Фильтры

Команды, которые были реализованы для использования совместно с программными **каналами**, называются **фильтрами**. Эти **фильтры** реализуются в виде простейших программ, которые крайне эффективно выполняют одну определенную задачу. Исходя из всего вышесказанного, они могут использоваться в качестве **строительных блоков** при создании сложных конструкций.

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

Фильтр cat

При размещении **фильтра cat** между двумя программными **каналами** не будет осуществляться какой-либо обработки передающихся через них данных (за исключением передачи этих данных из стандартного потока ввода **stdin** в стандартный поток вывода **stdout** фильтра).

```
[paul@RHEL4b pipes]$ tac count.txt | cat | cat | cat |
cat | cat
пять
четыре
три
два
один
[paul@RHEL4b pipes]$
```

Фильтр tee

Создание сложных **конвейеров** при работе с интерфейсом командной строки системы Unix является занимательным процессом, но иногда вам могут потребоваться промежуточные результаты работы конвейера. Это именно тот случай, когда фильтр **tee** может оказаться очень полезным. Фильтр **tee** перемещает данные из стандартного потока ввода **stdin** в стандартный поток вывода **stdout**, а также записывает их в файл. Исходя из вышесказанного, фильтр **tee** функционирует аналогично фильтру **cat**, за исключением того, что он имеет два идентичных вывода.

```
[paul@RHEL4b pipes]$ tac count.txt | tee temp.txt | tac
один
два
три
четыре
пять
[paul@RHEL4b pipes]$ cat temp.txt
пять
четыре
три
два
один
[paul@RHEL4b pipes]$
```

Фильтр grep

Фильтр **grep** снискал известность среди пользователей систем Unix. Наиболее простым сценарием использования фильтра **grep** является фильтрация строк текста, содержащих (или не содержащих) определенную подстроку.

```
[paul@RHEL4b pipes]$ cat tennis.txt
Amelie Mauresmo, Fra
Kim Clijsters, BEL
Justine Henin, Bel
Serena Williams, usa
Venus Williams, USA
[paul@RHEL4b pipes]$ cat tennis.txt | grep Williams
Serena Williams, usa
Venus Williams, USA
```

Вы можете выполнить эту же задачу без задействования фильтра cat.

```
[paul@RHEL4b pipes]$ grep Williams tennis.txt
Serena Williams, usa
Venus Williams, USA
```

Одним из наиболее полезных параметров фильтра grep является параметр **grep -i**, который позволяет производить фильтрацию строк без учета регистра.

```
[paul@RHEL4b pipes]$ grep Bel tennis.txt
Justine Henin, Bel
[paul@RHEL4b pipes]$ grep -i Bel tennis.txt
Kim Clijsters, BEL
Justine Henin, Bel
[paul@RHEL4b pipes]$
```

Другим полезным параметром является параметр **grep -v**, который позволяет осуществлять вывод строк, не содержащих заданную строку.

```
[paul@RHEL4b pipes]$ grep -v Fra tennis.txt
Kim Clijsters, BEL
Justine Henin, Bel
Serena Williams, usa
Venus Williams, USA
[paul@RHEL4b pipes]$
```

И, конечно же, оба описанных выше параметра могут комбинироваться для фильтрации всех строк без учета регистра и вывода тех из них, которые не содержат заданной строки.

```
[paul@RHEL4b pipes]$ grep -vi usa tennis.txt
Amelie Mauresmo, Fra
Kim Clijsters, BEL
Justine Henin, Bel
[paul@RHEL4b pipes]$
```

При использовании параметра **grep -A1** в вывод также будет добавлена одна строка, располагающаяся **после** обнаруженной строки.

```
paul@debian5:~/pipes$ grep -A1 Henin tennis.txt
Justine Henin, Bel
Serena Williams, usa
```

В случае использования параметра **grep -B1** в вывод будет добавлена одна строка, располагающаяся **до** обнаруженной строки.

```
paul@debian5:~/pipes$ grep -B1 Henin tennis.txt
```

```
Kim Clijsters, BEL
Justine Henin, Bel
```

С помощью параметра **grep -C1** (контекст) в вывод может быть добавлена одна строка, находящейся **до** обнаруженной строки, и одна строка, находящаяся **после** нее. Все три параметра (A, B и C) могут быть использованы для вывода произвольного количества дополнительных строк (например, могут быть использованы параметры A2, B4 или C20).

```
paul@debian5:~/pipes$ grep -C1 Henin tennis.txt
Kim Clijsters, BEL
Justine Henin, Bel
Serena Williams, usa
```

Фильтр cut

Фильтр **cut** может использоваться для извлечения данных из столбцов расположенных в файлах таблиц с указанием разделителя столбцов или количества байт данных в столбцах. В примере ниже фильтр **cut** используется для извлечения имени пользователя и его идентификатора из файла **/etc/passwd**. В качестве разделителя столбцов таблицы из данного файла используется символ двоеточия, при этом производится выборка значений первого и третьего столбцов.

```
[paul@RHEL4b pipes]$ cut -d: -f1,3 /etc/passwd | tail -4
Figo:510
Pfaff:511
Harry:516
Hermione:517
[paul@RHEL4b pipes]$
```

В случае использования фильтра **cut** с символом пробела в качестве разделителя вам придется экранировать этот символ пробела.

```
[paul@RHEL4b pipes]$ cut -d" " -f1 tennis.txt
Amelie
Kim
Justine
Serena
Venus
[paul@RHEL4b pipes]$
```

А в данном примере фильтр **cut** используется для вывода фрагментов строк файла **/etc/passwd** со второго по седьмой символ.

```
[paul@RHEL4b pipes]$ cut -c2-7 /etc/passwd | tail -4
igo:x:
faff:x
arry:x
ermion
[paul@RHEL4b pipes]$
```

Фильтр tr

Вы можете преобразовывать символы с помощью фильтра **tr**. В примере ниже показана процедура преобразования всех обнаруженных в потоке данных символов e в символы E.

```
[paul@RHEL4b pipes]$ cat tennis.txt | tr 'e' 'E'
AmEliE MaurEsmo, Fra
Kim ClijstErs, BEL
JustinE HEnin, BEl
SErEna Williams, usa
VEnus Williams, USA
```

В данном случае мы переводим все буквенные символы в верхний регистр, указывая два диапазона значений.

```
[paul@RHEL4b pipes]$ cat tennis.txt | tr 'a-z' 'A-Z'
AMELIE MAURESMO, FRA
KIM CLIJSTERS, BEL
JUSTINE HENIN, BEL
SERENA WILLIAMS, USA
VENUS WILLIAMS, USA
[paul@RHEL4b pipes]$
```

А здесь мы преобразовываем все символы новых строк в символы пробелов.

```
[paul@RHEL4b pipes]$ cat count.txt
один
два
три
четыре
```

```
пять
[paul@RHEL4b pipes]$ cat count.txt | tr '\n' ' '
один два три четыре пять [paul@RHEL4b pipes]$
```

Параметр **tr -s** также может использоваться для преобразования последовательностей из множества заданных символов в один символ.

```
[paul@RHEL4b pipes]$ cat spaces.txt
один    два         три
     четыре   пять  шесть
[paul@RHEL4b pipes]$ cat spaces.txt | tr -s ' '
один два три
 четыре пять шесть
[paul@RHEL4b pipes]$
```

Вы можете использовать фильтр **tr** даже для 'шифрования' текстов с использованием алгоритма **rot13**.

```
[paul@RHEL4b pipes]$ cat count.txt | tr 'a-z'
'nopqrstuvwxyzabcdefghijklm'
bar
gjb
guerr
sbhe
svir
[paul@RHEL4b pipes]$ cat count.txt | tr 'a-z' 'n-za-m'
bar
gjb
guerr
sbhe
svir
[paul@RHEL4b pipes]$
```

В последнем примере мы используем параметр **tr -d** для удаления заданного символа.

```
paul@debian5:~/pipes$ cat tennis.txt | tr -d e
Amli Maursmo, Fra
Kim Clijstrs, BEL
Justin Hnin, Bl
```

```
Srna Williams, usa
Vnus Williams, USA
```

Фильтр wc

Подсчет слов, строк и символов в файле осуществляется достаточно просто в случае использования фильтра **wc**.

```
[paul@RHEL4b pipes]$ wc tennis.txt
  5  15 100 tennis.txt
[paul@RHEL4b pipes]$ wc -l tennis.txt
5 tennis.txt
[paul@RHEL4b pipes]$ wc -w tennis.txt
15 tennis.txt
[paul@RHEL4b pipes]$ wc -c tennis.txt
100 tennis.txt
[paul@RHEL4b pipes]$
```

Фильтр sort

Фильтр **sort** по умолчанию сортирует строки файла в алфавитном порядке.

```
paul@debian5:~/pipes$ cat music.txt
Queen
Brel
Led Zeppelin
Abba
paul@debian5:~/pipes$ sort music.txt
Abba
Brel
Led Zeppelin
Queen
```

Но при этом фильтр **sort** поддерживает большое количество параметров, позволяющих повлиять на принцип его работы. В следующем примере показана методика сортировки строк на основе значений различных столбцов (столбца 1 и столбца 2 соответственно).

```
[paul@RHEL4b pipes]$ sort -k1 country.txt
Belgium, Brussels, 10
France, Paris, 60
```

```
Germany, Berlin, 100
Iran, Teheran, 70
Italy, Rome, 50
[paul@RHEL4b pipes]$ sort -k2 country.txt
Germany, Berlin, 100
Belgium, Brussels, 10
France, Paris, 60
Italy, Rome, 50
Iran, Teheran, 70
```

В примере ниже продемонстрировано различие между сортировкой в алфавитном порядке и сортировкой по числовым значениям (обе сортировки осуществлены на основе значений из третьего столбца).

```
[paul@RHEL4b pipes]$ sort -k3 country.txt
Belgium, Brussels, 10
Germany, Berlin, 100
Italy, Rome, 50
France, Paris, 60
Iran, Teheran, 70
[paul@RHEL4b pipes]$ sort -n -k3 country.txt
Belgium, Brussels, 10
Italy, Rome, 50
France, Paris, 60
Iran, Teheran, 70
Germany, Berlin, 100
```

Фильтр `uniq`

С помощью фильтра **`uniq`** вы можете удалить повторяющиеся строки из **отсортированного списка** строк.

```
paul@debian5:~/pipes$ cat music.txt
Queen
Brel
Queen
Abba
paul@debian5:~/pipes$ sort music.txt
```

```
Abba
Brel
Queen
Queen
paul@debian5:~/pipes$ sort music.txt |uniq
Abba
Brel
Queen
```

Также в случае использования параметра **-c** фильтр **uniq** может вести подсчет повторений строк.

```
paul@debian5:~/pipes$ sort music.txt |uniq -c
      1 Abba
      1 Brel
      2 Queen
```

Фильтр comm

Сравнение потоков данных (или файлов) может быть осуществлено с помощью фильтра **comm**. По умолчанию фильтр **comm** будет выводить данные в трех столбцах. В данном примере строки Abba, Cure и Queen присутствуют в списках из обоих файлов, строки Bowie и Sweet только в списке из первого файла, а строка Turnet - только в списке из второго файла.

```
paul@debian5:~/pipes$ cat > list1.txt
Abba
Bowie
Cure
Queen
Sweet
paul@debian5:~/pipes$ cat > list2.txt
Abba
Cure
Queen
Turner
paul@debian5:~/pipes$ comm list1.txt list2.txt
                Abba
```

```
Bowie
                Cure
                Queen
Sweet
        Turner
```

Вывод фильтра **comm** лучше читается в случае формирования одного столбца. При этом с помощью цифровых параметров должны быть указаны столбцы, содержимое которых не должно выводиться.

```
paul@debian5:~/pipes$ comm -12 list1.txt list2.txt
Abba
Cure
Queen
paul@debian5:~/pipes$ comm -13 list1.txt list2.txt
Turner
paul@debian5:~/pipes$ comm -23 list1.txt list2.txt
Bowie
Sweet
```

Фильтр od

Несмотря на то, что жители Европы предпочитают работать с символами ascii, компьютеры используют байты для хранения данных файлов. В примере ниже создается простой файл, после чего для показа его содержимого в форме шестнадцатеричных значений байт используется фильтр **od**.

```
paul@laika:~/test$ cat > text.txt
abcdefg
1234567
paul@laika:~/test$ od -t x1 text.txt
0000000 61 62 63 64 65 66 67 0a 31 32 33 34 35 36 37 0a
0000020
```

Содержимое этого же файла может быть выведено и в форме восьмеричных значений байт.

```
paul@laika:~/test$ od -b text.txt
0000000 141 142 143 144 145 146 147 012 061 062 063 064
065 066 067 012
0000020
```

А это содержимое рассматриваемого файла в форме символов ascii (или экранированных символов).

```
paul@laika:~/test$ od -c text.txt
0000000   a   b   c   d   e   f   g  \n   1   2   3   4
5   6   7  \n
0000020
```

Фильтр sed

Фильтр **sed** (расшифровывается как **s**tream **ed**itor **-** редактор потока данных) позволяет выполнять различные операции редактирования при обработке потока данных с задействованием **регулярных выражений**.

```
paul@debian5:~/pipes$ echo уровень5 | sed 's/5/42/'
уровень42
paul@debian5:~/pipes$ echo уровень5 | sed 's/уровень/пе-
реход/'
переход5
```

Следует добавить флаг регулярного выражения **g** для осуществления глобальной замены (замены всех вхождений заданной строки в строку из потока данных).

```
paul@debian5:~/pipes$ echo уровень5 уровень7 | sed
's/уровень/переход/'
переход5 уровень7
paul@debian5:~/pipes$ echo уровень5 уровень7 | sed
's/уровень/переход/g'
переход5 переход7
```

С помощью флага регулярного выражения **d** вы можете осуществить удаление строк, содержащих заданную последовательность символов, из потока данных.

```
paul@debian5:~/test42$ cat tennis.txt
Venus Williams, USA
Martina Hingis, SUI
Justine Henin, BE
Serena williams, USA
Kim Clijsters, BE
Yanina Wickmayer, BE
paul@debian5:~/test42$ cat tennis.txt | sed '/BE/d'
```

```
Venus Williams, USA
Martina Hingis, SUI
Serena williams, USA
```

Примеры конвейеров

Конвейер `who | wc`

Сколькими пользователями был осуществлен вход в систему?

```
[paul@RHEL4b pipes]$ who
root     tty1         июл 25 10:50
paul     pts/0        июл 25 09:29 (laika)
Harry    pts/1        июл 25 12:26 (barry)
paul     pts/2        июл 25 12:26 (pasha)
[paul@RHEL4b pipes]$ who | wc -l
4
```

Конвейер `who | cut | sort`

Вывод отсортированного списка пользователей, осуществивших вход в систему.

```
[paul@RHEL4b pipes]$ who | cut -d' ' -f1 | sort
Harry
paul
paul
root
```

Вывод отсортированного списка пользователей, осуществивших вход в систему, в котором имя каждого пользователя приводится лишь единожды.

```
[paul@RHEL4b pipes]$ who | cut -d' ' -f1 | sort | uniq
Harry
paul
root
```

Конвейер `grep | cut`

Вывод списка всех **учетных записей пользователей**, использующих командную оболочку bash на данном компьютере. Учетные записи пользователей будут подробно обсуждаться позднее.

```
paul@debian5:~$ grep bash /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

```
paul:x:1000:1000:paul,,,:/home/paul:/bin/bash
serena:x:1001:1001::/home/serena:/bin/bash
paul@debian5:~$ grep bash /etc/passwd | cut -d: -f1
root
paul
serena
```