РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Лабораторная работа No 11. Программирование в командном процессоре ОС UNIX. Ветвления и циклы

Абдуллина Ляйсан Раисовна НПИбд-01-21

Содержание

# 1 Цель работы

Изучить основы программирования в оболочке ОС UNIX. Научится писать более сложные командные файлы с использованиемлогических управляющих конструкций и циклов.

# 2 Теоретическое введение

*Командный процессор* (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера.В операционных системахтипа UNIX/Linux наиболее часто используются следующие реализации командных оболочек: - оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux,содержащая базовый,но при этом полный набор функций; - С-оболочка (или csh)—надстройка на оболочкой Борна,использующая С-подобный синтаксис команд с возможностью сохранения истории выполнения команд; - оболочка Корна (или ksh)—напоминает оболочку С,но операторы управления программой совместимы с операторами оболочки Борна; - BASH — сокращение от Bourne Again Shell (опять оболочка Борна),в основе своей совмещает свойства оболочек С и Корна (разработка компании Free Software Foundation). POSIX (Portable Operating System Interface for Computer Environments)— набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ. Стандарты POSIX разработаны комитетом IEEE (Institute of Electrical and Electronics Engineers) для обеспечения совместимости различных UNIX/Linux-подобных операционных систем и переносимости прикладных программ на уровне исходного кода. POSIX-совместимые оболочки разработаны на базе оболочки Корна.

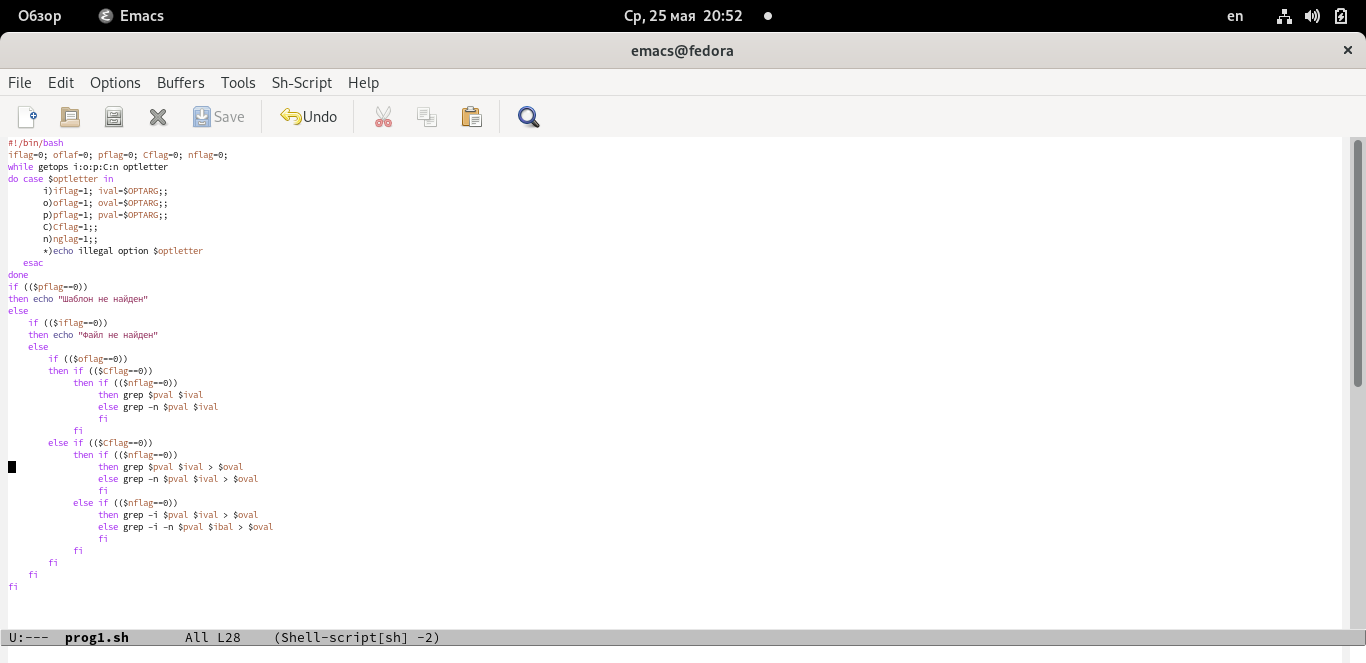
# 3 Выполнение лабораторной работы

##1

Используя команды getopts grep, напишем командный файл,который анализирует командную строку с ключами:

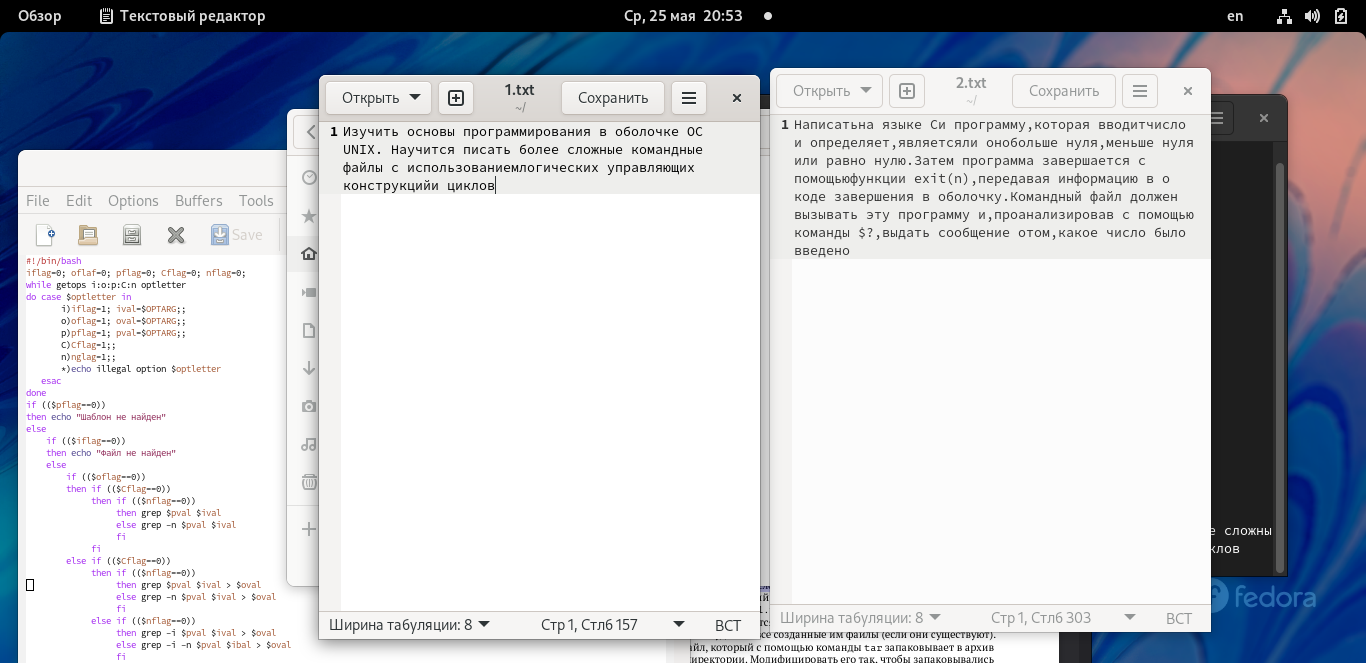
* -i inputfile —прочитатьданные из указанного файла;
* -o outputfile —вывести данные в указанный файл;
* -p шаблон —указать шаблон для поиска;
* -C —различать большие и малые буквы;
* -n —выдавать номера строк. а затем ищет в указанном файле нужные строки,определяемые ключом -p.

Для этого мы создаем файл prog1.sh, после чего открываем emacs в фоновом и режиме и начинаем работу. (скриншот 1)

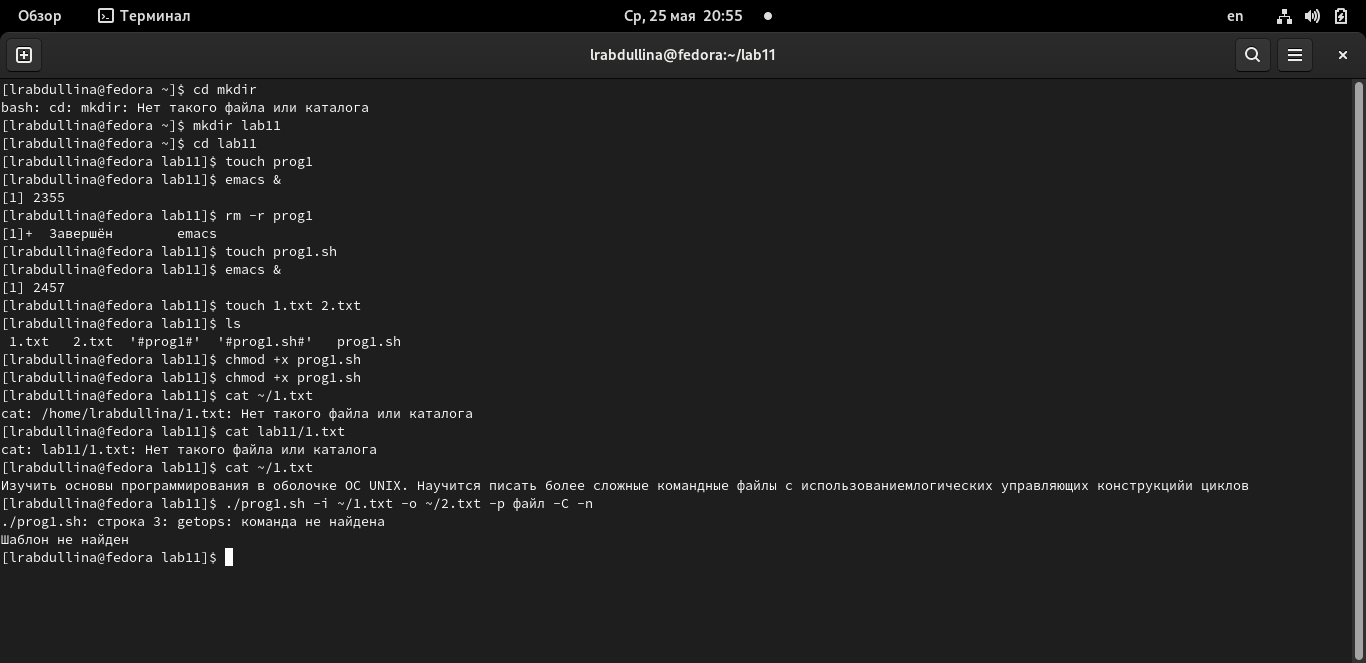


Скриншот 1: Написанная программа

Для проверки работы нашей программы предварительно создадим файлы 1.txt и 2.txt и внесем туда информацию. А также не забудем сделать наш файл с программой исполняемым через команду chmod +x prog1.sh. После этого вводим команды в консоли cat ~/1.txt и ./prog1.sh -i ~/1.txt -o ~2.txt -p “” -С -т и смотрим на результат. (скриншоты 2, 3)



Скриншот 2: Создание файлов для проверки программы

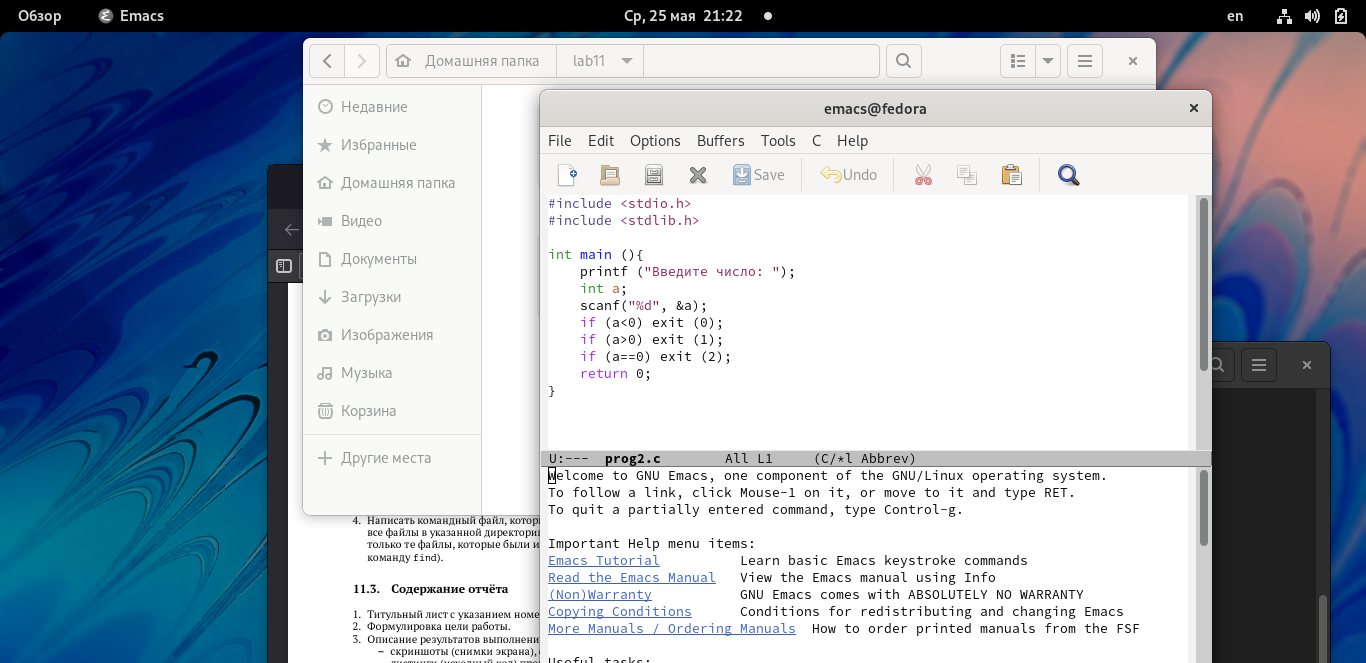


Скриншот 3: Проверка - работает успешно

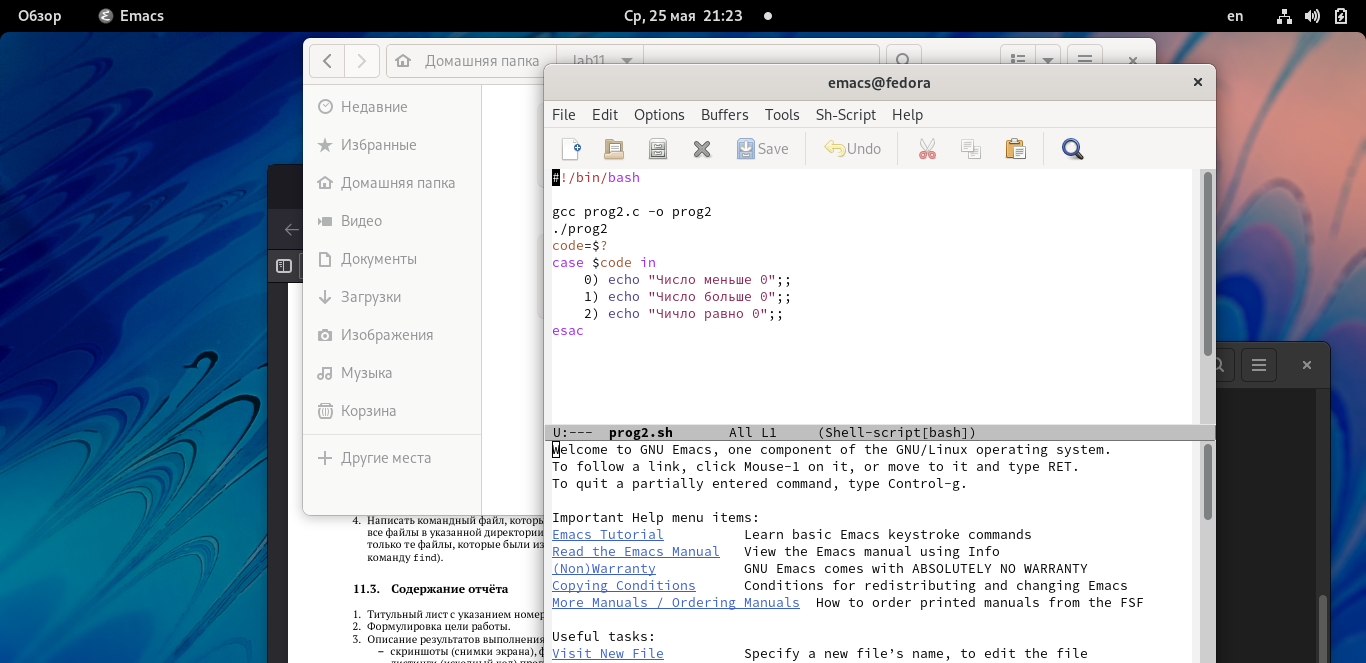
## 3.1 2

Напишем на языке Си программу,которая вводит число и определяет,является ли оно больше нуля,меньше нуля или равно нулю.Затем программа завершается с помощью функции exit(n),передавая информацию в о коде завершения в оболочку.Командный файл будет вызывать эту программу и,проанализировав с помощью команды $?,выдать сообщение отом,какое число было введено.

Для этого мы создаем 2 файла: prog2.c, prog2.sh. В них мы пишем код программы на Си и bash соответвенно. (скриншоты 4, 5)

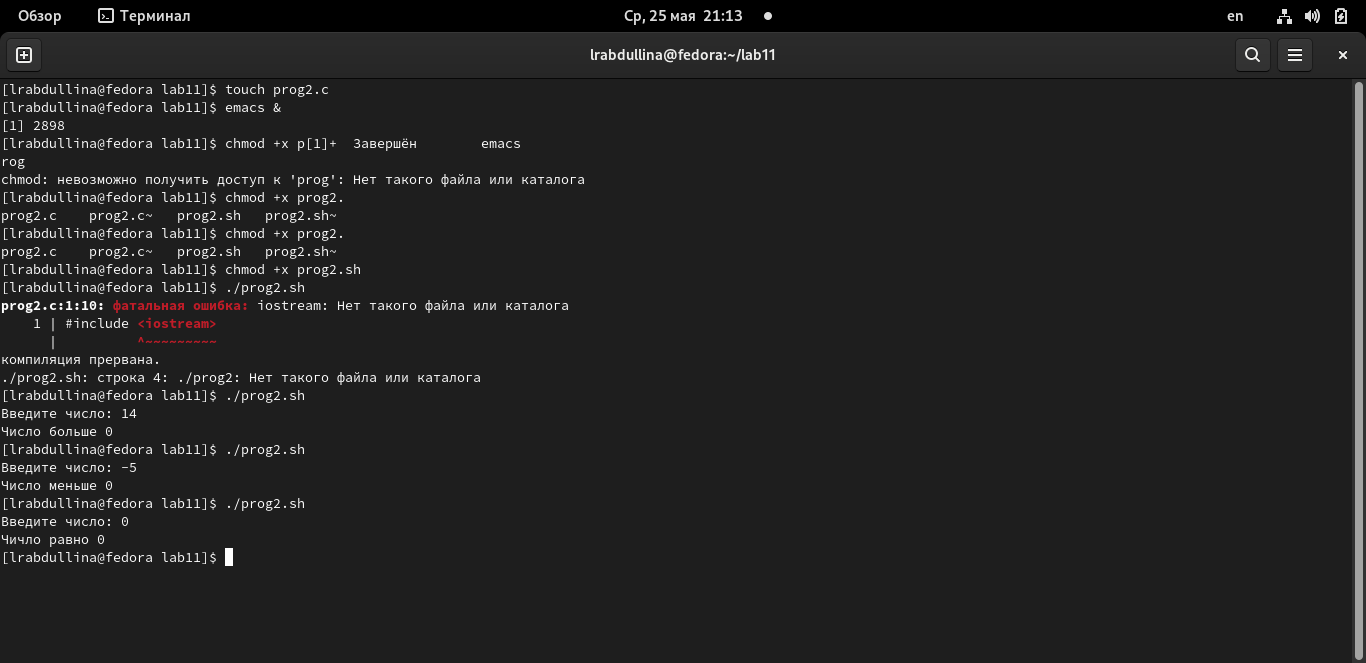


Скриншот 4: Написанная программа на Си



Скриншот 5: Написанная программа на bash

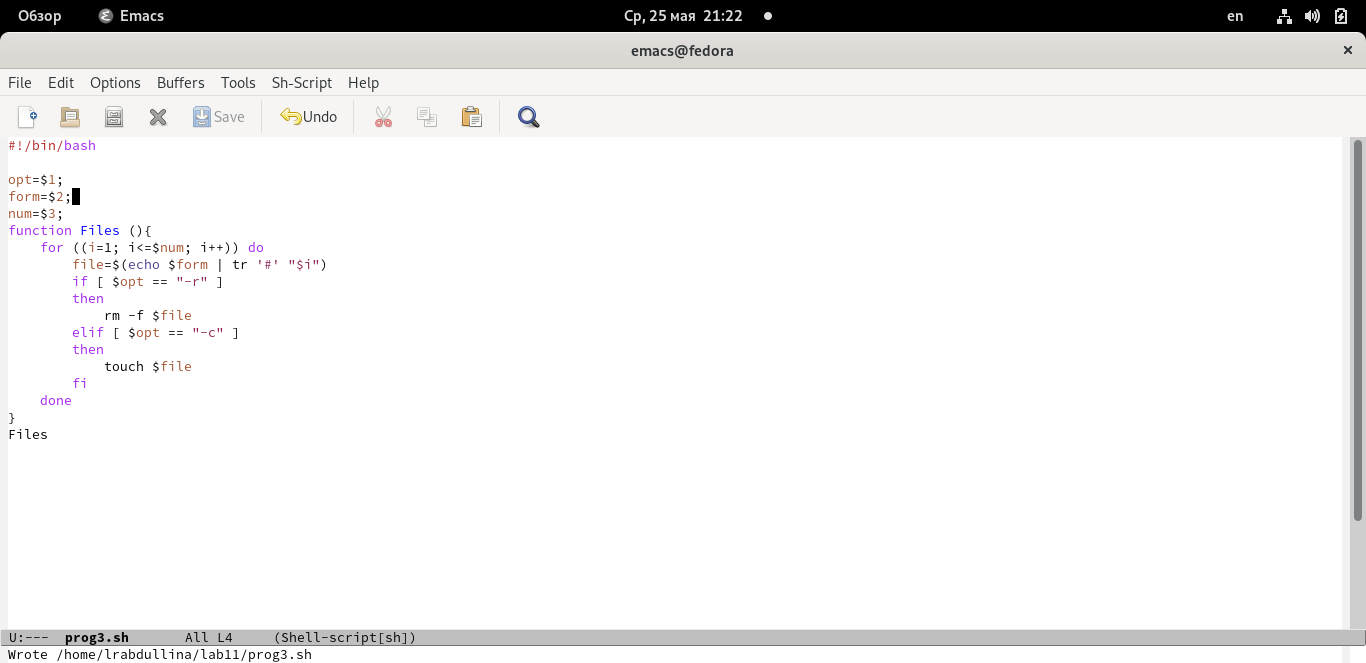
Сделаем файл исполняемым через команду chmod +x prog2.sh. И проверим его работу. Задаем случайные числа и смотрим, правильный ли ответ. (скриншот 6)



Скриншот 6: Проверка - работает успешно

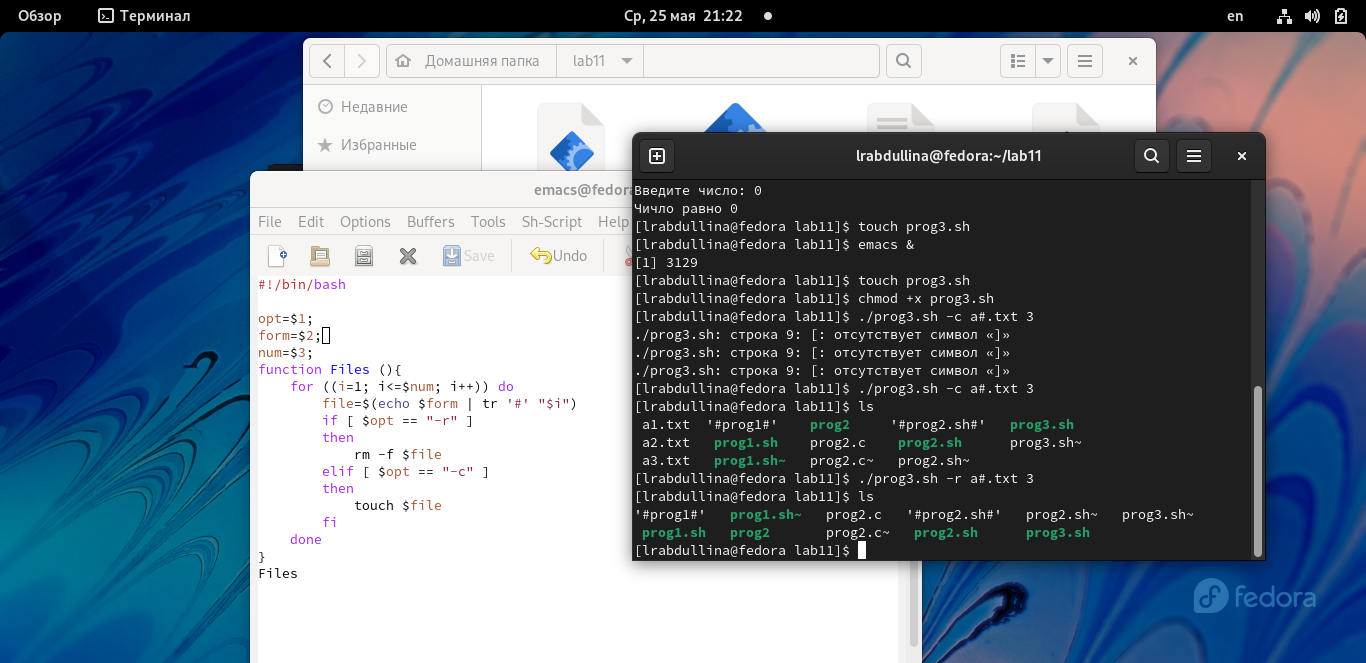
1. Напишем командный файл,создающий указанное число файлов,пронумерованных последовательно от 1 до N (например1.tmp,2.tmp,3.tmp,4.tmp ит.д.).Число файлов, которые необходимо создать,передаётся в аргументы командной строки.Этот же командный файл должен уметь удалять все созданные им файлы (если они существуют).

Для начала созданим новый файл prog3.sh и напишем программу. (скриншот 7)



Скриншот 7: Написанная программа

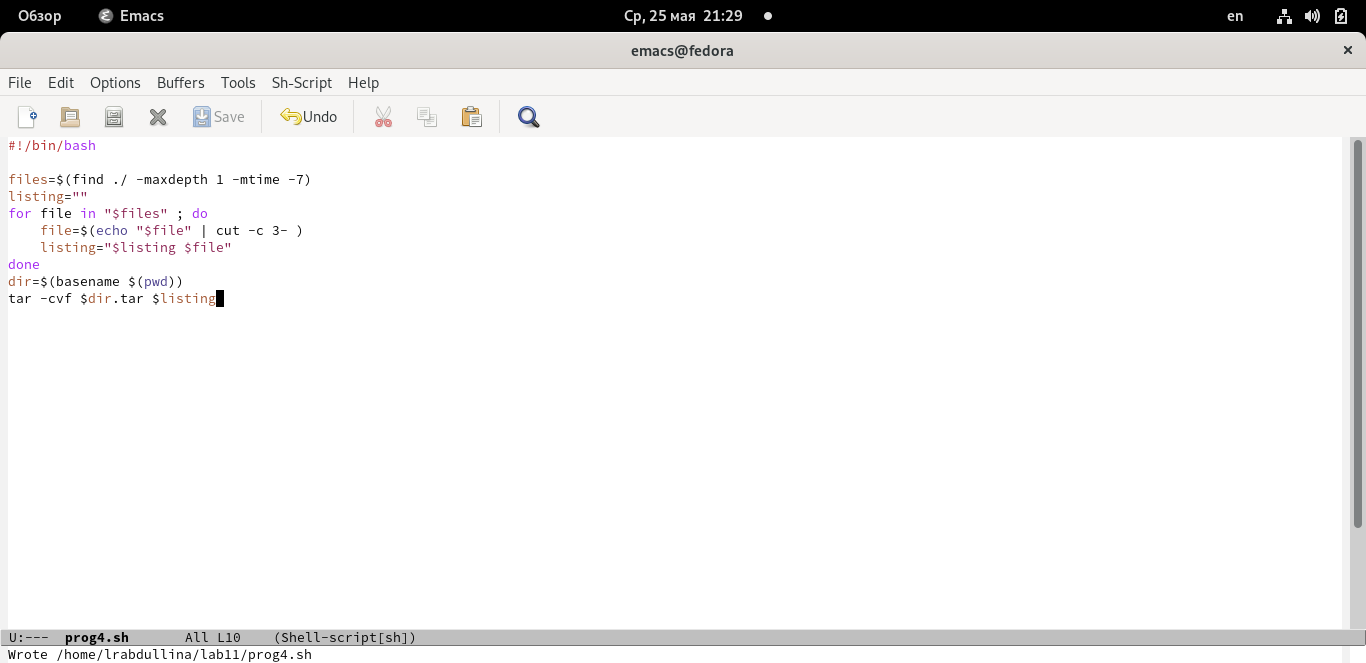
Сделаем файл исполняемым через команду chmod +x prog3.sh. И проверим его работу. Создадим через команду ./prog3.sh -c a#.txt 3 3 новых файла. А затем через команду ./prog3.sh -r a#.txt 3 удалим их (скриншот 8)



Скриншот 8: Проверка - работает успешно

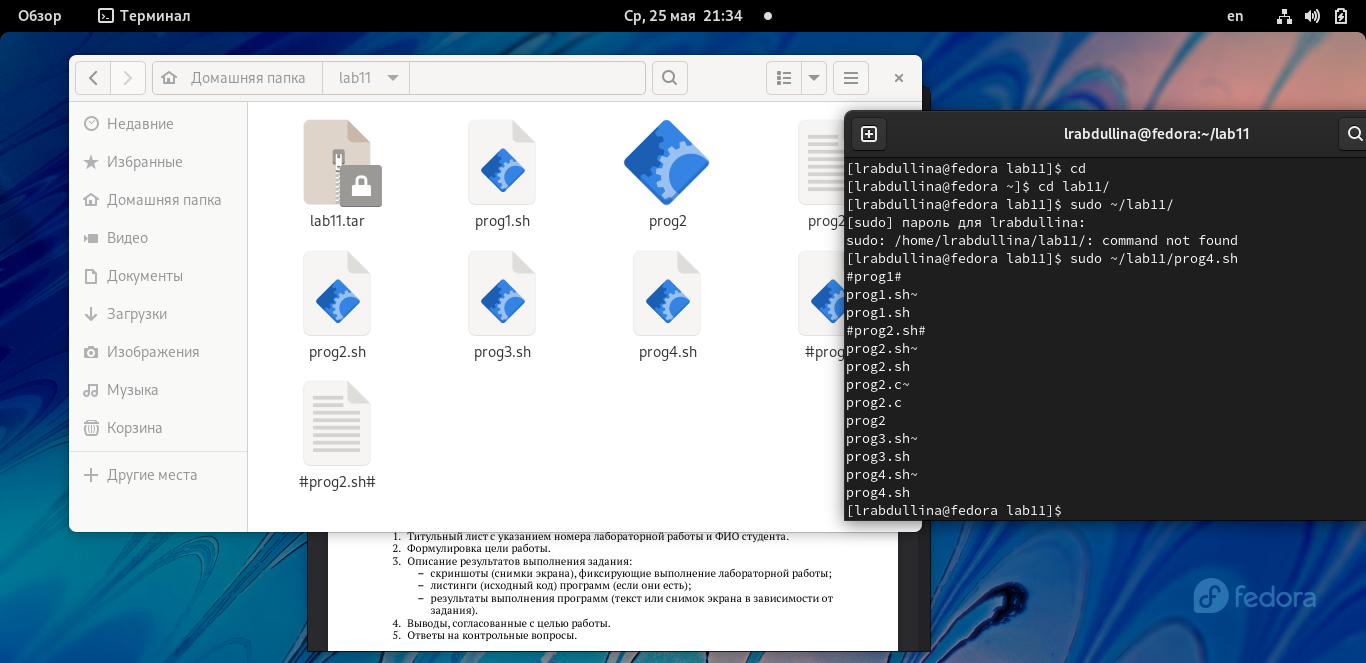
1. Напишем командный файл,который с помощью команды tar запаковываетв архив все файлы в указанной директории.Модифицируем его так,чтобы запаковывались только те файлы,которые были изменены менее недели тому назад (использовать команду find).

Для начала созданим новый файл prog4.sh и напишем программу. (скриншот 9)



Скриншот 9: Написанная программа

Вводим команду sudo ~/lab11/prog4.sh и смотрим на результат.



Скриншот 10: Проверка - работает успешно

# 4 Контрольные вопросы

1. Каково предназначение команды getopts?

Команда getopts осуществляет синтаксический анализ командной строки, выделяя флаги, и используется 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 − это список возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет сопровождаться некоторым аргументом, то за символом, обозначающим этот флаг, должно следовать двоеточие. Соответствующей переменной присваивается буква данной опции. Если команда getopts может распознать аргумент, то она возвращает 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 дополнительное значение, 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. Какое отношение метасимволы имеют к генерации имён файлов?

При перечислении имён файлов текущего каталога можно использовать следующие символы:

* − - соответствует произвольной, в том числе и пустой строке;
* ? − соответствует любому одинарному символу;
* [c1-c2] − соответствует любому символу, лексикографически находящемуся между символами с1 и с2. Например,
* echo \* − выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды ls;
* ls \*.c − выведет все файлы с последними двумя символами, совпадающими с .c.
* echo prog.? − выведет все файлы, состоящие из пяти или шести символов, первыми пятью символами которых являются prog..
* [a-z]\* − соответствует произвольному имени файла в текущем каталоге, начинающемуся с любой строчной буквы латинского алфавита.

1. Какие операторы управления действиями вы знаете?

Часто бывает необходимо обеспечить проведение каких-либо действий циклически и управление дальнейшими действиями в зависимости отрезультатов проверки некоторого условия. Для решения подобных задач язык программирования bash предоставляет возможность использовать такие управляющие конструкции, как for, case, if и while. С точки зрения командного процессора эти управляющие 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 программирования bash термин оператор будет использоваться наравне с термином команда. Команды ОС UNIX возвращают код завершения, значение которого может быть использовано для принятия решения о дальнейших действиях. Команда test, например, создана специально для использования в командных файлах. Единственная функция этой команды заключается в выработке кода завершения.

1. Какие операторы используются для прерывания цикла?

Два несложных способа позволяют вам прерывать циклы в оболочке bash. Команда break завершает выполнение цикла, а команда continue завершает данную 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 операторов, но вы можете захотеть продолжить проверять данный блок на других условных выражениях.

1. Для чего нужны команды false и true?

Следующие две команды ОС UNIX используются только совместно с управляющими конструкциями языка программирования bash: это команда true, которая всегда возвращает код завершения, равный нулю (т.е. истина), и команда false, которая всегда возвращает код завершения, не равный нулю (т. е. ложь). Примеры бесконечных циклов: while true do echo hello andy done until false do echo hello mike done

1. Что означает строка if test -f mani.$s,встреченная в командном файле?

Строка if test -f mans/i.sпроверяет,существуетлифайлmans/i.s и является ли этот файл обычным файлом. Если данный файл является каталогом, то команда вернет нулевое значение (ложь).

1. Объясните различия между конструкциями while и until

Выполнение оператора цикла while сводится к тому, что сначала выполняется 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 завершения (истина), выполняется последовательность команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово do, после чего осуществляется безусловный переход на начало оператора цикла while. Выход из цикла будет осуществлён тогда, когда последняя выполненная команда из последовательности команд (операторов), которую задаёт список-команд в строке, содержащей 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 противоположное. В остальном оператор цикла while и оператор цикла until идентичны.

# 5 Выводы

В ходе лабораторной работы мы изучили основы программирования в оболочке ОС UNIX. Научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

# 6 Список литературы

https://esystem.rudn.ru/course/view.php?id=5790 :::