Программы для работы с текстом

Основная философия, которая началась с Unix и продолжается в Linux, заключается в том, что команды должны читать текст как ввод и писать текст как вывод. Чтобы соответствовать этой философии, существует большое количество команд, которые были разработаны с основной целью обработки текстовых файлов. Как вы увидите, многие из этих команд действуют как фильтры, так как они каким-то образом изменяют текст, который они читают, прежде чем они произведут свой вывод.

Команда cat

С одним аргументом файла cat просто выведет содержимое файла:

```
cat /etc/passwd
```

Для просмотра файлов с большим содержимым необходимо использовать программу more или less

```
cat /var/log/messages | more
less /var/log/messages
```

Команда file

Чтобы избежать просмотра бинарных файлов, используйте команду **file**. Команда **file** отобразит тип файла в зависимости от содержимого файла. В следующем примере обратите внимание, что первый файл / bin / ls - это формат исполняемой ссылки (ELF); этот файл не должен отображаться командой cat. Файл / etc / profile находится в текстовом формате ASCII на английском языке и может быть просмотрен без проблем:

```
file /bin/ls
```

Команда split

Если **cat** может объединить два или более файлов в один файл, команда **split** может взять один файл и разбить его на несколько файлов. Одно из применений команды split - взять файл, который слишком велик для размещения на каком-либо съемном носителе, и разбить этот файл на части, чтобы каждый фрагмент мог быть сохранен на отдельном носителе. Таким образом, фрагменты разделенных файлов могут помещаться на сменный носитель для передачи в другую систему.

Это также может быть полезным способом отправки больших файлов по медленной сети, в которой есть проблемы с подключением. Разбиение файлов на более мелкие части облегчило бы передачу данных через плохое сетевое соединение.

После того, как фрагменты скопированы в другую систему, можно использовать команду cat для повторной сборки фрагментов в один файл.

Синтаксис для команды split:

```
split [OPTION]... [INPUT [PREFIX]]
```

По умолчанию новые файлы будут иметь имена с префиксом x и алфавитным суффиксом aa, ab и т.д .:

```
cp /var/log/messages messages
split messages
ls
```

Как указано в синтаксисе команды, можно указать префикс, отличный от x. Например, разделить файл messages. обозначает префикс **file.**:

Опция -d позволит разделенным файлам иметь числовой суффикс вместо алфавитного суффикса по умолчанию. Если -d добавлен к предыдущей команде, например, так разделить -d longfile.txt файл. результирующие имена файлов будут такими:

По умолчанию команда split разбивает файл на 1000 строк. Первые 1000 строк исходного файла перейдут в первый файл, вторые 1000 строк перейдут во второй файл и т. Д. Вышеупомянутый файл longfile.txt разбит на 10 файлов, что означает, что его длина составляет приблизительно 10000 строк.

Опцию **-l** можно использовать для указания количества строк, на которые нужно разделить. В качестве альтернативы, опция **-b** может использоваться для указания максимального количества байтов для использования в файле.

Команда nl

Команда nl будет нумеровать строки. По умолчанию это только номера непустых строк:

```
nl messages | more
```

Команда head

Целью команды **head** является просмотр начала файла или вывода. По умолчанию команда **head** отображает первые десять строк содержимого файла. Например, следующая команда отображает первые десять строк файла messages:

```
head messages
```

Есть несколько опций для команды head, которые полезны. Например, есть возможность использовать число в качестве опции, чтобы указать, сколько строк вывода отображать. Например, для отображения первых трех строк файла messages выполните:

```
head -3 messages
```

Существует также опция -n, которая принимает аргумент для количества отображаемых строк. Таким образом, следующее также будет отображать первые три строки файла alpha.txt:

```
head -n3 messages
```

Отрицательное число также может использоваться в качестве аргумента опции -n, которая сообщает команде head, сколько строк пропустить в нижней части файла. Например, выполнив команду head -n -90 для файла длиной 100 строк, вывод будет включать только первые 10 строк, пропуская последние 90.

```
head –n -90 messages
```

Команда tail

Команда tail, противоположная команде head, отображает содержимое с конца файла, а не с начала. Он может использоваться в командных каналах аналогично команде head. Команда tail также отображает десять строк по умолчанию без опций:

```
tail messages
```

Вы можете использовать число или параметр -n в качестве аргумента для хвостовой команды, чтобы указать, сколько строк вы хотите вывести из конца файла. Следовательно, следующие две команды эквивалентны:

```
tail -n3 messages
```

У опции tail -n также есть интересный поворот. При использовании опции -n с номером, начинающимся с префикса + (знак плюс), номер интерпретируется как номер строки в файле, с которого начинается отображение содержимого; он будет отображаться от этой строки до конца файла. Другими словами, команда tail -n +20 alpha.txt отобразит содержимое файла alpha.txt, начиная со строки двадцать и продолжая до конца файла.

```
tail -n +20 messages
```

Команда cut

Команда cut извлекает поля информации из текстового файла. Разделителем полей по умолчанию является пробел или табуляция; это можно изменить с помощью опции -d. Используйте параметр -f, чтобы указать список номеров полей, разделенных запятыми, для отображения или дефисный диапазон отображаемых полей.

Обратите внимание на вывод следующей команды head. Есть 7 полей, каждое из которых разделено разделителем двоеточия:

```
head -1 /etc/passwd
```

```
root:x:0:0:root:/root:/bin/bash
  |  | | |  |     |      |
  1  2 3 4  5     6      7
```

Чтобы извлечь определенные поля из файла / etc / passwd, необходимо указать разделитель в качестве символа двоеточия. Если бы важными были только поля 1,5,6 и 7, их можно было бы извлечь с помощью следующей команды:

```
head -1 /etc/passwd | cut -d: -f1,5,6,7
```

Для файла, который имеет поля в фиксированных позициях символов, команда cut может использоваться для извлечения символов по одному, используя параметр -c, за которым следует диапазон.

Например, в файле / var / syslog; первые пятнадцать символов указывают метку времени. Чтобы извлечь только те символы из текста, укажите от 1 до 15:

```
tail /var/log/syslog | cut -c1-15
```

Команда sort

Команда sort используется для отображения файла, отсортированного по определенному полю данных. Эта команда может сортировать по любому полю так же, как электронная таблица по любому столбцу. Использование сортировки без параметров для /etc/passwd приводит к сортировке строк в порядке ASCII, который аналогичен алфавитному порядку:

```
sort /etc/passwd
```

По умолчанию сортировка разбивает каждую строку файла на поля, используя пробелы (табуляции или пробелы) в качестве разделителей. Чтобы указать альтернативный разделитель, используйте параметр -t. Чтобы указать поля для сортировки от первого до последнего, используйте один или несколько параметров -k. В следующем примере файл /etc/passwd отсортирован по второму полю данных с использованием символов в качестве разделителей:

```
sort –t ‘:’ –k2 /etc/passwd
```

Чтобы изменить направление сортировки с восходящего на нисходящее, добавьте r в спецификацию ключевого поля:

```
sort –t ‘:’ –k2 –r /etc/passwd
```

Команду sort можно комбинировать с другими командами для выполнения более сложных задач. Например, выходные данные команды cut -f7 -d: / etc / passwd содержат много повторяющихся значений:

```
cut -f7 -d: /etc/passwd | head -n4
```

Посылая эти выходные данные команде сортировки и используя опцию -u (u для уникального), дубликаты будут удалены из выходных данных:

```
cut -f7 -d: /etc/passwd | sort –u
/bin/bash
/bin/false
/bin/sync
/usr/sbin/nologin
```

Команда pr

Команда **pr** подготавливает файл для печати, разбивая его на «страницы» и отображая информацию заголовка вверху каждой страницы. При выполнении команды pr file файл будет отправлен на стандартный вывод с заголовками страниц для каждой страницы, содержащей метку времени, путь к файлу и номер страницы.

Некоторые полезные опции **pr:**

По умолчанию команда pr по умолчанию имеет длину 56 строк на странице, но параметр **-l** можно использовать для указания длины.

Каждая страница также содержит информацию «заголовка», которая включает метку времени, имя файла и номер страницы. Для подавления заголовков используйте опцию **-t**.

Опция **-d** может использоваться для автоматического вывода вывода через два интервала.

Смещение или поле можно указать с помощью параметра **-o**, а ширину - с помощью параметра **-w**.

В следующем примере отображается заголовок страницы и двойные пробелы в документе:

```
pr -d messages
```

Команда tr

Команда **tr** может использоваться для перевода одного набора символов в другой. В следующем примере все строчные буквы в messages будут переведены в прописные буквы.

tr 'a-z' 'A-Z' < messages

Символы, которые переводит команда tr, не обязательно должны находиться в диапазоне, их можно просто перечислить для каждого набора символов. Будьте осторожны, чтобы сбалансировать наборы, чтобы убедиться, что они имеют одинаковое количество символов, иначе могут возникнуть странные результаты. В следующем примере команда tr заменяет обычные символы символами:

cat messages | tr 'aeiou' '@&1*^'

Помимо выполнения функции перевода, команда tr также может «сжимать» повторяющиеся символы и удалять символы. При выполнении этих функций необходимо указывать только один набор символов. Например, чтобы удалить дубликаты, используйте опцию -s, чтобы сжать повторы:

echo 'aaaaaappleeeeee' | tr -s 'ae'

Использование опции -d удалит символы из первого набора:

cat messages | tr -d 'AEIOUaeiou'

Команда sed

Команда редактора потока sed - это неинтерактивный редактор, который можно использовать для изменения текста.

Чтобы выполнения простого поиска и замены в файле messages слова root на слово toor, необходимо передать команде **sed** аргумент **'s/root/toor/'**. Он будет искать то, что находится между первыми двумя косыми чертами, и если он найдет этот текст, то он заменит его тем, что находится между двумя последними косыми чертами.

sed 's/root/toor'

В отличие от большинства команд фильтрации, команда sed может изменять исходный файл с помощью параметра **-i**. Параметр **-i** позволяет указать необязательный аргумент, который будет расширением, в результате чего будет создан новый файл, который является копией исходного файла. , Например, если вы хотите сделать резервную копию исходного файла в виде файла messages.change01, выполните следующее:

sed -i'.change01' 's/root/toor/' messages

Модификатор "g"

При выполнении операции поиска и замены команда sed по умолчанию заменяет только первое вхождение шаблона. Чтобы заменить все вхождения шаблона, добавьте глобальный модификатор g после последней косой черты.

Например, для замены oo на 00 выражение 's /oo/OO/' заменит только первое вхождение шаблона oo. Если вы хотите заменить каждое вхождение oo на 00, используйте выражение 's /oo/00/g':

sed 's/oo/OO/g' messages

```
root@L-FW:~# sed 's/oo/OO/g' passwd
rOOt:x:0:0:rOOt:/rOOt:/bin/bash
```

Команда sed также может вставлять текст перед шаблоном или после шаблона. Для этого выражения не используйте символ s перед первой косой чертой; используйте изменение вставки с помощью \ \ или изменение дополнения с помощью \. Обратите внимание, что изменение влияет либо на строку до (с **i**\), либо после (с **a**\), где был найден шаблон:

sed '/root/i\Hello' passwd

```
root@L-FW:~# sed '/root/i\HELLO' passwd
HELLO
root:x:0:0:root:/root:/bin/bash
```

sed '/root/i\Hello' passwd

```
root@L-FW:~# sed '/root/a\HELLO' passwd
root:x:0:0:root:/root:/bin/bash
HELLO
```

Команда sed также может использоваться для поиска строки текста, содержащей шаблон, а затем для удаления соответствующих строк. Поместите шаблон для поиска между двумя косыми чертами, а затем введите **d**, чтобы выполнить эту операцию. Следующая команда удалит строку, содержащую root:

sed '/root/d' passwd

Чтобы изменить всю строку, которая соответствует шаблону, на что-то другое, используйте косую черту, соответствующий шаблон, еще одну косую черту, с \, а затем новый текст. Например, чтобы изменить строку, содержащую root на ROOT, используйте следующую команду:

sed '/root/c\ROOT' passwd

```
root@L-FW:~# sed '/root/c\ROOT' passwd
ROOT
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
```

!!! Имейте в виду, что будет заменена вся строка, а не только соответствующий шаблон:

Команда sed обычно анализирует только одно выражение для фильтрации текста. Чтобы использовать несколько выражений, используйте параметр **-e** с каждым выражением для изменения файла.

sed -e '/root/c\ROOT' -e '/daemon/d' passwd

```
root@L-FW:~# sed -e '/root/c\ROOT' -e '/daemon/d' passwd
ROOT
bin:x:2:2:bin:/bin:/usr/sbin/nologin
```

Порядок размещения этих выражений имеет значение

Самый мощный аспект команды sed - это ее использование с регулярными выражениями. До сих пор в каждом примере команда sed использовала буквальные выражения, включающие символы, такие как t или o, или строки, подобные трем. Регулярные выражения можно использовать для создания шаблонов для использования с командой sed и многими другими, гораздо более мощными командами.

Команда wc

Команда **wc** может использоваться для анализа текстового файла. По умолчанию wc считает количество строк, слов и байтов в отрывке и выводит эту информацию в следующем формате:

lines words bytes filename

Команда cat выводит содержимое файла:

cat passwd

Использование команды wc в том же файле показывает следующее:

wc passwd

```
root@L-FW:~# wc passwd
  27   39 1421 passwd
```

Команда wc сообщает, что имеется 27 строк, содержащих 39 слов общим объемом 1421 байт. Общее количество байтов включает пробелы, знаки пунктуации и возврат каретки.

Для просмотра каждого из этих номеров в отдельности можно использовать следующие параметры:

wc passwd –l
wc passwd –w
wc passwd -m

В первом примере опция -l использовалась для подсчета количества строк. Во втором примере опция -w использовалась для подсчета количества слов. Наконец, опция -m указывает количество символов.

Можно предположить, что опция -c будет подсчитывать количество символов, но -c используется для подсчета количества байтов. Количество байтов может варьироваться, если в файле есть непечатаемые символы. В этом случае они одинаковы:

wc passwd -c

Последний параметр -L возвращает максимальную длину строки в файле.

wc passwd -L

Если требуется подсчет слов для всех файлов в определенной папке, можно использовать символ звездочки *:

wc *.

```
    9    23   221 delzone
   12    40   350 delzone1
   24    96   724 ipbase
    0     0     0 lpicfile.txt
11974 171749 1204635 messages
   27    39  1421 passwd
12046 171947 1207351 total
```