**Содержание**

Введение………………………………………………………………………………... 4

1. Создание файлов……………………………………………………………………….. 5
2. Grep……………………………………………………………………………………… 6
   1. –i……………………………………………………………………………………. 7
   2. –с……………………………………………………………………………………. 7
   3. –n…………………………………………………………………………………… 8
   4. –v………………………………………………………………………….………… 8
   5. –w…………………………………………………………………………………… 9
   6. –x……………………………………………………………………………………. 9
   7. –l…………………………………………………………………………………... 10
   8. –L…………………………………………………………………………………. 10
3. Ack……………………………………………………………………………………... 12
   1. –f………………………………………………………………………………….. 13
   2. –w…………………………………………………………………………………. 14
   3. –ch………………………………………………………………………………… 14
   4. –c………………………………………………………………………………….. 14
4. Awk……………………………………………………………………………………..16
   1. –F………………………………………………………………………………….. 17
   2. –v………………………………………………………………………………….. 18
   3. –f…………………………………………………………………………………... 18
   4. –length…………………………………………………………………………….. 19
   5. –BEGIN/END…………………………………………………………………….. 19
5. Sort……………………………………………………………………………………... 20
   1. –r…………………………………………………………………………………... 20
   2. –k………………………………………………………………………………….. 21
   3. –n………………………………………………………………………………….. 21
   4. –m…………………………………………………………………………………. 21
   5. –u………………………………………………………………………………….. 22
6. Uniq……………………………………………………………………………………. 24
   1. –c………………………………………………………………………………….. 25
   2. –d…………………………………………………………………………………. 25
   3. –u…………………………………………………………………………………. 26
   4. –D…………………………………….…………………………………………… 26
7. Wс……………………………………………………………………………………… 28
   1. –l…………………………………………………………………………………... 28
   2. –w…………………………………………………………………………………. 28
   3. –c………………………………………………………………………………….. 29
8. Tr………………………………………………………………………………………. 30
   1. -s…………………………………………………………………………………... 30
   2. -d………………………………………………………………………………….. 30
   3. -c/cd……………………………………………………………………………….. 31

Заключение………………………………………………………………………………33 Список используемой литературы…………………………………………………….. 34  **Введение**

**Цель работы** – познакомиться с оболочкой bash и утилитами, используемыми it – специалистом в повседневной жизни.

В данной курсовой работе мы познакомимся с фильтрами, которые позволяют строить сложные конвейеры для манипуляций с текстом. Мы узнаем, как выводить текст, сортировать его, выполнять подсчет слов и строк, преобразовывать символы, а также многое другое.

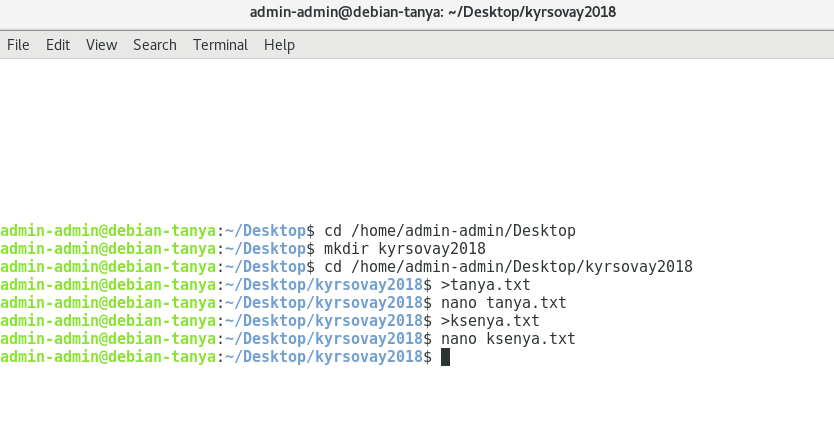
Фильтрация текста – это процесс получения входного текстового потока, выполнения неких преобразований над ним и передача измененных данных в выходной поток. Хотя входные или выходные данные могут поступать из файлов, в UNIX® и Linux фильтрация обычно осуществляется путем составления конвейеров из команд, в которых вывод одной команды передается по программному каналу (или перенаправляется) на вход следующей команде.

В рамках данной курсовой работы представлена информация о наиболее часто используемых фильтрах. В результате комбинирования простых команд и фильтров с использованием программных каналов могут быть созданы элегантные решения.

К числу команд-фильтров относятся grep, ack, awk, sort, uniq, ws, tr. Далее рассмотрим их более подробно.

**1.Создание файлов**

1. Переходим в директорию рабочего стола
2. Создаем там папку kyrsovay2018
3. Переходим в директорию этой папки
4. Создаем файл tanya.txt в данной директории
5. Заносим данные в файл
6. Создаем файл ksenya.txt в данной директории
7. Заносим данные в файл



В курсовой работе используются не только два файла. Эти файлы приведены в качестве примера.

**2.Grep**

Команда grep, одна из самых известных и употребительных команд Юниксовидных ОС, ведет свое начало от первого текстового редактора Юникс - ed. В этом редакторе была команда g/re/p (global/regular expression/print), которая и дала свое название новой программе.

Утилита grep осуществляет во входных данных поиск строк, совпадающих с данным выражением (данными выражениями). Если строка ввода совпадает с любым из выражений, она считается "выбранной". По умолчанию выбранные строки записываются в стандартный поток вывода.

Утилита grep распознает три типа регулярных выражений: базовые, расширенные и фиксированные.

Базовые и расширенные регулярные выражения аналогичны арифметическим выражениям, в которых крупные выражения формируются путем объединения более мелких выражений и операторов по установленному правилу приоритетов.

В регулярных выражениях присутствует "невидимый" оператор, т.е. конкатенация. Конкатенация двух выражений означает сопоставление лево, а затем правой части.

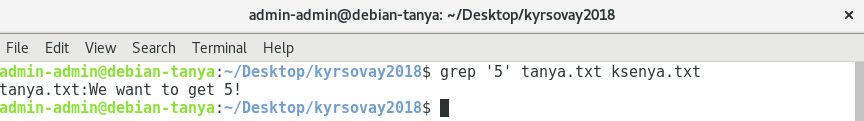
Наименьшим выражением является один символ.

***Задача:*** Команда grep может обрабатывать любое количество файлов одновременно. Возьмем два исходных файла, которые мы создали.

***Команда:***

grep ‘5’ tanya.txt ksenya.txt

***Результат:***



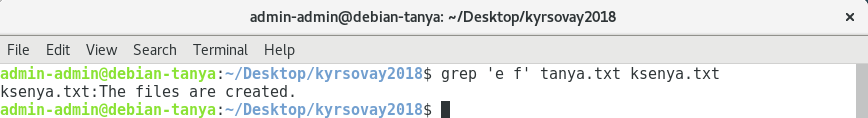
В выводе перечислены файлы, и указано, в каком из них какая строка содержит символ «5». Символ пришлось взять в кавычки, чтобы командный интерпретатор понял, что имеется в виду символ, а не условный знак.

Команда grep вовсе не ограничена одним выражением, можно задавать хоть целые фразы. Только их тоже нужно заключать в кавычки (одинарные или двойные):

***Команда:***

grep ‘e f ’ tanya.txt ksenya.txt

***Результат:***



***Опции grep***

*1.Опция –i*

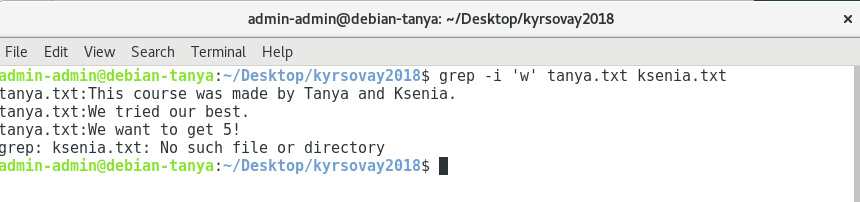
--ignore-case

***Задача:*** Приказывает команде игнорировать регистр символов, таким образом, поиск будет производиться как среди заглавных, так и среди строчных букв.

***Команда:***

grep –i ‘w’ tanya.txt ksenya.txt

***Результат:***



Изначально была задана строчная буква ‘w’, а на экран произошел вывод как строчных, так и заглавных букв ‘w’

*2. Опция –с*

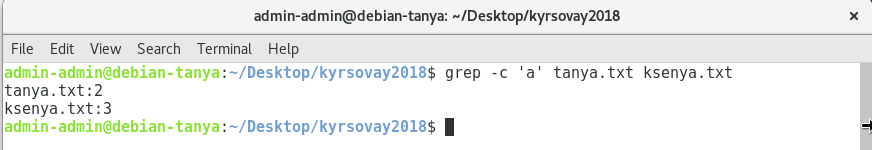
--count

***Задача:*** Эта опция не выводит строки, а подсчитывает количество строк, в которых обнаружена заданная конфигурация символов.

***Команда:***

grep –с ‘a’ tanya.txt ksenya.txt

***Результат:***



То есть в двух строках файла tanya.txt и трех строках файла ksenya.txt встречается сочетание символа «а».

*3. Опция –n*

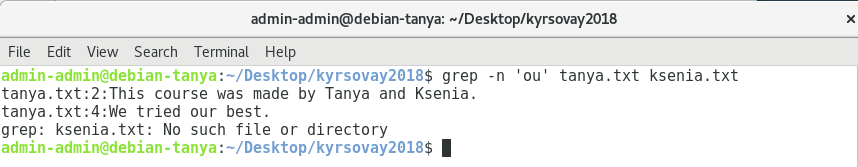
--line-number

***Задача:*** При использовании этой опции вывод команды grep будет указывать номера строк, содержащих заданную конфигурацию.

***Команда:***

grep –n ‘ou’ tanya.txt ksenya.txt

***Результат:***



Во второй и четвертой строках файла tanya.txt найдено сочетание ‘ou’

*4. Опция –v*

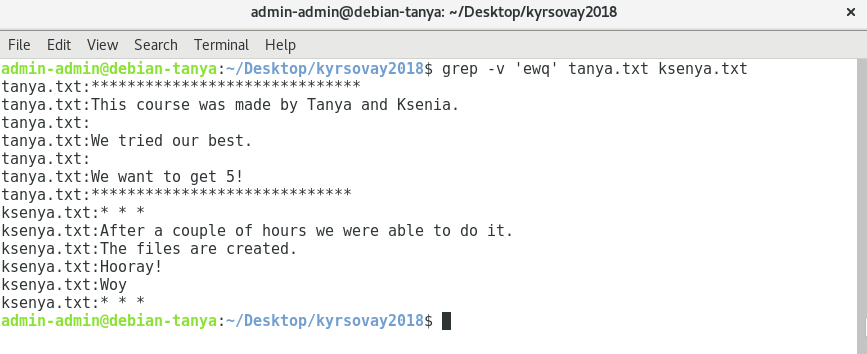
--invert-match

***Задача:*** Выполняет работу, обратную обычной - выводит строки, в которых заданная конфигурация не встречается.

***Команда:***

Grep –v ‘ewq’ tanya.txt ksenya.txt

***Результат:***

**

Во всех строках, которые выведены на экран, нет конфигурации символов ‘ewq’.

*5.Опция –w*

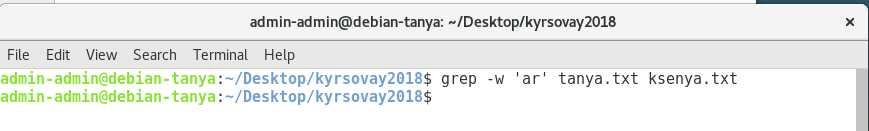
--word-regexp

***Задача:*** Заставит команду grep искать только строки, содержащие все слово или фразу, составляющую данную конфигурацию.

***Команда:***

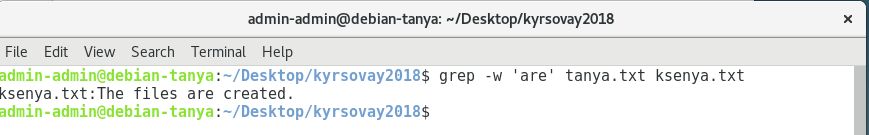
Grep –w ‘ar’ tanya.txt ksenya.txt

***Результат:***



Вывода нет, то есть не нашлись строки, содержащих выражение "ar".

А вот в данном случае:



Команда находит точное соответствие в файле ksenia.txt.

*6.Опция –x*

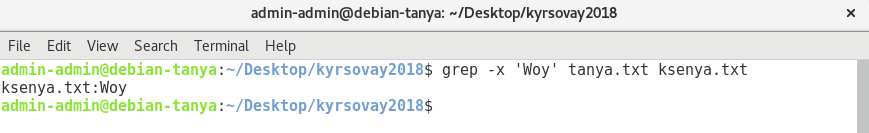
--line-regexp

***Задача:*** Еще более строгая команда. Она отберет только те строки исследуемого файла или файлов, которые полностью совпадают с заданной конфигурацией.

***Команда:***

Grep –x ‘Woy’ tanya.txt ksenya.txt

***Результат:***



Строгое совпадение нашлось только в файле ksenya.txt. Это строка ‘Woy’

*7.Опция –l*

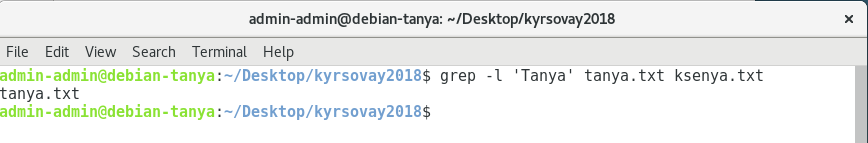
--files-with-matches

***Задача:*** Команда grep с этой опцией не возвращает строки, содержащие заданную конфигурацию, а сообщает лишь имена файлов, в которых данный образец найден.

***Команда:***

Grep –l ‘Tanya’ tanya.txt ksenya.txt

***Результат:***



Конфигурация символов ‘Tanya’ была найдена только в файле tanya.txt .

Стоит отметить, что сканирование каждого из заданных файлов продолжается только до первого совпадения с заданной конфигурацией.

*8.Опция –L*

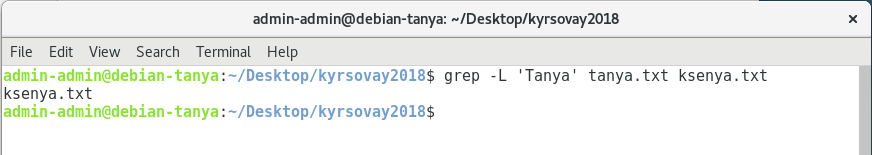
--files-without-match

***Задача:*** Эта команда наоборот, сообщает имена тех файлов, где не встретилась заданная конфигурация.

***Команда:***

Grep – L ‘Tanya’ tanya.txt ksenya.txt

***Результат:***

**

Конфигурация символов ‘Tanya’ была не найдена только в файле ksenya.txt

Команда grep настолько полезна, многофункциональна и проста в употреблении, что, однажды познакомившись с ней, невозможно представить себе работы без нее.

**3.Ack**

Инструмент ack был разработан специально для поиска исходного кода программ. Потому ack ищет определенные файлы и игнорирует другие.

К примеру, при поиске структуры каталогов проекта иерархия репозитория системы управления версиями, как правило, не нужна. Она содержит информацию о более старых версиях файла и, скорее всего, приведет к многочисленным повторам. Ack понимает, что здесь искать не нужно, поэтому он игнорирует эти каталоги, благодаря чему результат получается более сжатым и содержит меньше ошибочных данных.  
Подобным образом он игнорирует общие резервные файлы, созданные некоторыми текстовыми редакторами. Также нет смысла искать с помощью ack некоторые файлы, которые обычно встречаются в исходных каталогах: сжатые версии веб-файлов, изображения, PDF-файлы и т.д. Все эти функции положительно влияют на результаты почти каждого поиска. Кроме того, эти настройки всегда можно изменить.

Для начала нужно установить инструмент ack на сервер. На сервере Ubuntu или Debian это так же просто, как установить утилиту из репозитория по умолчанию. Пакет называется ack-grep:

# apt-get update # sudo apt-get install ack-grep

Поскольку исполняемый файл также называется ack-grep, можно сказать системе сократить его имя до ack, чтобы использовать его в командной строке, введя следующую команду:

# dpkg-divert --local --divert /usr/bin/ack --rename --add /usr/bin/ack-grep

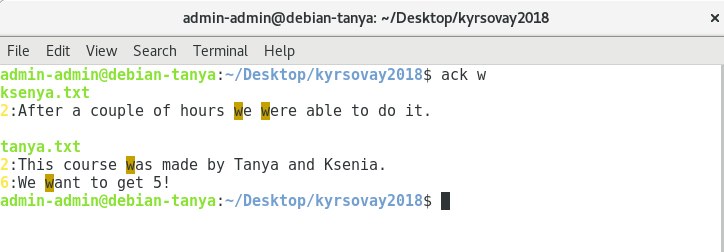
Теперь инструмент будет отвечать на имя ack вместо ack-grep.

***Задача:*** Вывести файлы в которых был найден указанный шаблон.

***Команда:***

ack w

***Результат:***



Данная команда нашла все файлы, в которых встречается шаблон «w».

**Опции ack.**

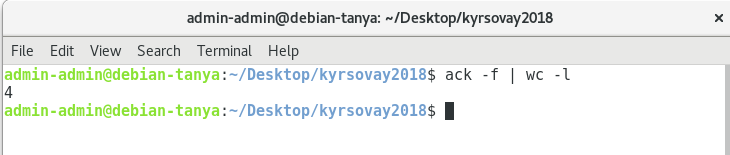
*1.Опция-f*

***Задача:*** Узнать количество файлов в директории, в которой мы сейчас находимся.

***Команда:***

ack –f | wc -l

***Результат:***



В данной директории было найдено 4 файла.

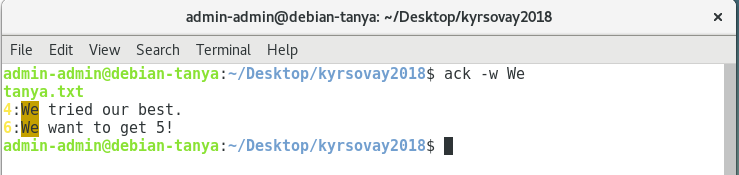
*2. Опция-w*

***Задача:*** Найти файлы, в которых слова четко совпадают с границами шаблона.

***Команда:***

ack –w We

***Результат:***



Точное совпадение нашлось только в файле tanya.txt.

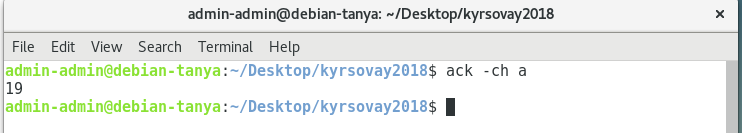
*3.Опция-ch*

***Задача:*** Узнать общее количество найденных совпадений с шаблоном в файлах.

***Команда:***

ack –ch a

***Результат:***



В файлах, данной директории, нашлось 19 совпадений указанного шаблона “a”.

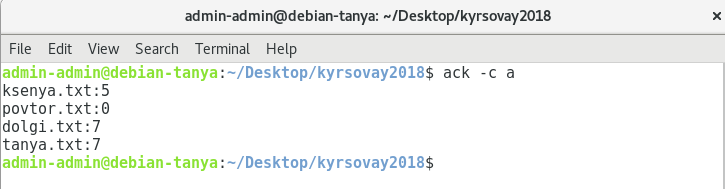
*4.Опция-с*

***Задача:*** Узнать количество совпадений с шаблоном в каждом файле.

***Команда:***

ack –c a

***Результат:***



Шаблон “а” был встречен в файле ksenya.txt 5 раз, в файле povtor.txt 0 раз , в файлах dolgi.txt и tanya.txt по 7 раз.

Как видите, ack — очень мощный и гибкий инструмент для работы с исходными кодами, который будет полезен даже при простом поиске файлов в среде Linux.

**4.Awk**

Утилита awk, или точнее GNU awk, в сравнении с sed, выводит обработку потоков данных на более высокий уровень. Благодаря awk в нашем распоряжении оказывается язык программирования, а не довольно скромный набор команд, отдаваемых редактору. С помощью языка

программирования awk можно выполнять следующие действия:

-объявлять переменные для хранения данных;

-использовать арифметические и строковые операторы для работы с данными;

-использовать структурные элементы и управляющие конструкции языка, такие, как оператор if-then и циклы, что позволяет реализовать сложные алгоритмы обработки данных;

-создавать форматированные отчёты.

Если говорить лишь о возможности создавать форматированные отчёты, которые удобно читать и анализировать, то это оказывается очень кстати при работе с лог-файлами, которые могут содержать миллионы записей. Но awk — это намного больше, чем средство подготовки отчётов.

Схема вызова awk выглядит так:

awk options program file

Awk воспринимает поступающие к нему данные в виде набора записей. Записи представляют собой наборы полей. Упрощенно, если не учитывать возможности настройки awk и говорить о некоем вполне обычном тексте, строки которого разделены символами перевода строки, запись — это строка. Поле — это слово в строке.

Одна из основных функций awk заключается в возможности манипулировать данными в текстовых файлах. Делается это путём автоматического назначения переменной каждому элементу в строке. По умолчанию awk назначает следующие переменные каждому полю данных, обнаруженному им в записи:

$0 — представляет всю строку текста (запись).

$1 — первое поле.

$2 — второе поле.

$n — n-ное поле.

Поля выделяются из текста с использованием символа-разделителя. По умолчанию — это пробельные символы вроде пробела или символа табуляции.

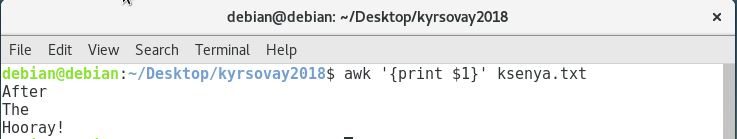
Рассмотрим использование этих переменных на простом примере. А именно, обработаем файл, в котором содержится несколько строк.

***Задача:*** Вывести только первый столбец в указанном файле. Где разделителем столбцов по умолчанию считается пробел.

***Команда:***

awk ‘{print $1}’ ksenya.txt

***Результат:***



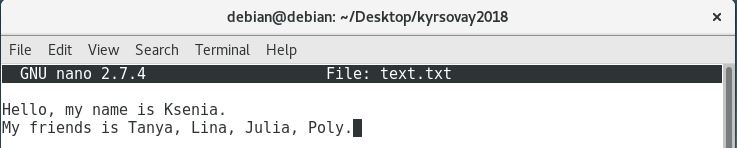
Данная команда вывела первые слова (первые поля) каждой записи, содержащиеся в файле ksenya.txt.

***Опции awk***

*1.Опция –F.*

***Задача:*** Вывести поля, в которых используется указанный в команде разделитель.

Текст файла text.txt



В качестве разделителя зададим символ ‘ ,’.

***Команда:***

awk –F, ‘{print $1}’ text.txt

***Результат:***



Данная команда вывела первые поля файла text.txt, считая в качестве символа разделителя (конца поля) символ “ , ”.

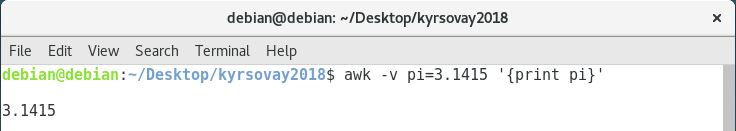
*2.Опция –v*

***Задача:*** Передать в программу переменную, а именно константу.

***Команда:***

awk –v pi=3.1415 ‘{print pi}’

***Результат:***

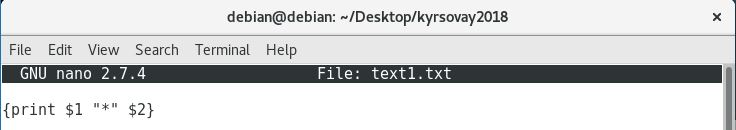


В результате работы команды было выведено число, занесённое нами в константу.

*3. Опция-f*

***Задача:*** Использовать скрипты хранящиеся в файлах, и обратиться к ним в команде.

Подготовим файл text1.txt, в который запишем следующее:

**

Вызовем awk, указав этот файл в качестве источника команд.

***Команда:***

awk –F, -f text1.txt text.txt

***Результат:***



В качестве разделителя строк мы указали символ ‘,’. Поэтому в ходе выполнения команды, между первым и вторым полем разделитель поменялся на символ ‘\*’, который мы указали в файле источника команд text1.txt.

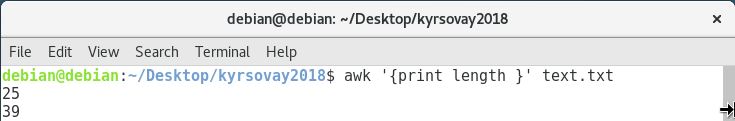
*4. length*

***Задача:*** Получить длину каждой строки, использую ключ length.

***Команда:***

awk ‘{print length}’ text.txt

***Результат:***

****

В файле text.txt, в первом предложении количество символов 25, во втором 39.

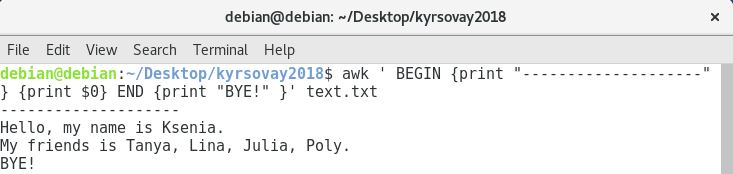
*5. BEGIN/END*

***Задача:*** Вывести перед файлом и после него указанные данные.

***Команда:***

awk ‘BEGIN {print “--------------------”} {print $0} END {print “BYE!”} text.txt

***Результат:***

****

В результате выполнения данной команды, в начало и конец файла text.txt, который мы выводим полностью (указав $0), будут добавлены строки указанные в команде.

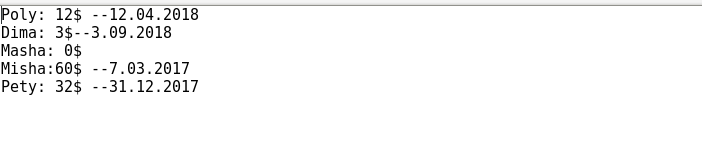
Awk это мощнейший инструмент обработки данных, масштабы которого сопоставимы с отдельным языком программирования.

**5.Sort**

Команда sort сортирует содержимое файла в алфавитном или нумерологическом порядке. Если задать несколько файлов, то команда sort соединит их и, рассортировав, выдаст единым выводом. По умолчанию, объектом сортировки будут строки, однако опции позволяют выбирать объект сортировки: колонки, столбцы и прочие элементы форматирования файла. Разделителем между ними служат пробелы, однако соответствующие опции позволяют задать иные разделители.

Команда sort весьма древняя, она может служить образцом программирования утилит в ранних 70-х годах прошлого века.

Составим список долгов и запишем их в файл dolgi.txt

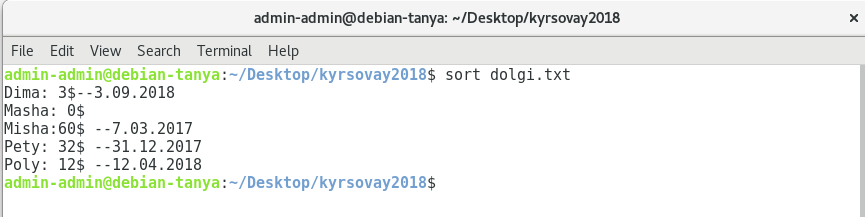


***Задача:*** Сделать рассортировку по имени в алфавитном порядке.

***Команда:***

Sort dolgi.txt

***Результат:***



Произошла рассортировка имен в алфавитном порядке.

***Опции sort***

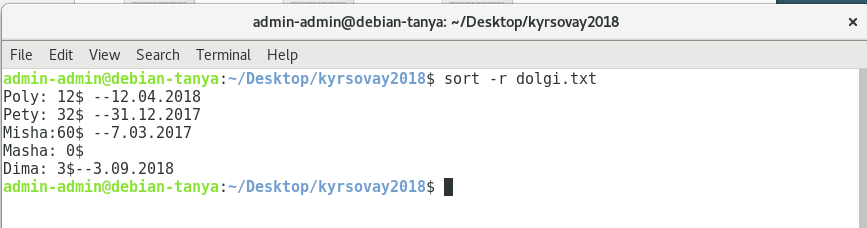
*1.Опция –r*

***Задача:*** Эта опция позволяет рассортировать файл в порядке обратном алфавитному.

***Команда:***

Sort –r dolgi.txt

***Результат:***



Произошла рассортировка имен в порядке обратном алфавитному.

*2,3.Опция –k и опция –n*

***Задача:*** Приходится рассматривать эти две опции вместе, так как они позволяют сразу показать весь спектр опций.

Опция -n используется всегда, когда нужно сортировать числа, разумеется в порядке возрастания (или убывания, добавив опцию -r).

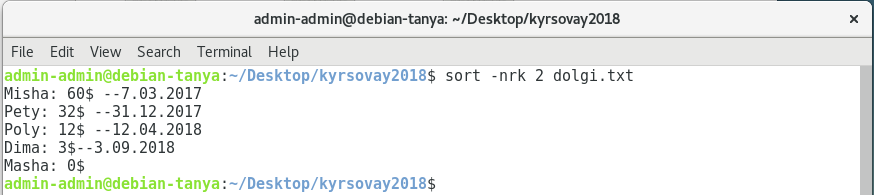
Опция -k позволяет задавать объект сортировки: все эти столбцы, колонки, и тому подобные элементы форматирования файла.

***Команда:***

Sort –nrk 2 dolgi.txt

***Результат:***

Итак, можно выявить самых злостных должников по мере убывания долга.



Опция -n сообщает команде, что сортировать придется числа, опция -r , что в обратном порядке, а опция -k задает объект - вторую колонку текста.

*4.Опция –m*

(--month-sort)

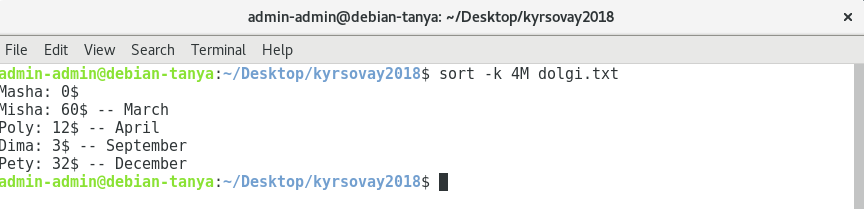
***Задача:*** Одной удивительной способностью команды sort является то, что она может сортировать даже месяцы.

***Команда:***

Sort –k 4M dolgi.txt

***Результат:***

Названия месяцев находятся в 4 колонке.



Опция М преобразует первые три непробельныx символа указанного столбца в заглавные буквы (Скажем, SEP означает SEPtember), а затем сравнивает их и располагает в порядке годового круга.

*5.Опция –u*

(--unique)

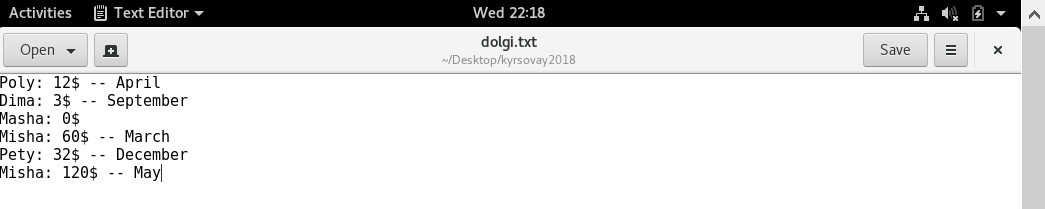
***Задача:*** Скрывает одинаковые объекты. Если в процессе сортировки выявилось несколько одинаковых объектов, то будет выведен только первый из них, остальные проигнорированы.

***Команда:***

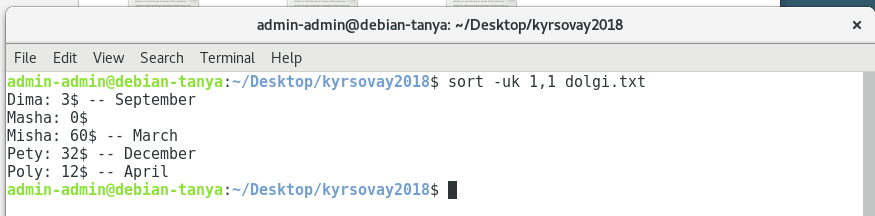
Sort -uk 1,1 dolgi.txt

***Результат:***

Начальный файл:



Результат:



Первая единица означает начало объекта сортировки (колонки текста), а вторая единица через запятую - конец объекта сортировки. То есть команде запрещено использовать для сортировки символы после последней буквы первой колонки.

Как оказалось, пользоваться утилитой sort для сортировки вывода предельно просто. Это полезная команда, с помощью которой не проблема выявить в огромной директории файлы, которые когда-то потерялись, или собрать в едино все ссылки, чтобы их проверить и так далее.

**6.Uniq**

Uniq — утилита Unix, с помощью которой можно вывести или отфильтровать повторяющиеся строки в файле. Если входной файл задан как («-») или не задан вовсе, чтение производится из стандартного ввода. Если выходной файл не задан, запись производится в стандартный вывод. Вторая и последующие копии повторяющихся соседних строк не записываются. Повторяющиеся входные строки не распознаются, если они не следуют строго друг за другом, поэтому может потребоваться предварительная сортировка файлов.

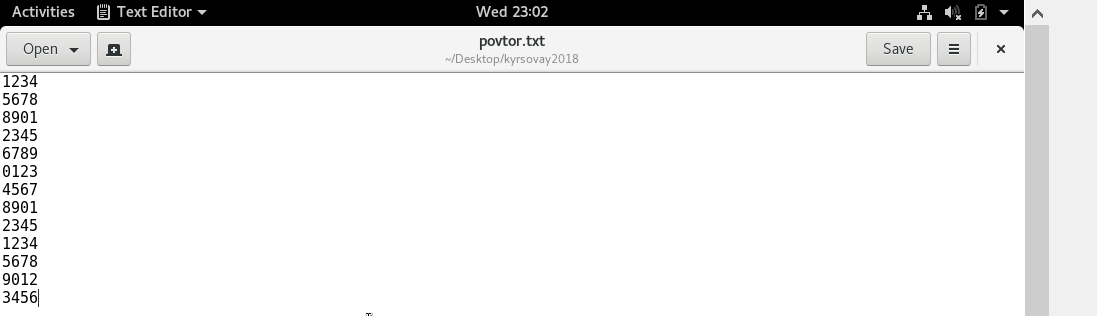
***Задача:*** Если в тексте следует подряд несколько одинаковых строк, то команда uniq уменьшит их количество до одной.

***Команда:***

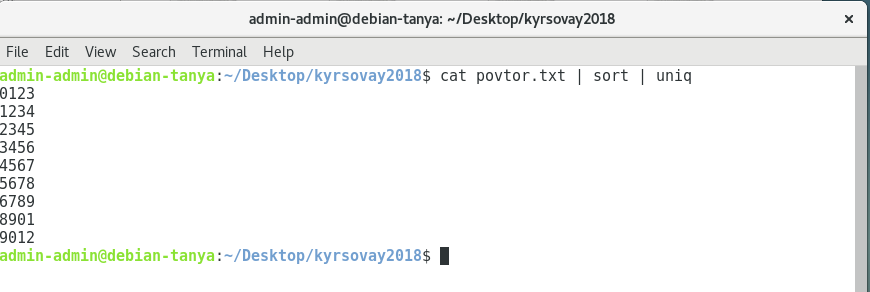
Cat povtor.txt | sort | uniq

***Результат:***

Начальный файл:

****

Результат:



Мы используем cat для отправки содержимого povtor.txt команде sort. Когда sort получает данные на вход, она сортирует их построчно в алфавитном порядке, и отправляет в таком виде программе uniq. uniq удаляет повторяющиеся строки. Тут нужно уточнить очень интересную вещь: команда uniq сравнивает только рядом расположенные строчки. Становится понятным, что команда uniq любит уже отсортированные тексты, где все одинаковые строчки собраны вместе.

***Опции uniq***

*1.Опция –c*

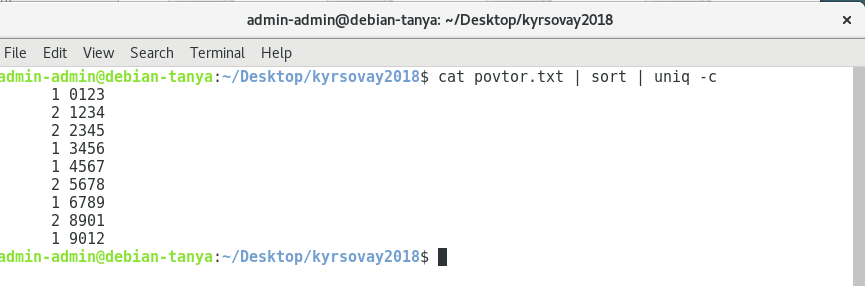
--count

***Задача:*** Сообщить, сколько было одинаковых строк до их урезания.

***Команда:***

Cat povtor.txt | sort | uniq -c  
 unia -c

***Результат:***

**

Например, для строки «1234» было 2 совпадения.

*2.Опция –d*

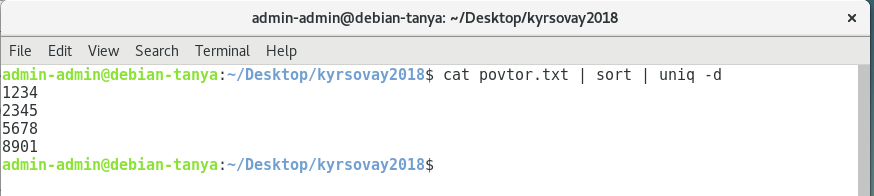
--repeated

***Задача:*** Эта опция, наоборот, выведет лишь ту строку, которая повторялась в тексте.

***Команда:***

Cat povtor.txt | sort | uniq -d

***Результат:***



На экране строки, которые повторились в тексте два и более раз.

*3.Опция –u*

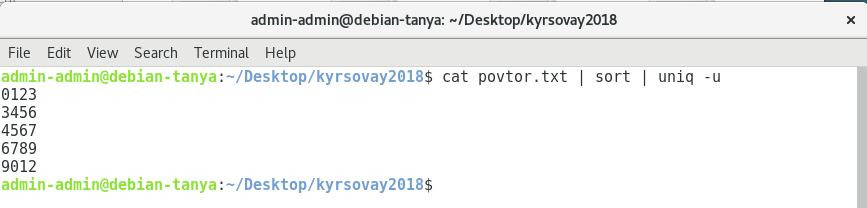
--unique

***Задача:*** Выводит только уникальные строки.

***Команда:***

Cat povtor.txt | sort | uniq –u

***Результат:***



На экране уникальные строки.

*4.Опция –D*

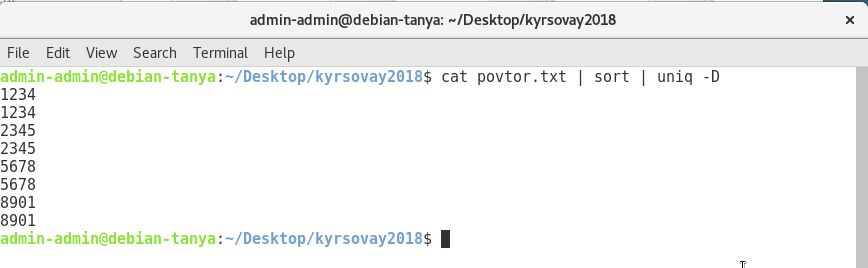
--all-repeated

***Задача:*** Выведет все повторяющиеся строки, не уменьшая их числа.

***Команда:***

Cat povtor.txt | sort | uniq -D

***Результат:***



На экране строки, которые повторялись несколько раз, при этом количество повторений сохранено.

Uniq хорошая команда для обработки файлов, когда есть повторяющиеся строки. Для работы с файлами без повторений лучше использовать команду sort.

**7.Wс**

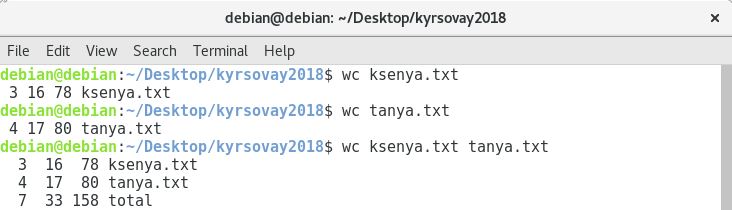
С помощью команды wc можно подсчитать число строк, слов и символов в указанном файле. Если указано более одного файла в командной строке, то программа wc осуществляет подсчет строк, слов и символов в каждом файле и затем выдает общее число.

***Задача:*** Подсчитать число строк, слов и символов в указанном файле.

***Команда:***

wс file

***Результат:***



***Опции wc***

*1.Опция-l*

***Задача:***  Вывести количество строк в указанном файле.

***Команда:***

wс –l tanya.txt

***Результат:***



В файле tanya.txt четыре строки.

*2.Опция-w*

***Задача:*** Вывести количество слов в указанном файле.

***Команда:***

wс –w ksenya.txt

***Результат:***



В файле ksenya.txt 16 строк.

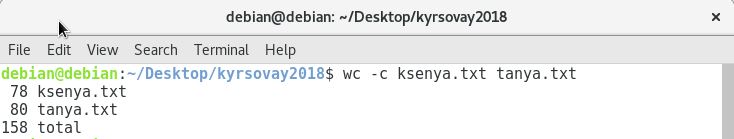
*3.Опция-c*

***Задача:*** Вывести количество символов в указанных файлах.

***Команда:***

wс –c ksenya.txt tanya.txt

***Результат:***



В файле ksenya.txt 78 символов, а в файле tanya.txt 80. Общее количество символов в файле 158.

Команда wc хороший помощник в подсчете количества строк, слов и символов. Эти данные смогут помочь нам в дальнейшей работе с файлами.

**8.Tr**

Команда tr служит для перевода (замены) выбранных символов в другие символы или удаления их.

В отличие от большинства других программ командной строки, команда tr не принимает имен файлов в качестве аргумента. Ввод команды tr осуществляется или со стандартного ввода, или с вывода других программ путем перенаправления.

Команде tr требуется минимум один аргумент, а принимается максимум два. Первый аргумент, называемый набор1 перечисляет символы, подлежащие замене или удалению. Второй аргумент, набор2, перечисляет символы, которые должны заменить символы из набора1.

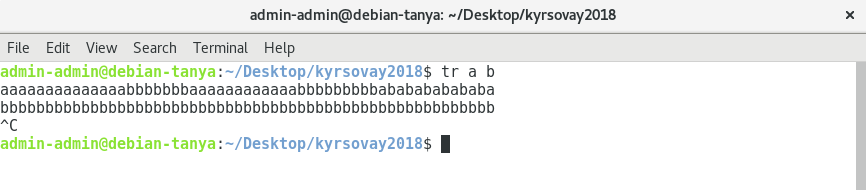
Команда tr без опций заменит символы из набора1 символами из набора2.

***Задача:*** Заменим все буквы a на буквы b. Для этого введем tr a b. Затем следует нажать Enter и команда будет готова к приему текста. Когда текс введен, то необходимо снова нажать Enter и так до бесконечности, пока необходимо будет заменять буквы в тексте. Когда нужно будет выйти из команды надо нажать клавиши Ctrl+C.

***Команда:***

tr a b

***Результат:***



***Опции tr***

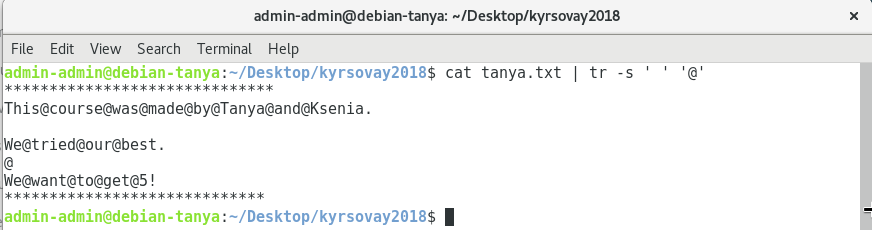
*1.Опция –s*

***Задача:*** Это одна из самых полезных опций команды tr. Эта опция позволяет заменить повторяющиеся подряд символы из набора1 на единственный символ из списка2. Если список 2 отсутствует, то заменяет множественные символы одиночными. Чаще всего эта уплотняющая текст опция применяется для замены множественных пробелов на один единственный.

***Команда:***

Cat tanya.txt | tr –s “ “ “@”

***Результат:***

**

Здесь заменили все пробелы, служащие разделителями в файле tanya.txt на собаку.

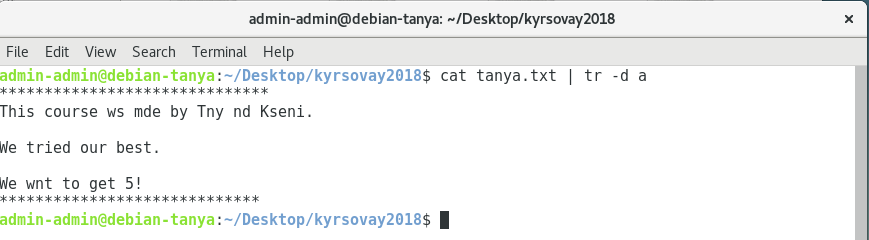
*2.Опция –d*

***Задача:*** Используется для удаления из текста символов, перечисленных в наборе1.

***Команда:***

Cat tanya.txt | tr –d a

***Результат:***



В примере удалились все символы «а» из файла tanya.txt

Опция может удалять также специальные символы, например, символ возврата каретки, который заканчивает строку в паре с символом новой строки в файлах ОС Windows. После его удаления останется только символ новой строки, который использует Юникс.

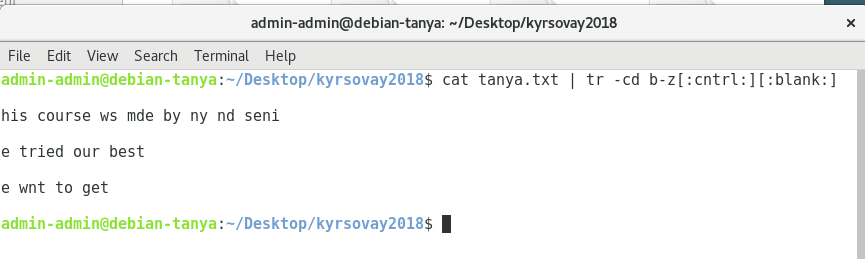
*3.Опция –c/cd*

***Задача:*** Заставляет команду работать с символами, которые отсутствуют в наборе 1. Другими словами: символы, перечисленные в наборе1 не используются в работе, а все остальные - используются.

***Команда:***

cat tanya.txt | tr –cd b-z[ :cntrl: ] [ :blank: ]

***Результат:***

**

В этом примере нет буквы а, вот ее и будет удалять программа. Символы из группы [:cntrl:] нужно указать, чтобы их не удалили, иначе не происходит перенос строки, а пробел - [:blank:] - указать, чтобы он тоже остался на месте.

Программа tr дает удивительные возможности изменять файлы, порой до неузнаваемости, простыми методами. Трудно предсказать, какое применение команде может найти пользователь, знающий ее тонкости.

**Заключение**

В данной курсовой работе мы рассмотрели основные команды фильтрации: grep, ack, awk, sort, uniq, ws, tr.

Одними из наиболее полезных и многофункциональных команд в терминале Linux являются команды фильтрации файлов. Эти, на первый взгляд, тривиальные команды очень мощны при верном использовании. Команды также значительно экономят время, потому стоит ознакомиться и подружиться с их параметрами и синтаксисом. Они очень обширны в функциональности, за счет большого количества поддерживаемых ими опций.

**Список используемой литературы.**

1. <https://habr.com/post/105926/>

2. <https://ru.wikipedia.org/wiki/Uniq>

3. <http://rus-linux.net/MyLDP/BOOKS/Linux_Foundations/17/ch17.html>

4.<http://www.linux-ink.ru/static/Docs/Courses/adv-user-guide/adv-user-guide/ch11s07.html>

5. <https://www.opennet.ru/docs/RUS/bash_scripting_guide/x7050.html>

6. <http://aidalinux.ru/w/Tr>

7. <http://aidalinux.ru/w/Uniq>

8. <http://aidalinux.ru/w/Grep>

9.<http://www.electronick.org.ua/articles/linux/kak-ispolzovat-komandu-grep-v-linux/>

10. <https://losst.ru/komanda-sort-v-linux>

11. <http://aidalinux.ru/w/Sort>

12. <http://citforum.ru/operating_systems/unixuser/gl3_10_5.shtml>

13. <https://habr.com/company/ruvds/blog/327754/>