Sed

Материал из Ай да Linux Wiki

Содержание

Введение

Команда sed - это редактор потока данных (Stream EDitor) для автоматического редактирования текстов. "Редактор потока" - в том смысле, что может редактировать входящий поток данных непрерывно, скажем, в составе программного канала (pipe). Автоматически - это значит, что, как только вы зададите правила редактирования, дальнейшее происходит без вашего утомительного участия. Другими словами, редактор sed не является интерактивным.

Программа sed сложнее, чем те команды, что мы уже успели рассмотреть в предыдущих статьях цикла HuMan. В ее составе арсенал собственных команд, поэтому, дабы избежать тавтологии и путаницы, в этой статье команда sed впредь будет именоваться "программой" или "редактором", а команды редактора sed - просто командами.

Программа sed способна выполнять сложные задания, и нужно потратить время, чтобы научиться эти задания формулировать.

Но наряду со сложными действиями, у команды sed есть простые, но весьма полезные возможности, освоить которые не труднее, чем прочие команды Юникс. Не позволяйте себе из-за сложности освоения всей программы, отказываться от ее простых аспектов.

Мы начнем от простого к сложному, так что вы всегда сможете понять, где следует остановиться.

Команда s - substitution (замена)

Программа sed имеет множество собственных команд. Большинство пользователей знают только команду s, и этого вполне хватает, чтобы работать с редактором sed. Команда s заменяет ОБРАЗЕЦ на ЗАМЕНУ:

sed s/ОБРАЗЕЦ/ЗАМЕНА/

```
$ echo день | sed s/день/ночь/ (Enter)
ночь
```

Проще не бывает. А вот пример с вводом из файла zar.txt:

```
По утрам он делал зарядку.
Молния - электрический заряд.
```

```
$ sed s/заряд/разряд/ zar.txt
По утрам он делал разрядку.
Молния - электрический разряд.
```

Я не брал выражение s/ОБРАЗЕЦ/ЗАМЕНУ/ в кавычки, так как данный пример не нуждается в кавычках, но если бы в нем присутствовали метасимволы, то кавычки были бы обязательны. Чтобы не ломать себе каждый раз голову, и не ошибиться ненароком, всегда ставьте кавычки, лучше более "сильные" одинарные, это хорошая привычка. Кашу маслом не испортишь. Я тоже во всех последующих примерах не буду манкировать кавычками.

Как мы видим, заменяющая команда s имеет четыре составляющих:

```
s               сама команда
/.../.../       разделитель
ОБРАЗЕЦ         образец для поиска и последующей замены
ЗАМЕНА          выражение, которое заменит собой ОБРАЗЕЦ, буде таковой найден.
```

Прямой слэш (/) используется в качестве разделителя по традиции, так как предок программы sed - редактор ed использует их (как и редактор vi). В некоторых случаях такой разделитель весьма неудобен, например, когда надо менять пути (path) к директориям, которые тоже содержат прямой слэш (/usr/local /bin). В этом случае приходится разделять прямые слэши обратными:

```
sed 's/\/usr\/local\/bin/\/common\/bin/'
```

Это называется "частокол" и выглядит весьма уродливо, а главное, непонятно.

Уникальность программы sed в том, что она позволяет использовать любой разделитель, например знак подчеркивания:

```
$ echo день | sed s_день_ночь_
ночь
```

или двоеточие:

```
$ echo день | sed s:день:ночь:
ночь
```

Если в поисках разделителя, который вам нравится, вы получаете сообщение "незавершенная команда `s'", значит этот символ не годится в качестве разделителя, или вы просто забыли поставить один-два

разделителя.

В этой статье я вынужден использовать традиционный разделитель (/) чтобы не сбивать читателя с толку, но в случае необходимости стану использовать в качестве разделителя тильду (~).

Регулярные выражения (РВ)

(Regular expressions, regexp, RE)

Тема регулярных выражений настолько обширна, что ей посвящены целые книги (смотри ссылки в конце статьи). Тем не менее, говорить всерьез о редакторе sed, не применяя регулярных выражений, также непродуктивно, как разговаривать о тригонометрии при помощи счетных палочек. Поэтому необходимо рассказать хотя бы о тех регулярных выражениях, которые часто используются с программой sed.

c Или любая другая буква. Большинство букв, цифр и прочих неспециальных символов считаются регулярными выражениями, представляющими сами себя.

***** Астериск, следующий за каким-либо символом или регулярным выражением, означает любое число (в том числе и нулевое) повторов этого символа или регулярного выражения.

\+ Означает один или более повтор символа или регулярного выражения.

\? Означает ни одного или один повтор.

\{i\} Означает ровно i повторов.

\{i,j\} Число повторов находится в интервале от i до j включительно.

\{i,\} Число повторов больше или равно i.

\{,j\} Число повторов меньше или равно j.

\(RE\) Запомнить регулярное выражение или его часть с целью дальнейшего использования как единое целое. Например, \(а-я\)* будет искать любое сочетание любого количества (в том числе и нулевого) строчных букв.

. Означает любой символ, в том числе символ новой строки.

^ Означает нулевое выражение в начале строки. Другими словами, то, перед чем стоит этот знак, должно появляться в начале строки. Например, ^#include будет искать строки, начинающиеся с #include.

$ То же, что и предыдущее, только относится к концу строки.

[СПИСОК] Означает любой символ из СПИСКА. Например, [aeiou] будет искать любую английскую гласную букву.

[^СПИСОК] Означает любой символ, кроме тех, что в списке. Например, [^aeiou] будет искать любую согласную. Примечание: СПИСОК может быть интервалом, например [а-я], что будет означать любую строчную букву. Если нужно включить в СПИСОК] (квадратную скобку) укажите ее в списке первой; если нужно включить в СПИСОК - (дефис), то укажите его в списке первым или последним.

RE1\|RE2 Означает РВ1 или РВ2.

RE1RE2 Означает объединение регулярных выражений РВ1 и РВ2.

\n Означает символ новой строки.

\$; *; \.; \[; \\; \^ Означают соответственно: *$; *; .; [; \; ^*

Внимание: Остальные условные обозначения на основе обратного слэша (\), принятые в языке С, не поддерживаются программой sed.

\1 \2 \3 \4 \5 \6 \7 \8 \9 Означает соответствующую по счету часть регулярного выражения, запомненную при помощи знаков \(и \).

Несколько примеров:

abcdef Означает abcdef

a*b Означает ноль или любое количество букв a и одна буква b. Например, aaaaaab; ab; или b.

a\?b Означает b или ab

a\+b\+ Означает одну или больше букв a и одну или больше букв b. Например: ab; aaaab; abbbbb; или aaaaaabbbbbbb.

.* Означает все символы на строке, на всех строках, включая пустые.

.\+ Означает все символы на строке, но только на строках, содержащих хотя бы один символ. Пустые строки не соответствуют данному регулярному выражению.

^main.*(.*) Будет искать строки, начинающиеся со слова main, а также имеющие в своем составе открывающую и закрывающие скобки, причем перед и после открывающей скобки может находиться любое количество символов (а может и не находиться).

^# Будет искать строки, начинающиеся со знака # (например комментарии).

\\$ Будет искать строки, заканчивающиеся обратным слэшем (\).

[a-zA-Z_] Любые буквы или цифры

[^]\+ (В квадратной скобке, кроме символа ^, содержится еще пробел и табуляция) --Означает один или любое количество любых символов, кроме пробела и табуляции. Обычно имеется в виду слово.

^.*A.*$ Означает заглавную букву А точно в середине строки.

A.\{9\}$ Означает заглавную букву А, точно десятую по счету от конца строки.

^.\{,15\}A Означает заглавную букву А, точно шестнадцатую по счету от начала строки.

Теперь, когда мы познакомились с некоторыми регулярными выражениями, вернемся к команде s редактора sed.

Использование символа & когда ОБРАЗЕЦ неизвестен "Как это неизвестен?", - спросите вы - "Ты разве не знаешь, что хочешь заменить?" Отвечу: я хочу взять в скобки любые цифры, найденные в тексте. Как это сделать? Ответ: применить символ &.

Символ & (амперсанд), будучи помещен в состав ЗАМЕНЫ, означает любой найденный в тексте ОБРАЗЕЦ. Например:

```
$ echo 1234 | sed 's/[0-9]*/(&)/'
(1234)
```

Звездочка (астериск) после интервала [0-9] нужна чтобы заменены были все цифры, встретившиеся в образце. Без нее получилось бы:

```
$ echo 1234 | sed 's/[0-9]/(&)/'
(1)234
```

То есть в качестве образца взята была первая же найденная цифра.

Вот пример со вполне осмысленной нагрузкой: составим файл formula.txt:

```
a+432-10=n
```

и применим к нему команду:

```
$ sed 's/[0-9]*-[0-9]*/(&)/' formula.txt
a+(432-10)=n
```

Математическая формула приобрела однозначный смысл.

Еще символ амперсанда можно использовать для удвоения ОБРАЗЦА:

```
$ echo 123 | sed "s/[0-9]*/& &/"
123 123
```

Тут есть одна тонкость. Если мы чуть усложним пример:

```
$ echo "123 abc" | sed "s/[0-9]*/& &/"
123 123 abc
```

как и следовало ожидать, удваиваются только цифры, так как в ОБРАЗЦЕ нет букв. Но если мы поменяем части текста местами:

```
$ echo "abc 123" | sed "s/[0-9]*/& &/"
abc 123
```

то никакого удвоения цифр не получится. Это особенность регулярного выражения [0-9]* - оно ищет соответствия только в первом символе строки. Если мы хотим удвоения цифр, где бы они ни находились, нужно доработать регулярное выражение в ЗАМЕНЕ:

```
$ echo "abc defg 123" | sed "s/[0-9][0-9]*/& &/"
abc defg 123 123
```

тогда цифры будут удваиваться, независимо от количества предшествующих "слов".

Использование условных знаков \(, \) и \1 для обработки части ОБРАЗЦА Условные знаки \(и \) (escaped parentheses) применяются для запоминания части регулярного выражения.

Условный знак \1 означает первую запомненную часть, \2 - вторую, и так далее, вплоть до девяти запомненных частей (больше программа не поддерживает). Разберем пример:

```
$ echo abcd123 | sed 's/\([a-z]*\).*/\1/'
abcd
```

Здесь \([a-z]*\) означает, что программа должна запомнить все буквенные символы в любом количестве; .* означает любое количество символов после первой запомненной части; а \1 означает , что мы хотим видеть только первую запомненную часть. Так и есть: в выводе программы мы видим только буквы и никаких цифр.

Для того, чтобы поменять слова местами, нужно запомнить два суб-ОБРАЗЦА, а потом поменять их местами:

```
$ echo глупый пингвин |sed 's/\([а-я]*\) \([а-я]*\)/\2 \1/'
пингвин глупый
```

Здесь \2 означает второй суб-ОБРАЗЕЦ, а \1 -первый. Обратите внимание на интервал между первым выражением \([а-я]*\) и вторым выражением \([а-я]*\). Он необходим, чтобы были найдены два слова.

Знак \1 вовсе не обязан быть только в ЗАМЕНЕ, он может присутствовать также и в ОБРАЗЦЕ, например, когда мы хотим удалить дубликаты слов:

```
$ echo пингвин пингвин | sed 's/\([а-я]*\) \1/\1/'
пингвин
```

Модификаторы замены команды s

Модификаторы замены ставятся после последнего разделителя. Эти модификаторы определяют действия программы в случае, если в строке нашлось более одного совпадения с ОБРАЗЦОМ, и каким образом производить замену.

Модификатор /g

- Глобальная замена (Global replacement)

Программа sed, как и большинство утилит Юникс, при работе с файлами считывают по одной строке. Если мы приказываем заменить слово, программа заменит только первое совпавшее с ОБРАЗЦОМ слово на данной строке. Если мы хотим изменить каждое слово, совпавшее с образцом, то следует ввести модификатор /g.

Без модификатора /g:

```
$ echo кот этот, был самый обычный кот | sed 's/кот/котенок/'
котенок этот, был самый обычный кот
```

Редактор заменил только первое совпавшее слово.

А теперь с модификатором глобальной замены:

```
$ echo кот этот, был самый обычный кот | sed 's/кот/котенок/g'
котенок этот, был самый обычный котенок
```

Все совпадения в данной строке были заменены.

А если нужно изменить все слова, скажем взять их в скобки? Тогда на помощь снова придут регулярные выражения. Чтобы выбрать все буквенные символы, как верхнего, так и нижнего регистра, можно воспользоваться конструкцией [А-Яа-я], но в нее не попадут такие слова как "что-то" или "с'езд". Гораздо удобнее конструкция [^]*, которая соответствует всем символам, кроме пробела. Итак:

```
$ echo глупый пингвин робко прячет | sed 's/[^ ]*/(&)/g'
(глупый) (пингвин) (робко) (прячет)
```

Как выбрать нужное совпадение из нескольких

Если не применять модификаторов, то программа sed заменит только первое совпавшее с ОБРАЗЦОМ слово. Если применить модификатор /g, то программа заменит каждое совпавшее слово. А как можно выбрать одно из совпадений, если их несколько на строке? - При помощи уже знакомых нам условных знаков \(и \) запомнить суб-ОБРАЗЦЫ и выбрать нужный при помощи знаков \1 - \9.

```
$ echo глупый пингвин | sed 's/\([а-я]*\) \([а-я]*\)/\2 /'
пингвин
```

В этом примере мы запомнили оба слова, и, поставив второе (пингвин) на первое место, первое (глупый) удалили, поставив в секции ЗАМЕНЫ вместо него пробел. Если мы поставим вместо пробела какое-либо слово, то оно заменит первое (глупый):

```
$ echo глупый пингвин | sed 's/\([а-я]*\) \([а-я]*\)/\2 умный /'
пингвин умный
```

Числовой модификатор

Это одно/двух/трех -значное число, которое ставится после последнего разделителя и указывает, какое по счету совпадение подлежит замене.

```
$ echo очень глупый пингвин | sed 's/[а-я]*/хороший/2'
очень хороший пингвин
```

В этом примере каждое слово является совпадением, и мы указали редактору, какое по счету слово мы хотим заменить, поставив модификатор 2 после секции ЗАМЕНЫ.

Можно комбинировать цифровой модификатор с модификатором /g. Если нужно оставить неизменным первое слово, а второе и последующие заменить на слово "(удалено)", то команда будет такая:

```
$ echo очень глупый пингвин | sed 's/[а-я]*/(удалено)/2g'
```

```
очень (удалено) (удалено)
```

Если нужно действительно удалить все последующие совпадения, кроме первого, то в секции ЗАМЕНЫ следует поставить пробел:

```
$ echo очень глупый пингвин | sed 's/[а-я]*/ /2g'
очень
```

Или вовсе ничего не ставить:

```
$ echo очень глупый пингвин | sed 's/[^ ]*//2g'
очень
```

Числовой модификатор может быть любым целым числом от 1 до 512. Например, если нужно поставить двоеточие после 80 символа каждой строки, то поможет команда:

```
$ sed 's/./&:/80' имя_файла
```

Модификатор /p - выдавать на стандартный выход (печатать - print)

Программа sed и так по умолчанию выдает результат на стандартный выход (например экран монитора). Этот модификатор применяется только с опцией sed -n, которая как раз блокирует вывод результата на экран.

Модификатор /w

Позволяет записывать результаты обработки текста в указанный файл:

```
$ sed 's/ОБРАЗЕЦ/ЗАМЕНА/w имя_файла
```

Модификатор /e (расширение GNU)

Позволяет указать команду шелла (не программы sed) в качестве ЗАМЕНЫ. Если соответствие ОБРАЗЦУ будет найдено, то оно будет заменено на вывод указанной в секции ЗАМЕНЫ команды. Пример:

```
$ echo ночь | sed 's/ночь/echo день/e'
день
```

Модификаторы /I и /i (расширение GNU)

Делают процесс замены нечувствительным к регистру символов.

```
$ echo Night | sed 's/night/day/i'
day
```

Комбинации модификаторов

Модификаторы можно комбинировать, когда это имеет смысл. При этом следует ставить модификатор w последним.

Условные обозначения (расширение GNU) Их всего пять:

\L переводит символы ЗАМЕНЫ в нижний регистр **\l** переводит следующий символ ЗАМЕНЫ в нижний регистр **\U** переводит символы ЗАМЕНЫ в верхний регистр **\u** переводит следующий символ ЗАМЕНЫ в верхний регистр **\E** отменяет перевод, начатый \L или \U По очевидным причинам эти условные обозначения применяются по одиночке. Например:

```
$ echo глупый пингвин | sed 's/глупый/\u&/'
Глупый пингвин
```

или:

```
$ echo маленький щенок | sed 's/[а-я]*/\u&/2'
маленький Щенок
```

Мы рассмотрели почти все аспекты команды s редактора sed. Теперь настала очередь рассмотреть опции этой программы.

Опции программы sed

Программа имеет на удивление мало опций. (Что несколько компенсирует избыток команд, модификаторов и прочих функций). Кроме общеизвестных опций --help (-h) и --version (-V), которые мы рассматривать не будем, их всего три:

Опция -e --expression=набор_команд

Один из способов выполнения нескольких команд - применение опции -e. Например:

```
sed -e 's/a/A/' -e 's/b/B/' имя_файла
```

Все предыдущие примеры в этой статье не требовали применения опции -e только потому, что содержали одну команду. Мы могли поставить в примерах опцию -e, это ничего бы не изменило.

Опция -f Если требуется выполнить большое количество команд, то удобнее записать их в файл и применить опцию -f:

```
sed -f sedscript имя-файла
```

Sedscript здесь - имя файла, содержащего команды. Этот файл называется скриптом программы sed (далее просто скрипт). Каждая команда скрипта должна занимать отдельную строку. Например:

```
# комментарий - Этот скрипт изменит все строчные гласные буквы на заглавные
s/a/A/g
s/e/E/g
s/i/I/g
s/o/O/g
s/u/U/g
```

Назвать скрипт можно как угодно, важно не путать файл скрипта с обрабатываемым файлом.

Опция -n Программа sed -n не выводит ничего на стандартный выход. Чтобы получить вывод нужно специальное указание. Мы уже познакомились с модификатором /p, при помощи которого можно дать такое указание. Вспомним файл zar.txt:

```
 $ sed 's/1-9/&/p' zar.txt
 По утрам он делал зарядку.
 Молния - электрический заряд.
```

Так как совпадений с ОБРАЗЦОМ не найдено (в файле нет цифр), то команда s с модификатором /p и знаком & в качестве ЗАМЕНЫ (напомню, что амперсанд означает сам ОБРАЗЕЦ), работает как команда cat.

Если ОБРАЗЕЦ будет найден в файле, то строки, содержащие ОБРАЗЕЦ, будут удвоены:

```
$ sed 's/зарядку/&/p' zar.txt
По утрам он делал зарядку.
По утрам он делал зарядку.
Молния - электрический заряд.
```

Теперь добавим опцию -n:

```
$ sed -n 's/зарядк/&/p' zar.txt
По утрам он делал зарядку.
```

Теперь наша программа работает как команда grep - возвращает только строки, содержащие ОБРАЗЕЦ.

Выбор нужных элементов редактируемого текста

Используя лишь одну команду s, мы убедились в необыкновенно широких возможностях редактора sed. А ведь все, что он делает, сводится к поиску и замене. Причем в процессе работы sed редактирует каждую строку поодиночке, не обращая внимания на другие. Было бы удобно ограничить круг строк, подлежащих изменению, например:

- Выбирать строки по номерам
- Выбирать строки в некотором диапазоне номеров
- Выбирать только строки, содержащие некое выражение
- Выбирать только строки между некоторыми выражениями
- Выбирать только строки от начала файла и до некоторого выражения
- Выбирать только строки от некоторого выражения и до конца файла

Программа sed умеет все это и даже больше. Любая команда редактора sed может применяться адресно, в некотором диапазоне адресов, или с вышеперечисленными ограничениями круга строк. Адрес или ограничение должны непосредственно предшествовать команде:

```
sed 'адрес/ограничение команда'
```

Выбор строк по номерам

Это самый простой случай. Просто указываем номер нужной строки перед командой:

```
$ sed '4 s/[а-я]*//i' gumilev.txt
Какая странная нега
В ранних сумерках утра,
В таянии вешнего снега,
всем, что гибнет и мудро.
```

Или:

```
$ sed '3 s/В/(В)/' gumilev.txt
Какая странная нега
В ранних сумерках утра,
(В) таянии вешнего снега,
Во всем, что гибнет и мудро.
```

Выбор строк в диапазоне номеров

Диапазон указывается, как не удивительно, через запятую:

```
$ sed '2,3 s/В/(В)/' gumilev.txt
Какая странная нега
(В) ранних сумерках утра,
(В) таянии вешнего снега,
Во всем, что гибнет и мудро.
```

Если нужно указать диапазон до последней строки файла, а вы не знаете, сколько в нем строк, то воспользуйтесь знаком $:

```
$ sed '2,$ s/в/(в)/i' gumilev.txt
Какая странная нега
(в) ранних сумерках утра,
(в) таянии вешнего снега,
(в)о всем, что гибнет и мудро.
```

Выбор строк, содержащих некое выражение

Искомое выражение заключается в прямые слэши (/) и ставится перед командой:

```
$ sed '/утра/ s/в/(в)/i' gumilev.txt
Какая странная нега
(в) ранних сумерках утра,
В таянии вешнего снега,
Во всем, что гибнет и мудро.
```

Выбор строк в диапазоне между двумя выражениями

Также как и в случае с номерами строк, диапазон задается через запятую:

```
$ sed '/утра/,/мудро/ s/в/(в)/i' gumilev.txt
Какая странная нега
(в) ранних сумерках утра,
(в) таянии вешнего снега,
(в)о всем, что гибнет и мудро.
```

Выбор строк от начала файла и до некоего выражения

```
$ sed '1,/снега/ s/в/(в)/i' gumilev.txt
Какая странная нега
(в) ранних сумерках утра,
(в) таянии вешнего снега,
Во всем, что гибнет и мудро.
```

Выбор строк от некоего выражения и до конца файла

```
$ sed '/снега/,$ s/в/(в)/i' gumilev.txt
Какая странная нега
В ранних сумерках утра,
(в) таянии вешнего снега,
(в)о всем, что гибнет и мудро.
```

Другие команды редактора sed

Команда d (delete)

Удаляет из стандартного вывода указанные строки:

```
$ sed '2 d' gumilev.txt
Какая странная нега
В таянии вешнего снега,
Во всем, что гибнет и мудро.
```

Причем чаще пишут проще (без пробела):

```
sed '2d' gumilev.txt
```

Все, что было сказано в предыдущем разделе о адресации строк, справедливо и для команды d (как и для почти всех команд редактора sed).

При помощи команды d удобно выбросить ненужную "шапку" какого-нибудь почтового сообщения:

```
$ sed '1,/^$/ d' имя_файла
```

(Удалить строки с первой и до первой пустой строки).

Избавится от комментариев в конфигурационном файле:

```
 $ sed '/^#/ d' /boot/grub/menu.lst
```

И мало ли, где нужно удалить лишние строчки!

Команда p (print)

Английское слово "print" переводится как "печатать", что в русском языке ассоциируется с принтером, или, по крайней мере, с клавиатурой. На самом же деле, слово это в английском контексте зачастую означает просто вывод на экран монитора. Так что команда p ничего не печатает, а просто выводит на экран указанные строки.

Будучи примененной сама по себе, команда p удваивает строки в выводе (ведь программа sed по умолчанию выводит строку на экран, а команда p выводит ту же строку вторично).

```
$ echo у меня есть кот | sed 'p'
у меня есть кот
у меня есть кот
```

Этому свойству находится применение, например удвоить пустые строки для улучшения вида текста:

```
$ sed '/^$/ p имя_файла
```

Но истинное свое лицо команда p раскрывает в сочетании с опцией -n, которая, как вы помните, запрещает вывод строк на экран. Комбинируя опцию -n с командой p, можно получить в выводе только нужные строки.

Например, просмотреть строки с первой по десятую:

```
$ sed -n '1,10 p' имя_файла
```

Или только комментарии:

```
$ sed -n '/^#/ p' /boot/grub/menu.lst
# GRUB configuration file '/boot/grub/menu.lst'.
# generated by 'grubconfig'.  Вск 23 Мар 2008 21:45:41
#
# Start GRUB global section
# End GRUB global section
# Linux bootable partition config begins
# Linux bootable partition config ends
# Linux bootable partition config begins
# Linux bootable partition config ends
```

Что весьма напоминает работу программы grep, с чем мы уже сталкивались, когда говорили об опции -n с модификатором /p. Но, в отличие от команды grep, редактор sed дает возможность не только найти эти строки, но и изменить их, заменив, например, везде Linux на Unix:

```
$ sed -n '/^#/ p' /boot/grub/menu.lst | sed 's/Linux/Unix/'
# GRUB configuration file '/boot/grub/menu.lst'.
# generated by 'grubconfig'.  Вск 23 Мар 2008 21:45:41
#
# Start GRUB global section
# End GRUB global section
# Unix bootable partition config begins
# Unix bootable partition config ends
# Unix bootable partition config begins
# Unix bootable partition config ends
```

Команда !

Иногда нужно бывает редактировать все строки, кроме тех, что соответствуют ОБРАЗЦУ, либо выбору. Символ восклицательного знака (!) инвертирует выбор. Например удалим все строки, кроме второй из четверостишия Гумилева:

```
$ sed '2 !d' gumilev.txt
В ранних сумерках утра,
```

Или выберем все строки, кроме комментариев, из файла /boot/grub/menu.lst:

```
$  sed -n '/^#/ !p' /boot/grub/menu.lst
default 1
timeout 20
gfxmenu (hd0,3)/boot/message
title SuSe on (/dev/hda3)
root (hd0,2)
kernel /boot/vmlinuz root=/dev/hda3 ro vga=773 acpi=off
title Linux on (/dev/hda4)
root (hd0,3)
kernel /boot/vmlinuz root=/dev/hda4 ro vga=0x317
```

Команда q (quit)

Команда q прекращает работу программы sed после указанной строки. Это удобно, если нужно прекратить редактирование после достижения определенного места в тексте:

```
$  sed '11 q' имя_файла
```

Эта команда закончит работу по достижению 11-й строки.

Команда q - одна из немногих команд sed, не принимающих диапазонов строк. Не может же команда прекратить работу 10 раз подряд, если мы введем:

```
sed '1,10 q'
Абсурд!
```

Команда w (write)

Подобно модификатору w команды s, эта команда позволяет записать вывод программы в файл:

```
$ sed -n '3,$ w gum.txt' gumilev.txt
```

Мы получим файл gum.txt, содержащий две последние строки четверостишия Гумилева из файла gumilev.txt. Причем, если такой файл уже существует, то будет перезаписан. Если не ввести опцию -n, то программа, кроме создания файла gum.txt, еще и выведет на экран все содержание файла gumilev.txt.

Для работы в командной строке, удобнее пользоваться обычным перенаправлением вывода (> или >>), но в sed скриптах, вероятно, команда w найдет свое применение.

Команда r (read)

Эта команда не только прочтет указанный файл, но и вставит его содержимое в нужное место редактируемого файла. Для выбора "нужного места" используется уже знакомая нам адресация (по номерам строк, по выражениям, и проч.). Пример:

```
$ echo Из стихотворения Гумилева: | sed 'r gumilev.txt'
```

Из стихотворения Гумилева:

```
Какая странная нега
В ранних сумерках утра,
В таянии вешнего снега,
Во всем, что гибнет и мудро.
```

Команда =

Выдаст номер указанной строки:

```
$ sed  '/снега/=' gumilev.txt
Какая странная нега
В ранних сумерках утра,
3
В таянии вешнего снега,
Во всем, что гибнет и мудро.
```

Или:

```
$ sed -n  '/снега/=' gumilev.txt
3
```

Команда принимает только один адрес, не принимает интервалов.

Команда y

Эта команда заменяет символы из секции ОБРАЗЕЦ символами секции ЗАМЕНА, работая как программа *tr*.

```
$  echo Автомобиль - наследие прошлого | sed 'y/Авто/Паро/'
Паромобиль - наследие прошлого
```

Команда *y* работает, только если количество символов в ОБРАЗЦЕ равно количеству символов в ЗАМЕНЕ.

Скрипты программы sed

Для того, чтобы пользоваться редактором sed как полноценным текстовым редактором, необходимо освоить составление скриптов sed. Программа sed имеет собственный несложный язык программирования, позволяющий составлять скрипты, способные творить чудеса.

Эта статья не может вместить описания скриптов sed, как и ее автор не ставит себе задачу освоение языка программирования sed. В этой статье я делал акцент на использование редактора sed в командной строке, имея прицел на использование его в качестве фильтра в программных каналах (pipes). По этой причине я опустил многочисленные команды sed, применяющиеся только в его скриптах.

Существует множество любителей редактора sed, и множество статей на тему скриптописания, в том числе и в Рунете. Так что для заинтересовавшихся этой замечательной программой не составит труда пополнить свои знания.

Программа sed и символы кириллицы

Как видно из примеров в этой статье, программа sed на правильно русифицированной системе свободно владеет "великим и могучим" языком.

Резюме программы sed

Программа sed - это многофункциональный редактор потока данных, незаменимый для:

- Редактирования больших текстовых массивов
- Редактирования файлов любой величины, когда последовательность редактирующих действий слишком сложна
- Редактирования данных по мере их поступления, в том числе в режиме реального времени - то есть в случаях, когда затруднительно или вовсе невозможно использовать интерактивные текстовые редакторы.

Для освоения программы sed в полном объеме потребуются недели или даже месяцы работы, так как для этого необходимо:

- Изучить регулярные выражения
- Научиться писать скрипты sed, освоив несложный язык программирования, применяющийся в этих скриптах

С другой стороны, освоить несколько наиболее употребительных команд редактора sed не сложнее, чем любую команду Юникс; надеюсь, данная статья поможет вам в этом.

Послесловие

До сих пор, в статьях цикла HuMan, я стремился хотя бы вкратце раскрыть каждую опцию, каждый параметр описываемой команды, так чтобы статья могла заменить маны. В дальнейшем я буду продолжать придерживаться этого принципа.

Данная статья является исключением, так как не описывает всех возможностей программы. Для полного их описания потребовалась бы не статья, а книга. Тем не менее, статья позволяет получить представление о редакторе sed и начать работать с этой удивительной программой, используя наиболее употребительные ее команды.
