Изучаем Linux, 101: Текстовые потоки и фильтры

Обработка текста в командной строке с использованием текстовых утилит GNU

Ян Шилдс (ishields@us.ibm.com)
Старший программист
IBM

15.09.2011

Помимо удаления и вставки существует множество операций, которые можно выполнять с текстовыми данными и которые особенно полезны, если вы не используете графический интерфейс. В этой статье Ян Шилдс расскажет вам о различных приемах работы с текстом в ОС Linux® с использованием фильтров из состава пакета GNU textutils. Освоив материал этой статьи, вы сможете работать с текстовыми данными на уровне эксперта. Вы можете использовать этот материал для подготовки к экзамену LPI 101 программы сертификации на администратора Linux начального уровня или просто для общего развития. [*Прим. ред.: первая строка листинга 7 была исправлена, благодаря внимательному читателю*]

Больше статей из этой серии

Развить навыки по этой теме

Этот материал — часть knowledge path для развития ваших навыков. Смотри Основы системного администрирования Linux: работа с консолью

Краткий обзор

В этой статье вы познакомитесь с *фильтрами*, которые позволяют строить сложные *конвейеры* для манипуляций с текстом. Вы узнаете, как выводить текст, сортировать его, выполнять подсчет слов и строк, преобразовывать символы, а также о многом другом. Также вы научитесь работать с потоковым редактором *sed*.

В этой статье будут рассмотрены следующие темы:

Об этой серии

Эта серия статей поможет вам освоить задачи администрирования операционной системы Linux. Вы также можете использовать материал этих статей для подготовки к экзаменам первого уровня сертификации профессионального института Linux (LPIC-1).

Чтобы посмотреть описания статей этой серии и получить ссылки на них, обратитесь к нашему перечню материалов для подготовки к экзаменам LPIC-1. Этот перечень постоянно дополняется новыми статьями по мере их готовности и содержит самые последние (по состоянию на апрель 2009 года) цели экзаменов сертификации LPIC-1. Если какая-либо статья отсутствует в перечне, можно найти ее более раннюю версию, соответствующую предыдущим целям LPIC-1 (до апреля 2009 года), обратившись к нашим руководствам для подготовки к экзаменам института Linux Professional Institute.

- Обработка текстовых файлов и потоков вывода текстовыми фильтрами для модификации вывода.
- Использование стандартных команд UNIX из состава пакета GNU textutils.
- Использование редактора sed для создания сценариев, состоящих из сложных последовательностей действий над текстовыми файлами.

Эта статья поможет вам подготовиться к сдаче экзамена LPI 101 на администратора начального уровня (LPIC-1) и содержит материалы цели 103.2 темы 103. Цель имеет вес 3. Материал этой статьи соответствует целям экзамена LPI 101 по состоянию на апрель 2009 года. Всегда обращайтесь к Web-сайту программы сертификации LPIC, чтобы уточнить последние цели.

Необходимые условия

Чтобы извлечь наибольшую пользу из наших статей, необходимо обладать базовыми знаниями о Linux и иметь работоспособный компьютер с Linux, на котором можно будет выполнять все встречающиеся команды. Иногда различные версии программ выводят результаты по-разному, поэтому содержимое листингов и рисунков может отличаться от того, что вы увидите на вашем компьютере.

Фильтрация текста

Как связаться с Яном

Ян – один из наших наиболее популярных и плодовитых авторов. Ознакомьтесь со всеми статьями Яна (EN), опубликованными на сайте developerWorks. Вы можете найти контактные данные в профиле Яна и связаться с ним, а также с другими авторами и участниками ресурса My developerWorks.

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. Более подробно программные каналы и перенаправления рассматриваются в статье "*Потоки, программные каналы и перенаправления*" (см. перечень материалов для подготовки к экзаменам LPIC-1), а сейчас давайте рассмотрим программные каналы и простые перенаправления вывода с помощью операторов | и >.

Потоки

Поток – это всего лишь последовательность байтов, которые могут быть считаны или записаны с помощью библиотечных функций, скрывающих подробности реализации и

работы устройств от приложений. Одна и та же программа может считывать или отправлять данные на терминал, в файл или сетевое местоположение с помощью потоков независимо от используемого устройства. В современных средах программирования и командных интерпретаторах используются три стандартных потока:

- *stdin* – *стандартный поток ввода* (standard input stream), обеспечивающий ввод для команд.
- *stdout* – *стандартный поток вывода* (standard output stream), обеспечивающий отображение результатов выполнения команд.
- *stderr* – *стандартный поток ошибок* (standard error stream), обеспечивающий отображение ошибок, возникающих при выполнении команд.

Конвейеризация с использованием оператора |

Передаваемые командам параметры могут служить входными данными этих команд, а выходные данные могут выводиться на ваш терминал. Многие команды обработки текста (фильтры) могут получать входные данные либо из стандартного потока ввода, либо из файла. Чтобы передать вывод команды1 на вход команде2 (выступающей в качестве фильтра), необходимо соединить эти две команды оператором конвейеризации ввода/вывода (|). В листинге 1 показано, как перенаправить вывод команды `echo` на вход команде `sort`, которая сортирует полученный список слов.

Листинг 1. Передача вывода команды echo на вход команды sort

```
[ian@echidna ~]$ echo -e "apple\npear\nbanana"|sort
apple
banana
pear
```

У любой из этих команд могут иметься опции или аргументы. С помощью оператора | можно также перенаправить вывод второй команды на вход третьей команде и так далее. Построение длинных конвейеров из команд, каждая из которых имеет свой ограниченный функционал – это распространенный в Linux и UNIX прием, используемый для решения поставленных задач. Иногда аргументом команды может являться не имя файла, а знак дефиса (-); это означает, что входные данные следует принимать со стандартного устройства ввода, а не из файла.

Перенаправление вывода с помощью оператора >

Конечно, хорошо иметь возможность создавать конвейеры из нескольких команд и выводить результаты на экран терминала, однако иногда возникает необходимость сохранить вывод в файл. Для этого используется оператор перенаправления вывода (>).

В оставшихся примерах этой статьи мы будем использовать небольшие файлы, поэтому давайте создадим директорию с именем lpi103-2 и перейдем в нее. После этого давайте перенаправим с помощью оператора > вывод команды `echo` в файл с именем text1. Эти действия показаны в листинге 2. Заметьте, что поскольку весь вывод перенаправляется в файл, он не отображается на экране.

Листинг 2. Перенаправление вывода команды в файл

```
[ian@echidna ~]$ mkdir lpi103-2
[ian@echidna ~]$ cd lpi103-2
[ian@echidna lpi103-2]$ echo -e "1 apple\n2 pear\n3 banana" > text1
```

Теперь, когда у нас имеется несколько инструментов для конвейеризации и перенаправления, давайте рассмотрим несколько распространенных в Linux и UNIX команд обработки текста и фильтров. В этом разделе вы познакомитесь с некоторыми основными командами; для получения дополнительной информации о них обращайтесь к соответствующим man-страницам.

Команды cat, od и split

После того, как вы создали файл test1, вы можете просмотреть его содержимое. Для вывода содержимого файла на стандартное устройство вывода используется команда `cat` (сокращенно от *concatenate* – объединять). В листинге 3 на экран выводится содержимое только что созданного нами файла.

Листинг 3. Вывод содержимого файла с помощью команды cat

```
[ian@echidna lpi103-2]$ cat text1
1 apple
2 pear
3 banana
```

Если не указать имя файла (или поставить вместо имени файла дефис), то команда `cat` принимает входные данные со стандартного устройства ввода. Давайте используем эту возможность (а также перенаправление вывода) для создания еще одного текстового файла, как показано в листинге 4.

Листинг 4. Создание текстового файла с помощью команды cat

```
[ian@echidna lpi103-2]$ cat >text2
9       plum
3       banana
10      apple
```

Другие простые фильтры

Другим примером простого фильтра является команда `tac` (перевернутое имя команды `cat`). Эта команда выполняет действие команды `cat` наоборот – строки файла выводятся в обратном порядке. Попробуйте самостоятельно выполнить следующую команду:

```
tac text2 text1
```

В листинге 4 команда `cat` продолжает считывать данные с устройства stdin до тех пор, пока не будет достигнут конец файла. Чтобы обозначить конец файла, нажмите комбинацию клавиш **Ctrl-d** (удерживайте **Ctrl** и нажмите **d**). Эта же комбинация клавиш используется для выхода из командного интерпретатора bash. Используйте клавишу табуляции, чтобы выстроить названия фруктов в столбец.

Вы еще не забыли, что *cat* – это сокращение от *concatenate*? С помощью `cat` вы можете объединить несколько файлов и вывести их содержимое на экран. В листинге 5 показано содержимое двух файлов, которые мы создали.

Листинг 5. Объединение двух файлов с помощью команды cat

```
[ian@echidna lpi103-2]$ cat text*
1 apple
2 pear
3 banana
9       plum
3       banana
10      apple
```

Обратите внимание на различное выравнивание содержимого двух текстовых файлов при выводе их на экран с помощью `cat`. Чтобы разобраться, почему это происходит, необходимо посмотреть на управляющие символы, которые присутствуют в файлах. Эти символы влияют на вывод текста, но не имеют визуального отображения, поэтому необходимо создать *дамп* файла в формате, позволяющем увидеть и определить эти специальные символы. Для этих целей предназначена GNU-утилита `od` (*O*ctal *D*ump).

Команда `od` имеет несколько опций; например, опция `-A` управляет основанием смещения файлов, а опция `-t` – формой выводимого содержимого. Основание может быть указано как o (восьмеричное, используется по умолчанию), d (десятичное), x (шестнадцатеричное) или n (смещения не отображаются). Вы можете выводить содержимое в виде восьмеричных, шестнадцатеричных, десятичных значений, значений с плавающей точкой, ASCII-символов с escape-последовательностями или именованных символов (nl – для новой строки, ht – для горизонтальной табуляции и т. д.). В листинге 6 показано несколько доступных форматов дампа файла text2 из нашего примера.

Листинг 6. Дампы файлов, созданные с помощью команды od

```
[ian@echidna lpi103-2]$ od text2
0000000 004471 066160 066565 031412 061011 067141 067141 005141
0000020 030061 060411 070160 062554 000012
0000031
[ian@echidna lpi103-2]$ od -A d -t c text2
0000000   9  \t   p   l   u   m  \n   3  \t   b   a   n   a   n   a  \n
0000016   1   0  \t   a   p   p   l   e  \n
0000025
[ian@echidna lpi103-2]$ od -A n -t a text2
   9  ht   p   l   u   m  nl   3  ht   b   a   n   a   n   a  nl
   1   0  ht   a   p   p   l   e  nl
```

Примечания:

1. Опция `-A` команды `cat` также позволяет увидеть, где расположены символы табуляции и конца строки. Для получения дополнительной информации обратитесь к man-странице.
2. Если в вашем файле text2 вместо символов табуляции вы видите пробелы, то обратитесь к разделу Команды expand, unexpand и tr этой статьи, чтобы узнать, как переключаться между символами табуляции и пробелами.
3. Если вы имеет опыт работы с мэйнфреймами, то вам будет интересна утилита hexdump, которая входит в другой набор утилит и не рассматривается в этой статье.

> Если вы хотите больше узнать о hexdump, обратитесь к соответствующим man-страницам.

Файлы, используемые в наших примерах, очень малы, но иногда вам могут встретиться большие файлы, которые необходимо разделить на несколько более мелких. Например, вам может потребоваться разбить один большой файл на несколько частей такого размера, чтобы их можно было записать на компакт-диски. Для этого можно использовать команду `split`, которая разбивает файлы таким образом, что впоследствии их можно легко собрать обратно в единый файл при помощи команды `cat`. По умолчанию имена файлов, создаваемых командой `split`, состоят из префикса 'x', за которым следует суффикс 'aa', 'ab', 'ac', ..., 'ba', 'bb' и так далее. Эти умолчания можно изменить с помощью различных опций. Вы также можете задавать размер выходных файлов и определять, будут ли они содержать определенное количество строк или просто иметь определенный размер в байтах.

В листинге 7 показано разделение наших двух текстовых файлов с использованием различных префиксов для выходных файлов. Файл text1 мы разделили на файлы, содержащие максимум две строки, а файл text2 – на файлы максимальным размером в 18 байтов. Далее с помощью команды `cat` мы отобразили некоторые отдельные части, а также весь файл целиком, используя *универсализацию файловых имен*, которая рассматривается в статье "*Основы управления файлами и директориями*" (см. перечень материалов для подготовки к экзаменам LPIC-1).

Листинг 7. Разделение и восстановление файлов с помощью команд split и cat

```
[ian@echidna lpi103-2]$ split -l 2 text1
[ian@echidna lpi103-2]$ split -b 17 text2 y
[ian@echidna lpi103-2]$ cat yaa
9       plum
3       banana
1[ian@echidna lpi103-2]$ cat yab
0       apple
[ian@echidna lpi103-2]$ cat y* x*
9       plum
3       banana
10      apple
1 apple
2 pear
3 banana
```

Обратите внимание на то, что файл с именем yaa не оканчивается символом новой строки, поэтому, когда мы вывели его содержимое на экран с помощью команды `cat`, наше приглашение сдвинулось вправо.

Команды wc, head и tail

Команда `cat` выводит полное содержимое файла. Это хорошо подходит для небольших файлов (например, для наших примеров), но что делать, если объем файла очень большой? Итак, для начала можно оценить размер файла с помощью команды `wc` (*W*ord *C*ount – подсчет слов). Команда `wc` выводит количество содержащихся в файле строк и слов, а также размер файла в байтах, определить который можно также с помощью команды `ls -l`. В

листинге 8 показан подробный вывод сведений о наших двух текстовых файлах, а также вывод команды `wc`.

Листинг 8. Использование команды wc для работы с текстовыми файлами

```
[ian@echidna lpi103-2]$ ls -l text*
-rw-rw-r--. 1 ian ian 24 2009-08-11 14:02 text1
-rw-rw-r--. 1 ian ian 25 2009-08-11 14:27 text2
[ian@echidna lpi103-2]$ wc text*
 3  6 24 text1
 3  6 25 text2
 6 12 49 total
```

Различные опции позволяют вам управлять выводом команды `wc` или отображать другую информацию, например, максимальную длину строки. Для получения дополнительной информации обратитесь к man-странице.

Две другие команды позволяют отображать либо первую часть файла (*заголовок*), либо последнюю (*хвост*). Эти команды так и называются – `head` и `tail` соответственно. Их можно использовать в качестве фильтров или передавать им в качестве аргумента имя файла. По умолчанию эти команды отображают 10 первых (или последних) строк файла или потока. В листинге 9 совместно используются команды `dmesg` (отображение информации о загрузке системы), `wc`, `tail` и `head`; в результате мы видим, что файл содержит 791 сообщение, выводим последние 10 из них, а затем выводим шесть сообщений, начиная с 15 от конца.

Листинг 9. Использование команд wc, head и tail для вывода сообщений о загрузке

```
[ian@echidna lpi103-2]$ dmesg|wc
    791    5554   40186
[ian@echidna lpi103-2]$ dmesg | tail
input: HID 04b3:310b as /devices/pci0000:00/0000:00:1a.0/usb3/3-2/3-2.4/3-2.4:1.0/input/i
nput12
generic-usb 0003:04B3:310B.0009: input,hidraw1: USB HID v1.00 Mouse [HID 04b3:310b] on us
b-0000:00:1a.0-2.4/input0
usb 3-2.4: USB disconnect, address 11
usb 3-2.4: new low speed USB device using uhci_hcd and address 12
usb 3-2.4: New USB device found, idVendor=04b3, idProduct=310b
usb 3-2.4: New USB device strings: Mfr=0, Product=0, SerialNumber=0
usb 3-2.4: configuration #1 chosen from 1 choice
input: HID 04b3:310b as /devices/pci0000:00/0000:00:1a.0/usb3/3-2/3-2.4/3-2.4:1.0/input/i
nput13
generic-usb 0003:04B3:310B.000A: input,hidraw1: USB HID v1.00 Mouse [HID 04b3:310b] on us
b-0000:00:1a.0-2.4/input0
usb 3-2.4: USB disconnect, address 12
[ian@echidna lpi103-2]$ dmesg | tail -n15 | head -n 6
usb 3-2.4: USB disconnect, address 10
usb 3-2.4: new low speed USB device using uhci_hcd and address 11
usb 3-2.4: New USB device found, idVendor=04b3, idProduct=310b
usb 3-2.4: New USB device strings: Mfr=0, Product=0, SerialNumber=0
usb 3-2.4: configuration #1 chosen from 1 choice
input: HID 04b3:310b as /devices/pci0000:00/0000:00:1a.0/usb3/3-2/3-2.4/3-2.4:1.0/input/i
nput12
```

Другим распространенным применением команды `tail` является *слежение* за файлом; для этого используется опция `-f` и шаг, обычно равный одной строке. Это может оказаться

полезным в том случае, если у вас имеется фоновый процесс, генерирующий вывод данных в файл, и вы хотите следить за ходом его выполнения. В этом режиме команда `tail` будет работать и выводить строки по мере их добавления в файл до тех пор, пока вы не завершите ее работу, нажав **Ctrl-c**.

Команды expand, unexpand и tr

Когда мы создавали наши файлы text1 и text2, в последнем из них были использованы символы табуляции. Иногда вам может потребоваться заменить символы табуляции на символы пробела и наоборот. Для этого предназначены команды `expand` и `unexpand`. В обеих командах опция `-t` позволяет устанавливать позиции табуляции. Если после этой опции указывается только одно значение, то позиции табуляции будут периодически расставляться через этот указанный интервал. В листинге 10 показано, как сократить символы табуляции в файле text2 до одиночных пробелов, а также приведена причудливая последовательность из команд `expand` и `unexpand` и нарушающая выравнивание текста в файле text2.

Листинг 10. Использование команд expand и unexpand

```
[ian@echidna lpi103-2]$ expand -t 1 text2
9 plum
3 banana
10 apple
[ian@echidna lpi103-2]$ expand -t8 text2|unexpand -a -t2|expand -t3
9           plum
3           banana
10       apple
```

К сожалению, вы не можете использовать команду `unexpand` для замены пробелов в файле text1 на символы табуляции, поскольку для преобразования в символ табуляции команде `unexpand` требуется, как минимум, два последовательных пробела. Однако вы можете использовать команду `tr`, которая преобразует символы из одного набора (*набор1*) в соответствующие символы из другого набора (*набор2*). В листинге 11 показан пример использования команды `tr` для преобразования пробелов в символы табуляции. Поскольку команда `tr` представляет собой фильтр в чистом виде, то входные данные для нее генерируются с помощью команды `cat`. В этом примере также показан пример использования дефиса (-) с целью указать `cat` на то, что ввод будет осуществляться со стандартного устройства; таким образом, мы можем объединить вывод команды `tr` с содержимым файла text2.

Листинг 11. Использование команды tr

```
[ian@echidna lpi103-2]$ cat text1 |tr ' ' '\t'|cat - text2
1       apple
2       pear
3       banana
9       plum
3       banana
10      apple
```

Если вы не очень понимаете, что происходит в последних двух примерах, то попробуйте использовать команду `od`, чтобы последовательно выполнить каждую команду конвейера, например:

```
cat text1 |tr ' ' '\t' | od -tc
```

Команды Pr, nl и fmt

Команда `pr` используется для форматирования файлов перед печатью. По умолчанию заголовок включает в себя имя файла, дату и время создания файла, номер страницы, а также две пустых строки нижнего колонтитула. Когда данные поступают из нескольких файлов или со стандартного устройства ввода, вместо даты и времени создания файла используются текущие дата и время. Можно печатать файлы рядом, каждый в своем столбце, а также управлять многими возможностями форматирования с помощью различных опций. Как обычно, дополнительную информацию вы можете найти на man-странице.

Команда `nl` нумерует строки, что может оказаться полезным при печати файлов. Для нумерации строк можно также использовать команду `cat` с опцией `-n`. В листинге 12 показано, как распечатать наш текстовый файл, пронумеровать строки в файле text2 и вывести его на печать вместе с файлом text1.

Листинг 12. Нумерация строк и форматирование перед печатью

```
[ian@echidna lpi103-2]$ pr text1 | head


2009-08-11 14:02                     text1                      Page 1


1 apple
2 pear
3 banana


[ian@echidna lpi103-2]$ nl text2 | pr -m - text1 | head


2009-08-11 15:36                                                Page 1


     1  9       plum                        1 apple
     2  3       banana                      2 pear
     3  10      apple                       3 banana
```

Другой полезной командой для форматирования текста является команда `fmt`, которая форматирует текст таким образом, чтобы он не выходил за границы полей. Можно объединить несколько коротких строк в одну длинную и наоборот. В листинге 13 мы создали файл text3, используя одну длинную конструкцию из комбинаций символов `! #:*` (предназначенных для управления историей команд), благодаря чему, напечатанное предложение было сохранено в файле четыре раза. Также мы создали файл text4, содержащий по одному слову в каждой строке. Затем с помощью команды `cat` мы отобразили содержимое этих файлов в неформатированном виде, включая символ конца строки '$'. Наконец, с помощью команды `fmt` мы отформатировали эти файлы, ограничив максимальное значение длины строки 60 символами. Как обычно, дополнительную информацию вы можете найти на man-странице.

Листинг 13. Форматирование с указанием максимальной длины строки

```
[ian@echidna lpi103-2]$ echo "This is a sentence. " !#:* !#:1->text3
echo "This is a sentence. " "This is a sentence. " "This is a sentence. ">text3
[ian@echidna lpi103-2]$ echo -e "This\nis\nanother\nsentence.">text4
[ian@echidna lpi103-2]$ cat -et text3 text4
This is a sentence.  This is a sentence.  This is a sentence. $
This$
is$
another$
sentence.$
[ian@echidna lpi103-2]$ fmt -w 60 text3 text4
This is a sentence.  This is a sentence.  This is a
sentence.
This is another sentence.
```

Команды sort и uniq

Команда `sort` сортирует входные данные с использованием схемы упорядочивания локали (LC_COLLATE) системы. Команда `sort` также может объединять уже отсортированные файлы и определять, является ли файл отсортированным или нет.

В листинге 14 приведены примеры использования команды `sort` для сортировки двух текстовых файлов после замены в файле text1 пробелов на символы табуляции. Поскольку сортировка выполняется на основе символьных значений, вы можете удивиться, увидев результаты. К счастью, команда `sort` может выполнять сортировку не только на основе символьных, но также и на основе числовых значений. Вы можете указать требуемый метод сортировки для всей записи или для каждого *поля*. Если вы не указываете разделитель полей, то используются пробелы или символы табуляции. Во втором примере листинга 14 сортировка первого поля выполняется по числовым значениям, а сортировка второго поля – с использованием схемы упорядочивания (в алфавитном порядке). Также показан пример использования опции `-u` для удаления повторяющихся строк.

Листинг 14. Сортировка по символьным и числовым значениям

```
[ian@echidna lpi103-2]$ cat text1 | tr ' ' '\t' | sort - text2
10      apple
1       apple
2       pear
3       banana
3       banana
9       plum
[ian@echidna lpi103-2]$ cat text1|tr ' ' '\t'|sort -u -k1n -k2 - text2
1       apple
2       pear
3       banana
9       plum
10      apple
```

Заметьте, что в списке все равно присутствуют две строки со словом "apple", поскольку проверка уникальности выполнялась по всем ключам сортировки (в нашем случае это k1n и k2). Подумайте, какие команды нужно изменить или добавить в конвейер в последнем примере, чтобы исключить дублирование слова 'apple'.

Можно управлять удалением повторяющихся строк с помощью другой команды – `uniq`. В обычном режиме команда `uniq` работает с отсортированными файлами и удаляет

последовательные повторяющиеся строки из любого файла независимо от того, отсортирован он или нет. Также эта команда может игнорировать заданные поля. В листинге 15 выполняется сортировка наших двух текстовых файлов по второму полю (имя фрукта), после чего удаляются строки, в которых повторяются значения второго поля (т. е. при проверке мы не обращаем внимания на первое поле).

Листинг 15. Использование команды uniq

```
[ian@echidna lpi103-2]$ cat text1|tr ' ' '\t'|sort -k2 - text2|uniq -f1
10      apple
3       banana
2       pear
9       plum
```

В этом примере сортировка выполнялась с использованием схемы упорядочивания, поэтому команда `uniq` оставила запись "10 apple", а не "1 apple". Вы можете добавить сортировку первого поля по числовым значениями и посмотреть, что изменится в этом случае.

Команды cut, paste и join

Давайте рассмотрим еще три команды, которые работают с полями в текстовых данных. Эти команды особенно полезны при работе с табличными данными. Первая команда `cut` извлекает поля из текстовых файлов. Символом-разделителем по умолчанию является символ табуляции. В листинге 16 содержится пример, в котором команда `cut` используется для разделения двух столбцов файла text2, а затем в качестве разделителя выходных данных используется пробел, что является необычным способом преобразования символов табуляции в пробелы.

Листинг 16. Использование команды cut

```
[ian@echidna lpi103-2]$ cut -f1-2 --output-delimiter=' ' text2
9 plum
3 banana
10 apple
```

Команда `paste` вставляет (склеивает) строки из двух или более файлов, размещая их рядом (подобно тому, как команда `pr` объединяет файлы с помощью опции `-m`). В листинге 17 показан результат применения этой команды к нашим текстовым файлам.

Листинг 17. Склеивание файлов

```
[ian@echidna lpi103-2]$ paste text1 text2
1 apple 9       plum
2 pear  3       banana
3 banana        10      apple
```

В этом примере показана простейшая операция, тем не менее, команда `paste` может вставлять данные из одного или нескольких файлов различными способами. Для получения дополнительной информации обратитесь к man-странице.

Последняя команда для управления полями – это команда `join`, которая объединяет файлы на основе совпадения полей. Файл должен быть отсортирован по объединяемому полю.

Поскольку файл text2 не отсортирован по числовым значениям, то можно отсортировать его, а затем объединить с помощью команды join две строки с одинаковым значением поля, по которому выполняется объединение (в нашем примере это первое поле, содержащее значение 3).

Листинг 18. Объединение файлов по совпадающим полям

```
[ian@echidna lpi103-2]$ sort -n text2|join -j 1 text1 -
3 banana banana
join: file 2 is not in sorted order
```

Что же здесь пошло не так? Вспомните материал раздела Команды sort и uniq, в котором говорилось о сортировке на основе числовых и символьных значений. Объединение выполняется по совпадающим символам в соответствии со схемой упорядочивания locale. Объединение не будет выполняться для числовых полей до тех пор, пока все поля не будут иметь одинаковую длину.

Мы использовали опцию `-j 1` для объединения по первому полю в каждом файле. Для каждого файла можно указать отдельное поле, по которому будет выполняться объединение. Например, можно объединить поле 3 в одном файле с полем 10 другого файла.

Давайте создадим еще один файл, text5, выполнив сортировку файла text1 по второму полю (имя фрукта), а затем заменив пробелы на символы табуляции. Если теперь мы отсортируем файл text2 по второму полю и объединим его с файлом text5 по этому же полю, то получим два совпадения (apple и banana). Это объединение показано в листинге 19.

Листинг 19. Объединение файлов по совпадающим полям

```
[ian@echidna lpi103-2]$ sort -k2 text1|tr ' ' '\t'>text5
[ian@echidna lpi103-2]$ sort -k2 text2 | join -1 2 -2 2 text5 -
apple 1 10
banana 3 3
```

Редактор Sed

Sed (*s*tream *ed*itor) – это потоковый редактор. Ему посвящено несколько статей Web-сайта developerWorks, а также множество книг (см. раздел Ресурсы). Sed является чрезвычайно мощным инструментом, а круг решаемых им задач ограничен лишь вашим воображением. Этот небольшой обзор должен пробудить ваш интерес к sed, хотя он не является полным и всесторонним.

Как и многие команды для работы с текстом, которые мы здесь рассмотрели, sed может работать как фильтр или принимать входные данные из файла. Вывод осуществляется на стандартное устройство вывода. Sed загружает строки из входных данных в *область шаблонов*, применяет к ее содержимому команды редактирования и передает ее на стандартное устройство вывода. Sed может объединять в области шаблонов несколько строк; результат может быть записан в файл, может быть записан частично, а может быть не записан вообще.

Для поиска и выборочной замены текста в области шаблонов, а также для определения строк, над которыми необходимо выполнять те или иные команды редактирования, sed использует синтаксис регулярных выражений. Более подробно о регулярных выражениях рассказывается в статье "*Поиск в текстовых файлах с помощью регулярных выражений*" (см. перечень материалов для подготовки к экзаменам LPIC-1). Временным хранилищем текста служит *буфер удержания*. Буфер удержания может заместить собой область шаблонов, может быть добавлен к области шаблонов, а может обмениваться с ней данными. Хотя в sed имеется ограниченное число команд, их использование совместно с регулярными выражениями и буфером удержания открывает безграничные возможности. Набор команд sed обычно называется *сценарием sed*.

В листинге 20 показаны три простых сценария sed. В первом сценарии используется команда `s` (substitute – замена) для замены в каждой строке символа 'a' в нижнем регистре на этот же символ в верхнем регистре. В первом примере выполняется замена только первого символа 'a', поэтому во втором примере мы добавили флаг 'g' (global – глобальный), благодаря которому, будет выполняться замена всех найденных вхождений этого символа. В третьем сценарии мы используем команду `d` (delete – удалить) для удаления строки. В нашем примере мы использовали адрес 2, чтобы показать, что необходимо удалить только строку с этим номером. Мы разделяем команды точкой с запятой (;) и используем глобальную замену символов 'a' на 'A', как это было сделано во втором примере.

Листинг 20. Первые шаги по работе со сценариями sed

```
[ian@echidna lpi103-2]$ sed 's/a/A/' text1
1 Apple
2 peAr
3 bAnana
[ian@echidna lpi103-2]$ sed 's/a/A/g' text1
1 Apple
2 peAr
3 bAnAnA
[ian@echidna lpi103-2]$ sed '2d;$s/a/A/g' text1
1 apple
3 bAnAnA
```

Помимо работы с отдельными строками, sed может работать с диапазонами строк. Начало и конец диапазона разделяются запятой (,) и могут определяться в виде номера строки, регулярного выражения или знака доллара ($), означающего конец файла. Зная адрес или диапазон адресов, вы можете сгруппировать несколько команд, заключив их в фигурные скобки { и }; таким образом, эти команды будут работать только с теми строками, которые указаны в диапазоне. В листинге 21 показано два примера глобальной замены, которая применяется только к последним двум строкам нашего файла. Также приведен пример использования опции `-e` для добавления нескольких команд в сценарий.

Листинг 21. Адреса в sed

```
[ian@echidna lpi103-2]$ sed -e '2,${' -e 's/a/A/g' -e '}' text1
1 apple
2 peAr
3 bAnAnA
[ian@echidna lpi103-2]$ sed -e '/pear/,/bana/{' -e 's/a/A/g' -e '}' text1
1 apple
2 peAr
3 bAnAnA
```

Сценарии sed можно сохранять в виде файлов. Скорее всего, вы захотите использовать эту возможность для наиболее часто используемых сценариев. Вспомните команду `tr`, которую мы использовали для изменения пробелов в файле text1 на символы табуляции. Давайте теперь сделаем то же самое с помощью сценария sed, сохраненного в файле. Для создания файла мы используем команду `echo`. Результаты представлены в листинге 22.

Листинг 22. Короткая программа sed

```
[ian@echidna lpi103-2]$ echo -e "s/ /\t/g">sedtab
[ian@echidna lpi103-2]$ cat sedtab
s/ /    /g
[ian@echidna lpi103-2]$ sed -f sedtab text1
1       apple
2       pear
3       banana
```

Существует множество подобных коротких сценариев; ссылки на некоторые из них вы можете найти в разделе Ресурсы.

В нашем последнем примере сначала используется команда `=` для вывода номеров строк, а затем выполняется фильтрация полученного вывода с помощью sed (в результате мы получим такой же эффект, как от использования команды `nl` для нумерации строк). В листинге 23 с помощью команды `=` выводятся номера строк, затем с помощью команды `N` вторая строка ввода считывается в область шаблонов и, наконец, между двумя строками в области шаблонов удаляется символ новой строки (/n).

Листинг 23. Нумерация строк с помощью sed

```
[ian@echidna lpi103-2]$ sed '=' text2
1
9       plum
2
3       banana
3
10      apple
[ian@echidna lpi103-2]$ sed '=' text2|sed 'N;s/\n//'
19      plum
23      banana
310     apple
```

Не совсем то, что мы хотели получить! Вообще-то, мы ожидали получить выровненный столбец с номерами строк, после которых следуют сами строки файла, отделенные несколькими пробелами. В листинге 24 мы вводим несколько строк с командами (обратите

внимание на дополнительное приглашение >). Изучите этот пример и прочитайте его объяснение ниже.

Листинг 24. Нумерация строк с помощью sed, второй подход

```
[ian@echidna lpi103-2]$ cat text1 text2 text1 text2>text6
[ian@echidna lpi103-2]$ ht=$(echo -en "\t")
[ian@echidna lpi103-2]$ sed '=' text6|sed "N
> s/^/      /
> s/^.*\(......\)\n/\1$ht/"
     1  1 apple
     2  2 pear
     3  3 banana
     4  9       plum
     5  3       banana
     6  10      apple
     7  1 apple
     8  2 pear
     9  3 banana
    10  9       plum
    11  3       banana
    12  10      apple
```

Вот, что было сделано в этом примере:

1. Сначала с помощью команды `cat` мы создали файл, содержащий 12 строк из двух копий файлов text1 и text2 (если бы количество строк оказалось меньше 10, т. е. все числа были бы одного порядка, то не было бы никакого смысла в их форматировании).
2. В командном интерпретаторе bash клавиша табуляции используется для завершения команды, поэтому удобно иметь заранее определенный символ табуляции для использования в тех случаях, когда он вам потребуется. Для этого с помощью команды `echo` мы сохранили символ табуляции в переменной окружения 'ht'.
3. Мы создали поток, содержащий номера строк и следующие за ними строки с данными, как мы делали это раньше, и отфильтровали его с помощью второй копии sed.
4. Мы считали вторую строку в область шаблонов.
5. В начале области шаблонов (обозначаемой символом ^) мы добавили к номеру строки префикс, состоящий из шести пробелов.
6. Наконец, мы применили выравнивание с помощью разделителя, содержащего последние шесть символов и символ табуляции, и теперь в номера строк в выводе будут выровнены по первым шести символам. Обратите внимание на то, что в левой части команды 's' используются конструкции '\(' и '\)' для отметки символов, которые мы хотим использовать в правой части. В правой части мы ссылаемся на первый (и единственный в нашем примере) такой набор символов с помощью /1. Заметьте, что наша команда заключена в двойные кавычки ("), поэтому эта подстановка будет выполнена для переменной $ht.

Последняя (четвертая) версия редактора sed содержит документацию в формате `info` и включает множество превосходных примеров. В более старой версии, 3.02, эти возможности отсутствуют. Узнать версию редактора GNU sed можно с помощью команды `sed --version`.

Ресурсы

- Участвуйте в обсуждении данного материала на форуме.
- Оригинал статьи: Learn Linux, 101: Text streams and filters (EN).
- На Web-сайте программы сертификации LPIC (EN) вы найдете подробные цели, списки задач и примерные вопросы всех трех уровней сертификации на администратора Linux-систем профессионального института Linux. В частности, на этом сайте представлены цели экзаменов LPI 101 и LPI 102 по состоянию на апрель 2009 года. Всегда обращайтесь к Web-сайту программы сертификации LPIC, чтобы узнать последние цели.
- Просмотрите всю серию статей для подготовки к экзаменам института LPI (EN) на сайте developerWorks, основанных на предыдущих целях, определенных до апреля 2009 года, чтобы изучить основы администрирования Linux и подготовиться к экзаменам для получения сертификата администратора Linux.
- Прочитайте статью "Основные задачи для начинающих Linux-разработчиков" (EN) (developerWorks, март 2005 г.) и узнайте, как открыть окно терминала или командной оболочки, а также о многом другом.
- Web-сайт Linux Documentation Project (EN) содержит большое количество полезной документации, в особенности, HOWTO-руководств.
- Первая, вторая и третья части серии *Sed by example* (EN) – это отличный способ повысить свои навыки работы с sed.

Об авторе

Ян Шилдс

No bio.