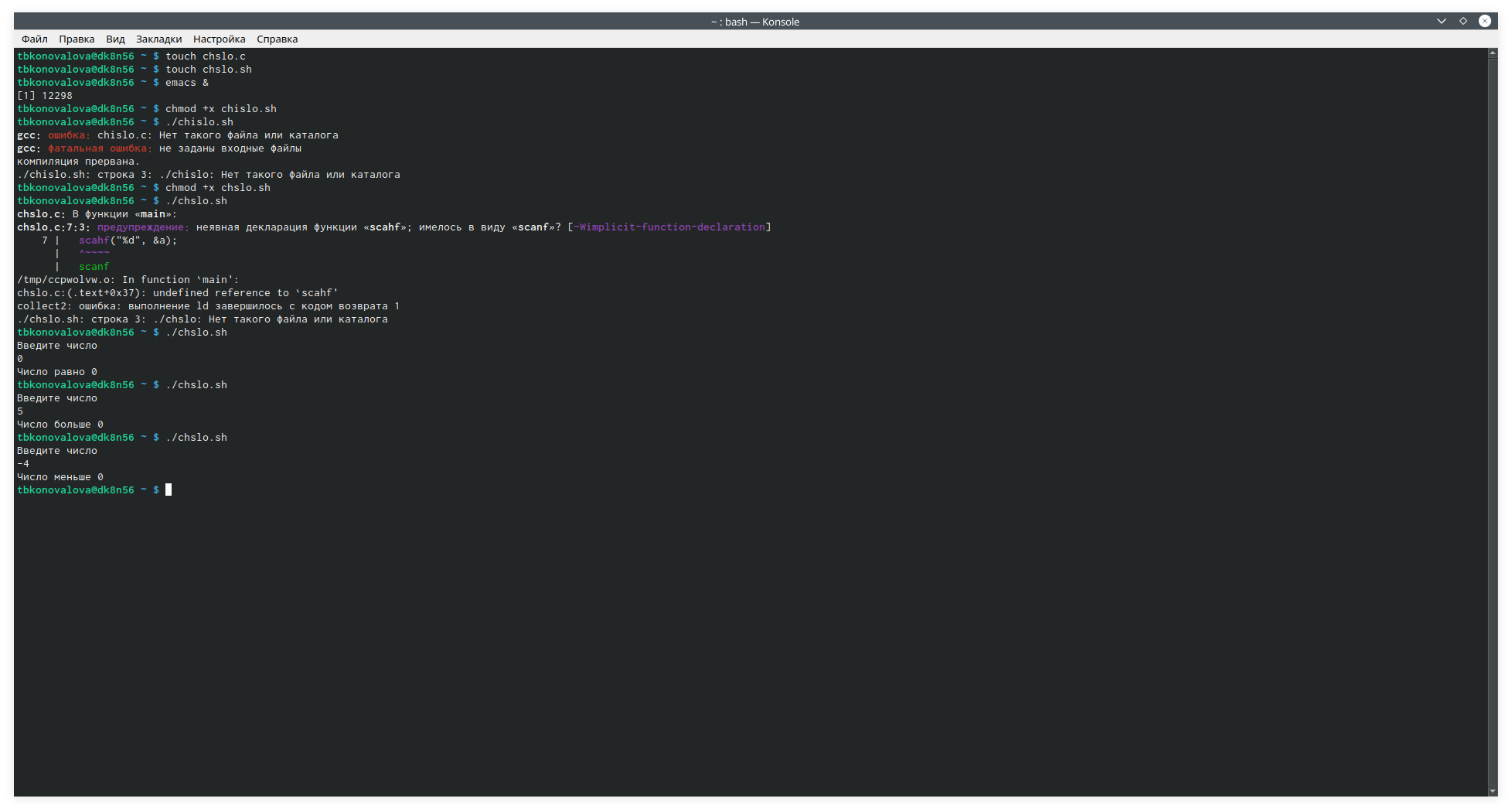


Figure 3.8: Проверка скрипта №2

* 1. Написала командный файл, создающий указанное число файлов, пронуме-

рованных последовательно от 1 до N (например 1.tmp, 2.tmp, 3.tmp,4.tmpи т.д.). Число файлов, которые необходимо создать, передаётся в аргументы командной строки. Этот же командный файл должен уметь удалять все созданные им файлы (если они существуют). Для данной задачи я создала файл: files.sh (Рисунок [3.9](#_bookmark11)

). и написала соответствующий скрипт (алгоритм действий представлен на рис.

[3.10](#_bookmark12) ).

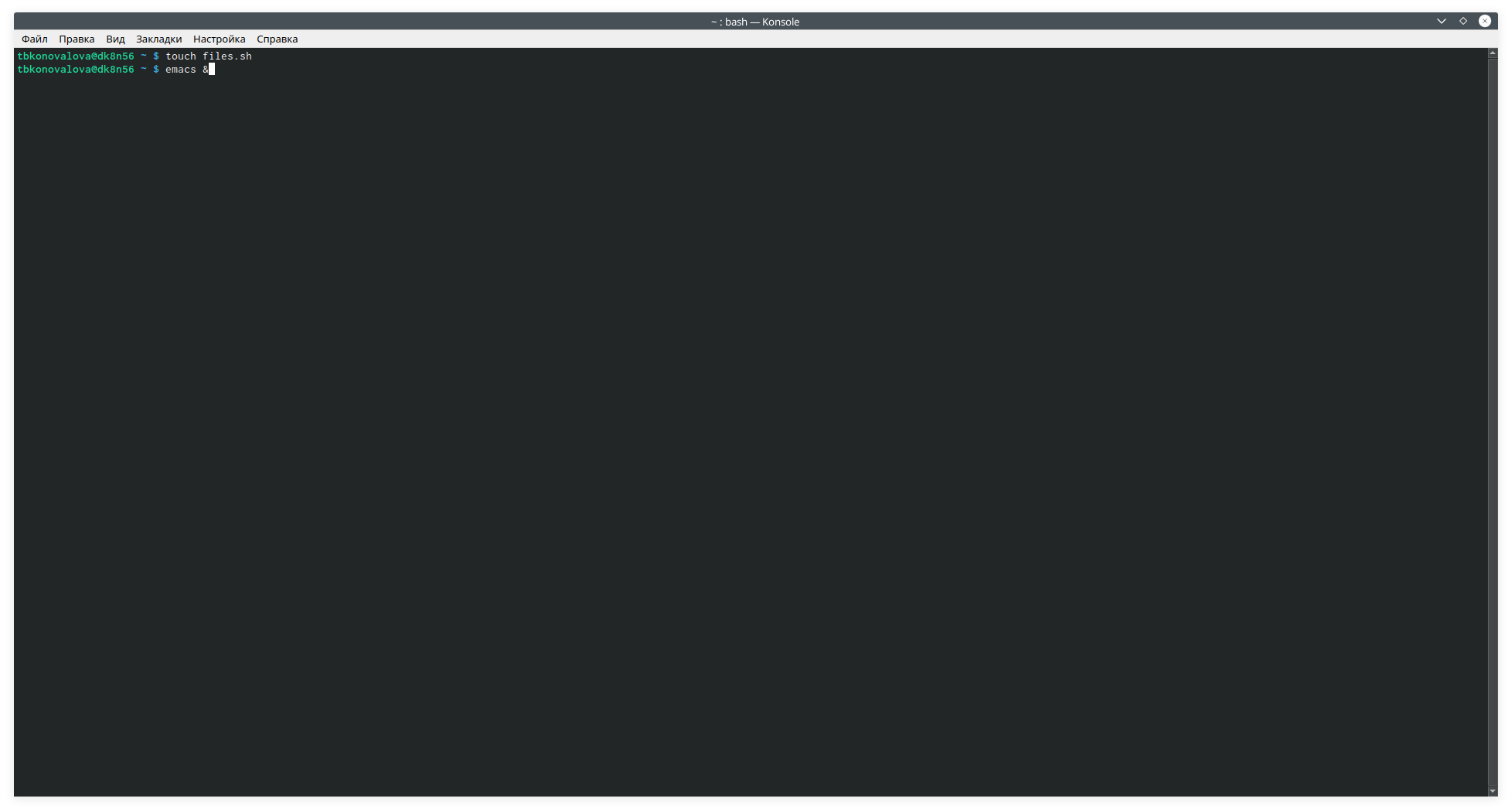


Figure 3.9: Создание файлов

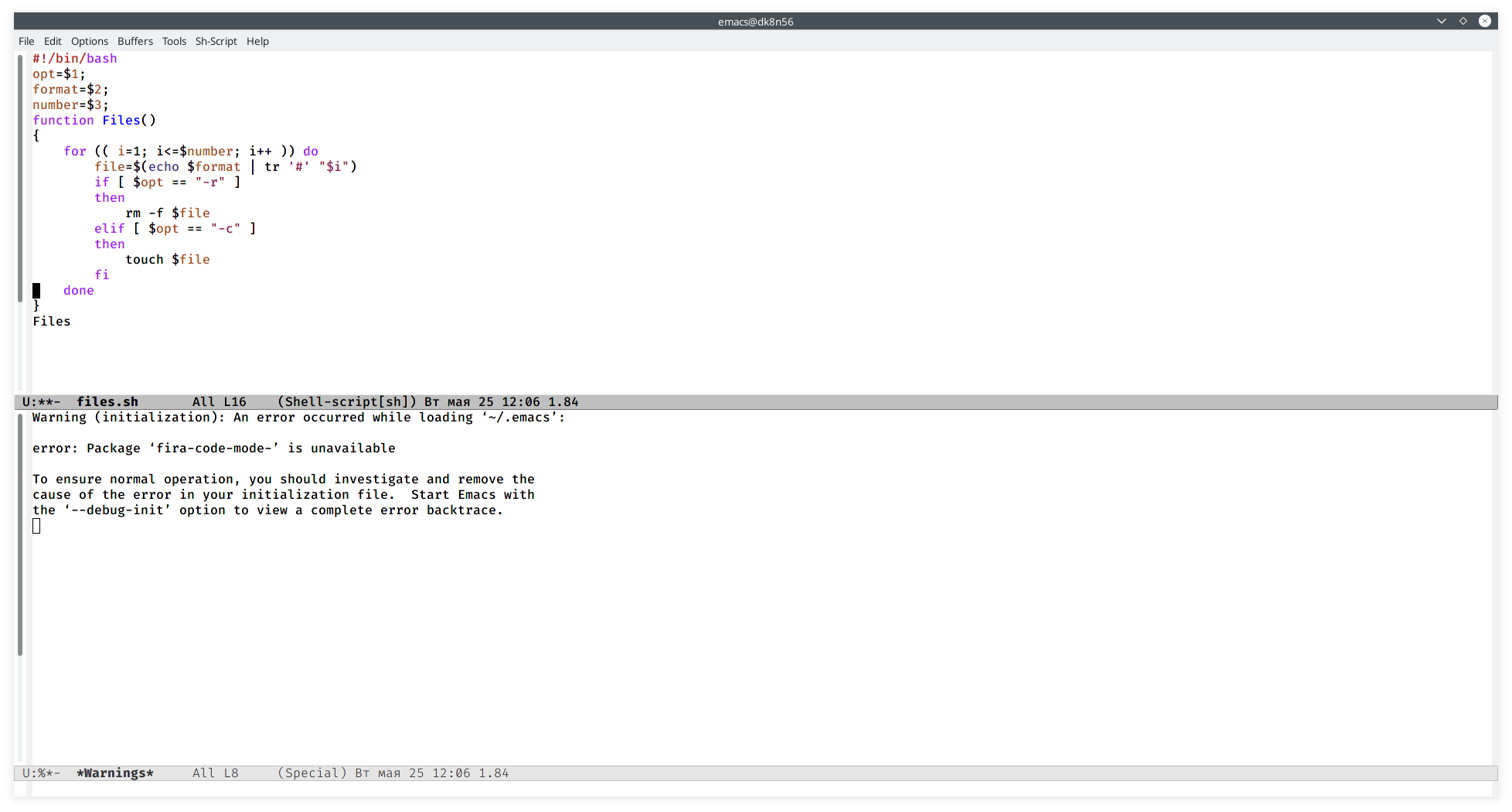


Figure 3.10: Скрипт №3

Далее я проверила работу написанного скрипта (команда «./files.sh»), пред- варительно добавив право на исполнение файла (команда «chmod+x files.sh»). Сначала я создала три файла (команда «./files.sh–cabc#.txt3»), удовлетворяющие

условию задачи, а потом удалила их (команда «./files.sh–rabc#.txt3») (Скриншот

[3.11](#_bookmark13) ).

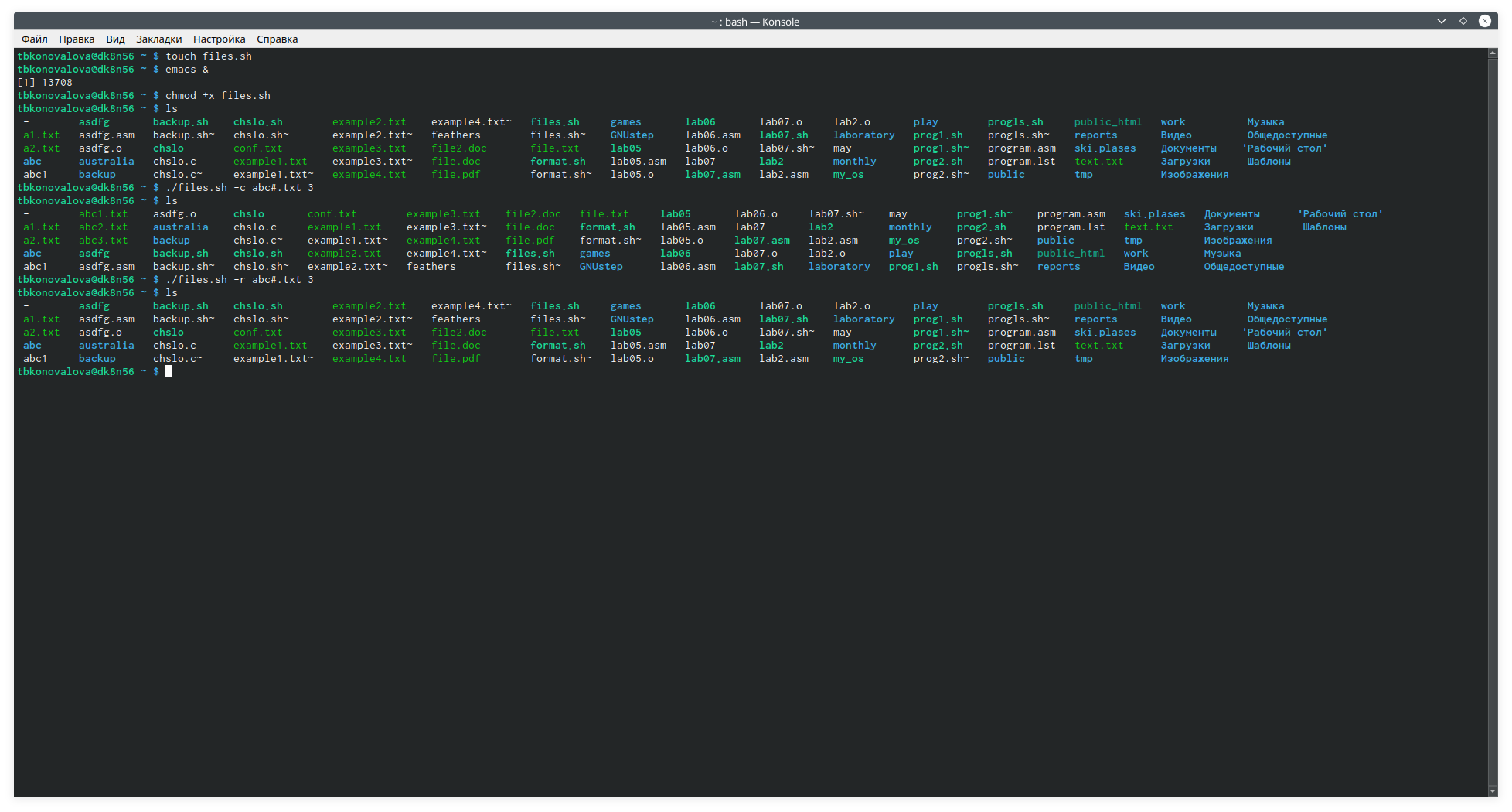


Figure 3.11: Проверка работы скрипта №3

* 1. Написала командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицировала его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду find). Для данной задачи я создала файл: prog4.sh (Скриншот [3.12](#_bookmark14) ) и написала соответствующий скрипт (См. рис. [3.13](#_bookmark15) ).

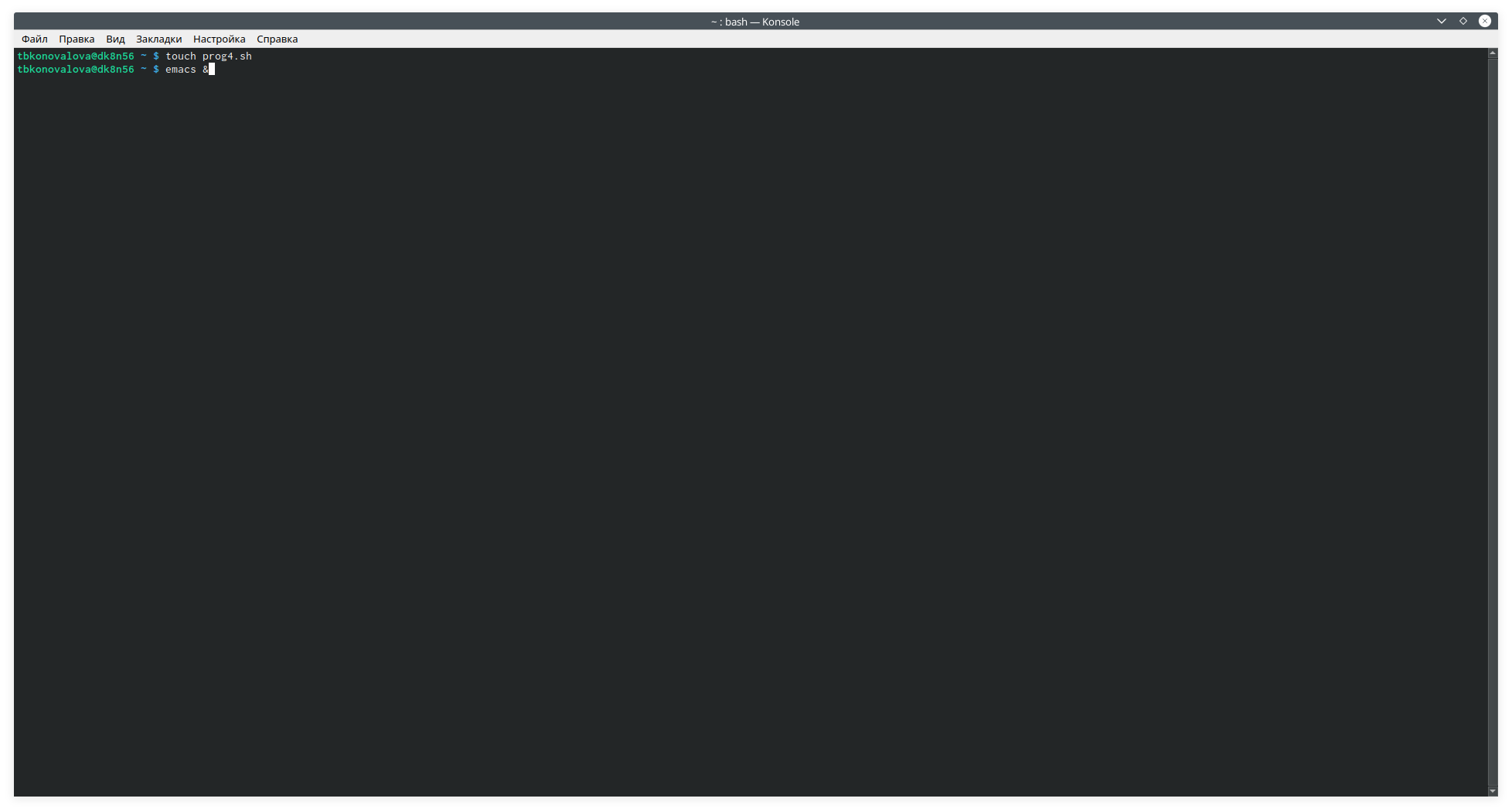


Figure 3.12: Создание файлов

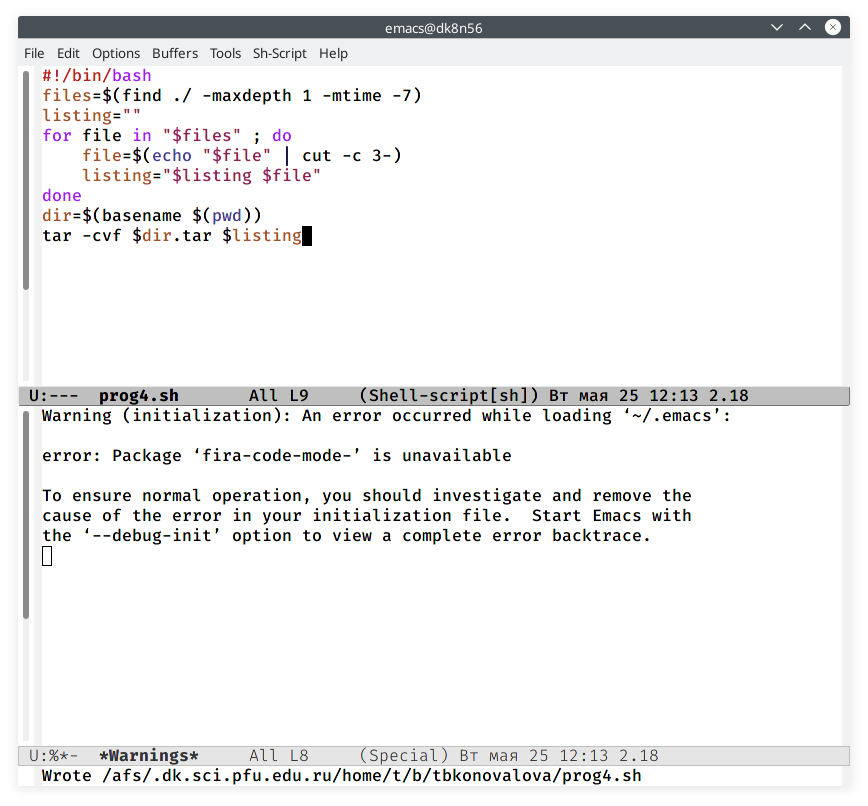


Figure 3.13: Скрипт №4

Далее я проверила работу написанного скрипта (команды «./prog4.sh» и «tar-tf Catalog1.tar»), предварительно добавив право на исполнение файла (команда

«chmod +x prog4.sh») и создав отдельный Catalog1 с несколькими файлами. Как видно из Рисунков [3.14](#_bookmark16) , файлы, измененные более недели назад, заархивированы не были. Скрипт работает корректно.

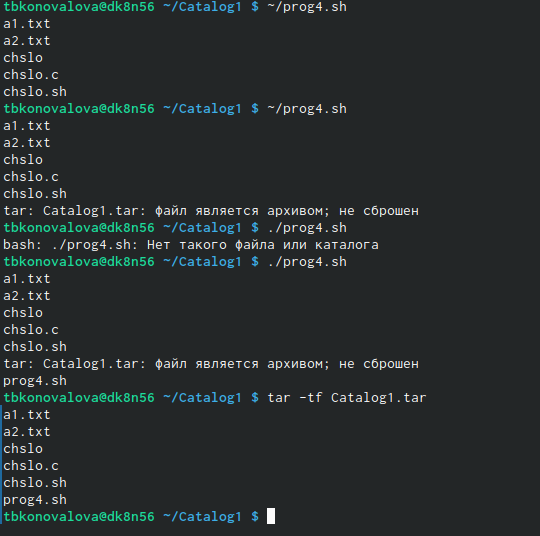


Figure 3.14: Проверка скрипта №4

Ответы на контрольные вопросы:

1. Команда getopts осуществляет синтаксический анализ командной строки, выделяя флаги, ииспользуется для объявления переменных. Синтаксис команды следующий: getopts option-string variable [arg…] Флаги − это опции командной строки, обычно помеченные знаком минус; Например,для команды ls флагом может являться -F. Строка опций option-string − эт осписок возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет со- провождаться некоторым аргументом, то за символом, обозначающим этот флаг, должно следовать двоеточие. Соответствующей переменной присваивается буква данной опции. Еслик оманда getopts может распознать аргумент, то она возвра- щает истину. Принято включать getopts в цикл while и анализировать введённые данные с помощью оператора case. Функция getopts включает две специальные переменные среды −OPTARG и OPTIND. Если ожидается доплнительное значе-

ние,то OPTARG устанавливается в значение этого аргумента. Функция getopts также понимает переменные типа массив, следовательно, можно использовать её в функции не только для синтаксического анализа аргументов функций, но и для анализа введённых пользователем данных.

1. Приперечислении имён файлов текущего каталога можно использовать сле- дующие символы: 1. *−соответствует произвольной, в том числе и пустой строке;*

*2. ?−соответствует любому одинарному символу; 3. [c1-c2] − соответствует любо-*

*му символу, лексикографически находящемуся между символами с1 и с2. Например,*

*1.1 echo* − выведет имена всех файлов текущего каталога, что представляет со- бой простейший аналог команды ls; 1.2. ls*.c−выведет все файлы с последними двумя символами, совпадающими с.c. 1.3. echoprog.?−выведет все файлы, состоя- щие из пяти или шести символов, первыми пятью символами которых являются prog.. 1.4.[a-z]*−соответствует произвольному имени файла в текущем каталоге, начинающемуся с любой строчной буквы латинского алфавита.

1. Часто бывает необходимо обеспечить проведение каких-либо действий цик- лически и управление дальнейшими действиями в зависимости от результатов проверки некоторого условия. Для решения подобных задач язык программиро- вания bash предоставляет возможность использовать такие управляющие кон- струкции, как for, case, if иwhile. С точки зрения командного процессора эти управ- ляющие конструкции являются обычными командами и могут использоваться как при создании командных файлов, так и при работе в интерактивном режиме. Команды,реализующие подобные конструкции, по сути, являются операторами языка программирования bash. Поэтому при описании языка программирова- ния bash термин оператор будет использоваться наравне с термином команда. Команды ОСUNIX возвращают код завершения, значение которого может быть использовано для принятия решения о дальнейших действиях. Команда test, например, создана специально для использования в командных файлах. Един- ственная функция этой команды заключается в выработке кода завершения.
2. Два несложных способа позволяют вам прерывать циклы в оболочке bash.

Команда break завершает выполнение цикла, а команда continue завершает дан- ную итерацию блока операторов. Команда break полезна для завершения цикла while в ситуациях, когда условие перестаёт быть правильным. Команда continue используется в ситуациях, когда больше нет необходимости выполнять блок опе- раторов, но вы можете захотеть продолжить проверять данный блок на других условных выражениях.

1. Следующие две команды ОСUNIX используются только совместно с управля- ющими конструкциями языка программирования bash: это команда true,которая всегда возвращает код завершения, равный нулю(т.е.истина),и команда false,ко- торая всегда возвращает код завершения,неравный нулю(т.е.ложь).Примеры бесконечных циклов:while true do echo hello andy done until false do echo hello mike done.
2. Строка if test-fman*𝑠/*i.*𝑠, 𝑚𝑎𝑛*s/*𝑖.*s и является ли этот файл обычным фай-

лом.Если данный файл является каталогом,то команда вернет нулевое значение (ложь).

1. Выполнение оператора цикла while сводится к тому,что сначала выполня- ется последовательность команд(операторов),которую задаёт список-команд в строке,содержащей служебное слово while,а затем,если последняя выполненная команда из этой последовательности команд возвращает нулевой код завер- шения(истина),выполняется последовательность команд(операторов),которую задаёт список-команд в строке,содержащей служебное слово do,после чего осу- ществляется безусловный переход на начало оператора цикла while.Выход из цикла будет осуществлён тогда,когда последняя выполненная команда из после- довательности команд (операторов),которую задаёт список-команд в строке,со- держащей служебное слово while, возвратит ненулевой код завершения(ложь). При замене в операторе цикла while служебного слова while на until условие,при выполнении которого осуществляется выход из цикла,меняется на противопо- ложное.В остальном оператор цикла while и оператор цикла until идентичны.

# Библиография

* 1. Программное обеспечение GNU/Linux. Лекция 3. FHS и процессы (Г. Курячий, МГУ);
  2. Программное обеспечение GNU/Linux. Лекция 4. Права доступа (Е. Алёхова, МГУ);
  3. Программное обеспечение GNU/Linux. Лекция 6. ПО не из хранилища дис- трибутива (Г. Курячий, МГУ)
  4. Электронный ресурс: [https://w](http://www.skleroznik.in.ua/2013/07/31/cikly-i-)ww[.skleroznik.in.ua/2013/07/31/cikly](http://www.skleroznik.in.ua/2013/07/31/cikly-i-)-i- vetvleniya/
  5. Электронный ресурс: ht[tps://w](http://www.opennet.ru/docs/RUS/bash_scripting_guide/c4875.html)ww[.opennet.ru/docs/RUS/bash\_scripting\_guide/c4875.html](http://www.opennet.ru/docs/RUS/bash_scripting_guide/c4875.html)

# Выводы

В ходе выполнения данной лабораторной работы я изучила основы программи- рования в оболочке ОС UNIX/Linuxи научилась писать небольшие командные файлы.