11

Команда find

В предыдущей главе мы рассматривали команды, позволяющие выполнять поиск файлов (`locate`) и данных внутри файлов (`grep`). Третья команда из той же группы — `find`. Программа `locate` выполняет поиск в базе данных имен файлов, что позволяет ей быстро работать, но ставит в зависимость от времени обновления базы, а команда `find` непосредственно ищет файлы по заданным условиям. Поскольку команда `find` осуществляет реальный обход дерева каталогов, она работает намного медленнее, чем `locate`, однако предоставляет возможности, недоступные посредством команды `locate`.

В этой главе мы будем выполнять поиск на внешнем жестком диске, смонтированном в каталоге `/media/music`. Обратите внимание на пробелы в именах файлов, которые объясняют наличие двойных кавычек во многих командах. Вы увидите, что команда `find` позволяет обрабатывать самые разнообразные запросы.

Поиск файлов по имени

`find -name`

Чаще всего команда `find` используется для поиска файлов по имени или его части. Для этой цели предусмотрена опция `-name`. По умолчанию поиск осуществляется рекурсивно в дереве каталогов. Давайте найдем mp3-файлы с записями группы `Shaggs`.

```
$ cd /media/music
$ find . -name Shaggs
./Outsider/Shaggs
```

Что-то не так! Команда `find` нашла каталог, а не файлы, содержащиеся в нем. Почему это произошло? Дело в том, что мы не указали символы групповых операций, поэтому команда `find` искала файлы по точному совпадению имени со строкой `Shaggs`. Такое имя имеет только один файл — каталог, содержащий mp3-файлы (как вы, наверное, помните, каталог — это файл специального типа).

Нам надо использовать символ групповой операции, но? чтобы предотвратить специальную интерпретацию его оболочкой, поместим имя, содержащее этот символ, в кавычки. Попробуем выполнить поиск снова.

```
$ find . -name "*Shaggs*"
./Outsider/Shaggs
./Outsider/Shaggs/Gimme Dat Ting (Live).mp3
./Outsider/Shaggs/My Pal Foot Foot.mp3
./Outsider/Shaggs/I Love.mp3
./Outsider/Shaggs/You're Somethin' Special To Me.mp3
./Outsider/Shaggs/Things I Wonder.mp3
```

Теперь мы получили информацию не только о каталоге, но и о файлах.

ЗАМЕЧАНИЕ

Не подозревая о том, мы использовали еще одну опцию: `-print`. Эта опция сообщает программе `find`, что результаты поиска надо вывести на терминал. По умолчанию эта опция включена, и ее не надо указывать явно при вызове `find`.

Важно отметить еще одну особенность программы `find`: формат представления результатов зависит от пути, указанного при вызове команды. В предыдущем примере мы использовали относительный путь — указали текущий каталог, поэтому результаты также были представлены относительно текущего каталога. Что произойдет, если мы зададим абсолютный путь, начинающийся с символа `/`?

```
$ find / -name "*Shaggs*"
/media/music/Outsider/Shaggs
/media/music/Outsider/Shaggs/Gimme Dat Ting
↳ (Live).mp3
/media/music/Outsider/Shaggs/My Pal Foot Foot.mp3
/media/music/Outsider/Shaggs/I Love.mp3
/media/music/Outsider/Shaggs/You're Somethin'
↳ Special To Me.mp3
/media/music/Outsider/Shaggs/Things I Wonder.mp3
```

При поиске по абсолютному пути результаты также представляются посредством абсолютного пути. Другие применения этого принципа мы увидим далее в данной главе. Сейчас же просто запомним эту особенность.

ЗАМЕЧАНИЕ

Более подробную информацию о группе Shaggs см. по адресам `http://www.allmusic.com/artist/the-shaggs-mn0000418794` и `http://en.wikipedia.org/wiki/The_Shaggs`.

Поиск файлов по имени владельца

```
find -user
find -group
```

Помимо поиска по имени файла, можно также осуществлять поиск по имени владельца. Предположим, что вы хотите найти файлы, принадлежащие пользователю `scott`. Задайте при вызове команды `find` опцию `-user`, введите после нее имя пользователя (или числовой идентификатор, который можно найти в файле `/etc/passwd`).

```
$ find . -user scott
```

Количество результатов слишком велико! Возможно, будет проще искать файлы, которые *не* принадлежат пользователю

scott. Сделать это можно, указав символ ! перед опцией, действие которой вы хотите изменить на обратное.

```
$ find . ! -user scott
./Punk/Stooges/Fun House/01 Down on the Street.mp3
$ ls -l "Punk/Stooges/Fun House/01 Down on the
↳Street.mp3"
gus music ./Punk/Stooges/Fun House/01 Down on the
↳Street.mp3
```

Один из файлов с записью композиций *The Stooges* принадлежит пользователю gus, а не scott. Помните, что при необходимости вы всегда можете использовать символ ! в качестве оператора отрицания. Так, только что мы указали команде find найти файлы, *не* принадлежащие пользователю scott.

Если вы хотите найти файлы, принадлежащие определенной группе, то вам необходимо указать опцию -group и ввести после нее имя или номер группы. На диске, используемом при поиске, владельцем файлов является пользователь scott, а в качестве группы указана music. Попробуем найти файлы, не принадлежащие группе music.

```
$ find . ! -group music
./Disco/Brides of Funkenstein/Disco to Go.mp3
./Disco/Sister Sledge/He's The Greatest Dancer.mp3
./Disco/Wild Cherry/Play_That_Funky_Music.mp3
./Electronica/New Order/Bizarre Love Triangle.mp3
```

Из огромного числа файлов только четыре не принадлежат данной группе. Заметьте, что мы снова использовали символ !, чтобы выполнить поиск файлов, не принадлежащих группе music.

Поиск файлов по размеру

file -size

В некоторых случаях приходится использовать в качестве критерия для поиска файлов их размер. Команда find

позволяет решить и эту задачу. Для того чтобы указать размер, надо задать опцию `-size` и ввести после нее букву, которая представляет используемую вами схему размера. Если вы не зададите букву, будет использовано значение по умолчанию, но следует иметь в виду, что оно не обязательно будет соответствовать вашим намерениям. По умолчанию, если после числа вы не введете букву, принимается значение размера в байтах, деленное на 512 и округленное вверх до ближайшего целого числа. Некоторым пользователям покажется, что здесь слишком много математики и проще указать суффикс, представляющий единицу изменения размера. Набор допустимых суффиксов представлен в табл. 10.1.

Таблица 10.1. Суффиксы, используемые при поиске файлов по размеру

Суффикс	Значение
`b`	512-байтовые блоки (по умолчанию)
`c`	Байты
`k`	Килобайты
`M`	Мегабайты
`G`	Гигабайты

ЗАМЕЧАНИЕ

Строго говоря, несмотря на то, что на страницах справочной системы `man`, посвященных команде `find`, используются термины килобайт, мегабайт и гигабайт, эти термины неправильные (см. главу 6). Множество программ по-прежнему используют старые термины, которые в настоящее время стали некорректными. За подробной информацией обращайтесь по адресу `http://en.wikipedia.org/wiki/Mebibyte`.

Предположим, нам надо найти файлы с записью песен группы “Clash” из знаменитого альбома “London Calling”, причем размер файлов должен составлять 10 Мбайт (конечно же,

использовалось кодирование, обеспечивающее наивысшее качество). Эта задача легко решается с помощью команды `find`.

```
$ cd Punk/Clash/1979_London_Calling
$ find . -size 10M
./07 - The Right Profile.mp3
./08 - Lost In The Supermarket.mp3
./09 - Clampdown.mp3
./12 - Death Or Glory.mp3
```

Результаты выглядят странно. Только четыре файла? Чтобы понять причины такого поведения команды `find`, надо учесть следующую ее особенность: если вы задаете значение 10 Мбайт, команда `find` ищет файлы, размер которых в точности равен 10 Мбайт (конечно же, с учетом округления). Если вам нужны файлы размером больше 10 Мбайт, то надо перед значением размера ввести символ +, если же размеры файлов должны быть меньше 10 Мбайт — то знак –.

```
$ find . -size +10M
./03 - Jimmy Jazz.mp3
./15 - Lover's Rock.mp3
./18 - Revolution Rock.mp3
```

И снова возникла проблема. Когда мы указали размер для поиска, равный 10 Мбайт, были найдены файлы, размер которых равен этому значению, но не больше его, а когда задали +10M, получили список файлов, размер которых превышает 10 Мбайт, но файлы, размер которых в точности равен 10 Мбайт, не были учтены. А как получить сведения об обоих наборах файлов? Ответ на этот вопрос вы найдете ниже в данной главе.

ПОДСКАЗКА

Если вы хотите отыскать большие текстовые файлы, используйте после числового значения букву c. Как показано в табл. 10.1, этот символ задает измерение размера в байтах. Каждый символ в текстовом файле занимает один байт, поэтому ясно, что символ c — это первая буква в слове characters.

Например, чтобы найти очень большой текстовый файл, можно использовать выражение

```
$ find /home/scott/documents -size +500000c
```

Поиск файлов по типу

find -type

Одна из самых полезных опций программы `find` — это `-type`, позволяющая указать тип объекта для поиска. В главе 1 было сказано, что все объекты в системе Unix являются файлами, поэтому если вы укажете программе `find` тип файла, то она найдет для вас нужный объект. В табл. 11.2 перечислены типы файлов, поддерживаемые `find`.

Таблица 11.2. Обозначения при поиске файлов по типу

Буква, представляющая тип файла	Значение
f	Обычный файл
d	Каталог
l	Символьная ссылка
b	Специальный файл блочного типа
c	Специальный файл символьного типа
p	FIFO
s	Сокет

Предположим, нам надо получить список разных вариаций легендарной песни Фрэнка Синатры “Come Fly With Me”, которые записаны на музыкальном диске. Для этого надо выполнить ряд команд (результаты приводятся в сокращенном виде — на самом деле их 14).

```
$ cd "Jazz - Vocal/Frank Sinatra"
$ find . -name "*Come Fly With Me*"
./1962 Live In Paris/26 - Come Fly With Me.mp3
./1957 Come Fly With Me
```

```
./1957 Come Fly With Me/01 - Come Fly With Me.mp3
./1966 Sinatra At The Sands/01 - Come Fly With. Me.mp3
```

Обратите внимание на второй результат. Это каталог, в котором хранится знаменитый альбом Фрэнка Синатры 1957 года под таким же названием: "Come Fly With Me". Но нам нужны только файлы, а не каталоги, поэтому с помощью опции `-type f` ограничим результаты только файлами.

```
$ find . -name "*Come Fly With Me*" -type f
./1962 Live In Paris/26 - Come Fly With Me.mp3
./1957 Come Fly With Me/01 - Come Fly With Me.mp3
./1966 Sinatra At The Sands/01 - Come Fly With. Me .mp3
```

Этот список удобен, но поскольку имя каждого альбома начинается с года его выпуска, его можно передать команде `sort` (см. главу 7) и проследить за изменениями, которые Синатра постепенно вносил в эту песню.

```
$ find . -name "*Come Fly With Me*" -type f | sort
./1957 Come Fly With Me/01 - Come Fly With Me.mp3
./1962 Live In Paris/26 - Come Fly With Me.mp3
./1966 Sinatra At The Sands/01 - Come Fly With. Me.mp3
```

Передача результатов команды `find` для последующей фильтрации может оказаться очень полезной; по мере освоения команды `find` вы научитесь создавать все более сложные фильтры.

Поиск файлов по времени

```
find -amin|-cmin|-mmin
find -atime|-ctime|-mtime
find -anewer|-cnewer|-newer|-newerXY
```

До сих пор мы вели речь об использовании команды `find` для создания списка файлов по именам, владельцам, размерам и типам. А как насчет времени? Команда `find` решает эту задачу блестяще.

Например, однажды я захотел выявить все файлы в каталоге и его подкаталогах, созданные не менее чем четыре часа назад. С помощью команды `find` это оказалось очень просто.

```
$ find . -mmin +240
```

Эта команда означает: "Создать список всех файлов, модифицированных более чем 240 минут назад". Когда вы используете команду `find` таким образом, вы должны задать число (в большинстве случаев). Это число можно выразить тремя способами, показанными в табл. 11.3.

Таблица 11.3. Числовые аргументы для поиска файлов по времени

Numeric Argument	Meaning
`+n`	Больше, чем `n`
`-n`	Меньше, чем `n`
`n`	Равно `n`

Следует помнить, что число `n` может быть количеством минут или часов в зависимости от конкретного случая. Кроме того, команда `find` может проверять, когда был последний *доступ* к файлу (accessed), когда он был *изменен* (changed) и когда *модифицирован* (modified). Несмотря на то что эти термины могут показаться синонимами, с точки зрения системы Linux в них вкладывается разный смысл.

- **Доступ к файлу** означает, что его содержимое читали, но не изменяли, например, с помощью команды `less`.
- **Изменение файла** означает изменение метаданных (или статуса файла), но не его содержимого, например, с помощью команд `chmod`, `chown`, `link` и `rename`.
- **Модификация файла** означает, что данные редактировали.

ПОДСКАЗКА

Вы можете проверить эту информацию с помощью очень полезной команды `stat` (см. пример):

```
$ stat foobar.txt
Access: 2013-04-14 16:57:24.768011000 -0500
Modify: 2012-12-01 22:27:24.424000023 -0600
Change: 2012-12-01 22:27:24.424000023 -0600
```

Итак, зная разницу между доступом, изменением и модификацией, мы можем, наконец, выполнить полную проверку файлов. Они сгруппированы логически в табл. 11.4. Напоминаю, что параметр `n` в табл. 11.4 может означать любое число из табл.11.3.

Таблица 11.4. Поиск файлов по времени

Тест	Описание
Минуты	
`-amin n`	Был доступ `n` минут назад
`-cmin n`	Статус был изменен `n` минут назад
`-mmin n`	Данные были изменены `n` минут назад
Часы (дробные части суток игнорируются)	
`-atime n`	Был доступ `n*24` минут назад
`-ctime n`	Статус был изменен `n*24` минут назад
`-mtime n`	Данные были изменены `n*24` минут назад

ЗАМЕЧАНИЕ

Может возникнуть вопрос: почему в качестве опции поиска не используется время создания файла? Причина этого очевидна: ядро системы Linux не отслеживает время создания файлов.

Хотите найти файлы, доступ к которым был выполнен более 45 минут назад? Используйте опцию `-amin +45`.

Хотите найти файлы, статус которых был изменен в течение последних 24 часов? Используйте опцию `-ctime 0`. Помните, что команда `find` отбрасывает дробные доли суток, поэтому, если вам нужно найти файлы, которые изменялись в

течение последних суток, используйте параметр `0` (это вызывает замешательство, я знаю).

А как найти файлы, которые были модифицированы, например, от двух до четырех суток назад? Используйте опцию `-mtime +2 -a -mtime +4`. (Мы объединили две опции с помощью оператора AND.) Вероятно, вы получите несколько вчерашних файлов, но их можно удалить с помощью команды `grep -v` (см. главу 10). Помните, что опции `-atime`, `-ctime` и `-mtime` означают `n*24` часов, отсчитывая от текущего момента времени, так что результаты могут трактоваться в широком смысле.

ЗАМЕЧАНИЕ

Существуют и другие выражения, использующие время, которые я не стал описывать из-за недостатка места. В частности, вам могут понадобиться опции `-anewer`, `-cnewer`, `-newer` и `-newerXY`.

Отображение результатов при выполнении всех выражений (AND)

```
find -a
```

Главной особенностью команды `find` является возможность объединять несколько опций, чтобы конкретизировать поиск. Опция `-a` (или `-and`) позволяет связать вместе столько опций, сколько вам необходимо. Предположим, например, что вы хотите найти песню "Let it Bleed" группы "Rolling Stones". Вы можете использовать опцию `-name "*Let It Bleed*"`, но она не даст необходимых результатов. Группа "Rolling Stones" записала альбом с таким же названием, поэтому нам надо отделить файлы от каталогов. Следовательно, надо также задать опцию `-type f`. Объединим вместе две опции.

```
$ cd Rolling_Stones
$ find . -name "*Let It Bleed*" -a -type f
./1972 More Hot Rocks/17 - Let It Bleed.mp3
./1995 Stripped/08 - Let It Bleed.mp3
./1969 Let It Bl eed/5 - Let It Bleed.mp3
```

Вроде бы все в порядке, но сколько песен группы "Rolling Stones" мы получили реально? Передадим результаты работы команды `find` программе `wc` (ее имя является сокращением от *word count*; см. главу 7) и зададим опцию `-l`, чтобы подсчитывать не слова, а строки.

```
$ cd Rolling_Stones
$ find . -name "*mp3*" -a -type f | wc -l
1323
```

Мы получили 1323 песни группы "Rolling Stones".

Отображение результатов при выполнении любого из выражений (OR)

```
find -o
```

Ранее мы использовали команду `find` для поиска всех записей группы "Clash" из альбома "London Calling", которые содержались в файлах размеров 10 Мбайт. Там же выполнялся поиск файлов, размер которых превышает 10 Мбайт, но объединить результаты опция `-size` не позволяла. Из предыдущего раздела вы узнали, что опция `-a` дает возможность объединять другие опции в виде логического выражения AND. Сейчас мы используем опцию `-o` (или `-or`) для реализации операции OR.

Итак, чтобы найти записи из альбома "London Calling", которые содержатся в файле размером не менее 10 Мбайт, воспользуемся следующей командой:

```
$ cd Clash
$ find . -size +10M -o -size 10M
./1977 The Clash/01 - Clash City Rockers.mp3
./1977 The Clash/13 - Police And Thieves.mp3
```

```
./1979 London Calling/15 - Lover's Rock.mp3
./1979 London Calling/18 - Revolution Rock.mp3
./2007 The Sandinista Project/04 - Jason
↳Ringenberg - Ivan Meets G.I. Joe.m4a
./1980 Sandinista!/01 - The Magnificent Seven.mp3
./1980 Sandinista!/02 - Hitsville U.K.mp3
[Results greatly truncated for length]
```

И снова проблема. Мы получили также сведения об альбоме "The Sandinista Project" группы "Clash", а это не входило в наши планы. Нам надо сделать следующее: во-первых, исключить данные о данном альбоме, а во-вторых, убедиться, что выражение `OR` работает корректно.

Чтобы убрать записи, посвященные альбому "The Sandinista Project", допишем в конец команды конструкцию `! -name "*Project*"`, чтобы получилась команда `find . -size +10M -o -size 10M ! -name "*Project"`. Однако она тоже не работает по нескольким причинам.

Опция `-name` является неправильной, потому что она ищет только имена файлов, игнорируя обратную косую черту перед ними. Слово, которое мы хотим исключить из списка результатов, — `Project` — представляет имя каталога, поэтому опция `-name` его никогда не обнаружит. Вместо этого следует использовать опцию `-wholename`, которая ищет весь путь, включая имя файла.

Другая проблема — способ организации команды. У нас есть два выражения, объединенных оператором `OR` с другим выражением для опции `wholename`. Чтобы проверить, что оператор `OR` делает именно то, что нам нужно, необходимо заключить его в скобки, которые объединят инструкции (как в алгебре!). Однако сочетания скобок и обратной косой черты следует избегать, потому что оболочка может их неправильно интерпретировать. По этой причине необходимо до и после инструкции вставить пробелы, иначе она не будет работать. В результате получаем следующую комбинацию:

```
$ find . \( -size +10M -o -size 10M \) -a
↳ ! -wholename "*Project*"
./1977 The Clash/01 - Clash City Rockers.mp3
```

```
./1977 The Clash/13 - Police And Thieves.mp3
./1979 London Calling/15 - Lover's Rock.mp3
./1979 London Calling/18 - Revolution Rock.mp3
./1980 Sandinista!/01 - The Magnificent Seven.mp3
./1980 Sandinista!/02 - Hitsville U.K.mp3
```

Альбом "Sandinista Project" прекрасен, но нам нужны только песни группы "Clash". Благодаря команде `find` мы нашли их.

ЗАМЕЧАНИЕ

Альбому "London Calling" посвящена страница сайта `Allmusic.com` (`www.allmusic.com/album/london-callingmw0000189413`) и статья в Википедии (`http://en.wikipedia.org/wiki/London_calling`).

Кроме того, с помощью опции `-o` можно выяснить, сколько песен записано на музыкальном диске. Можно начать со следующей команды, которую надо выполнить из каталога `/media/music`, используя опцию `-a` (терпение, мы еще доберемся до опции `-o`):

```
$ find . -name "*mp3" -a -type f | wc -l
105773
```

Сто пять тысяч? Не может быть. А, теперь понятно. Мы ищем только файлы `mp3`, а некоторые песни записаны в формате M4A. Поищем файлы `mp3` или `m4a`, а затем подсчитаем количество результатов с помощью команды `wc -l`:

```
$ find . \( -name "*mp3" -o -name "*.m4a" \) -a
↳-type f | wc -l
106666
```

Восемьсот девяносто три файла в формате m4a — вполне правдоподобный результат, но я забыл о файлах FLAC! Добавим еще одну опцию `-o`:

```
$ find . \( -name "*mp3" -o -name "*.m4a*" -o - name
↳"*.flac" \) -a -type f | wc -l
109709
```

Вот теперь кажется все: мы нашли около 110000 песен!

ЗАМЕЧАНИЕ

В предыдущем издании я получил такие результаты: 23407 файлов формата mp3, 0 файлов формата m4a, 18244 файла формата ogg (теперь их нет) и 556 файлов формата flac. Всего на диске было 42187 музыкальных записей. Очевидно, раньше у меня было меньше свободного времени!

Отображение результатов, если выражение не выполняется (NOT)

find -n

Мы уже использовали ранее символ `!` для того, чтобы инвертировать значение выражения. Вернемся к рассмотрению этого оператора. В предыдущем разделе с помощью команды `find` мы определили, сколько файлов `mp3`, `mp4a`, `ogg` и `flac` находится на диске. А сколько всего файлов на диске?

```
$ find . | wc -l
122255
```

Из 122255 файлов только 109709 имеют форматы mp3, m4a или flac. А зачем остальные? Сформируем команду `find`, которая исключит из поиска файлы, имена которых заканчиваются на `.mp3`, `.m4a` или `.flac`. Также нам не следует учитывать каталоги. Поместим эти четыре условия в круглые скобки и зададим перед всем выражением символ `!`. Таким образом мы отобразим лишь объекты, не удовлетворяющие четырем описанным выше условиям.

```
$ find . ! \( -name "*mp3" -o -name "*m4a" -o
↳-name "*flac" -o -type d \)
./Rock - Folk/Band/1970 Stage Fright/Art/
```

```
↳Cover.png
./Rock - Folk/Bob Dylan/2007 Dylan Hears a Who/
↳Art/Cover.jpg
./Rock - British Blues/Rolling Stones/1973 0908
↳ Wembley, London/Info.txt
./Classical - Opera/Puccini/Turandot/1972
.↳Sutherland, Pavarotti, Caballe/Libretto.pdf
./Rock - Pop/ABBA/1993 Abba Gold Greatest Hits/
↳01 - Dancing Queen.MP3
./Rock/Neil Young/1970 0301 Tea Party, Boston/
↳Info.htm
./Rock - Punk/Clash/2007 The Sandinista Project/
↳Booklet.pdf
./Jazz - Vocal/Frank Sinatra/1963 The Concert
.↳Sinatra/07 - This Nearly Was Mine.Mp3
[Результаты сокращены для экономии места]
```

Мы получили ответ, или, вернее, сведения, позволяющие его сформулировать. У нас есть файлы PDF, текстовые файлы, файлы HTML, а также файлы с расширением `MP3` вместо `mp3` (не считая файлов JEG, PNG и видеофайлов). Как и все программы Linux, команда `find` чувствительна к регистру символов (см. главу 1), поэтому при поиске по `"*mp3*"` не будут найдены файлы с суффиксом `.MP3`. Существует ли простой способ изменить имена файлов так, чтобы их суффиксы были представлены в нижнем регистре? Такой способ есть, и он предполагает использование команды `find`.

ПОДСКАЗКА

Выражение `-name` чувствительно к регистру, а выражение `-iname` — нет (буква `i` именно это и означает — *insensitive* (нечувствительный)). Следовательно, если нужно найти все файлы mp3, независимо от регистра их расширения, следует использовать команду `find . -iname "*mp3*"`, которая найдет файл с расширениями `mp3`, `MP3`, `Mp3` и `mP3`.

Выполнение действий над каждым найденным файлом

find -exec

Теперь вы готовы ознакомиться с той областью, в которой команда `find` реально проявляет свои огромные возможности. Над каждым файлом, найденным посредством команды `find`, можно выполнять произвольные действия. После опций, сужающих поиск (например, `-name`, `-type` или `-user`), можно задать опцию `-exec`, а затем ввести команду, которую необходимо применить к файлам. Для представления каждого файла используются символы {}, а команду надо завершить обратной косой чертой, чтобы отменить специальное действие точки с запятой, иначе оболочка воспримет ее как разделитель при объединении команд (см. главу 5).

В предыдущем разделе мы выяснили, что имена некоторых файлов на диске оканчиваются последовательностью символов `MP3`. Поскольку мы предпочитаем суффиксы, составленные из символов нижнего регистра, преобразуем все вхождения `MP3` в `mp3`. Для этой цели используем опцию `-exec` команды `find`. Сначала убедимся, что файлы с суффиксом `.MP3` действительно существуют.

```
$ find . -name "*MP3"
./Blues - Delta/Robert Johnson/1990 Complete
↳ Recordings/29 - Malted Milk.MP3
./Blues - Delta/Robert Johnson/1990 Complete
.↳Recordings/16 - Dead Shrimp Blues.MP3
./Blues - Delta/Robert Johnson/1990 Complete
↳ Recordings/11 - Terraplane Blues.MP3
```

Затем сформируем команду для изменения суффиксов. В качестве значения опции `-exec` зададим программу `rename`, которая позволяет изменять имя файла или его часть.

```
$ find . -name " *MP3"
↳ -exec rename 's/MP3/mp3/g' {} \;
```

ПОДСКАЗКА

Существует по крайней мере две разные версии команды `rename`, каждая из которых имеет свои синтаксис и опции. Я использовал пример, написанный Ларри Уоллом (Larry Wall), изобретателем языка программирования Perl, который — вполне ожидаемо — использует для подстановки синтаксис, похожий на язык Perl. Лучшим сообщением на эту тему является статья "A Tale of Two Renames", написанная Тимом Хини (Tim Heaney) и размещенная по адресу `http://chnsa.ws/68`.

Для проверки версии команды `rename` откройте справочную страницу `man rename` и прокрутите ее вниз до раздела AUTHOR. Если в вашем дистрибутивном пакете Linux нет команды `rename` (что очень даже возможно), вам придется инсталлировать ее, следуя инструкциям, изложенным в главе 14.

Команде `rename` передаются инструкции по изменению имени в формате `s/`*`старое_имя`*`/`*`новое_имя`*. (В данном случае `s` — первая буква слова *substitute* (подставить).) Ознакомимся с результатами выполнения команды.

```
$ find . -name "*MP3"
./Blues - Delta/Robert Johnson/1990 Complete
↳ Recordings/29 - Malted Milk.mp3
./Blues - Delta/Robert Johnson/1990 Complete
↳ Recordings/16 - Dead Shrimp Blues.mp3
./Blues - Delta/Robert Johnson/1990 Complete
↳ Recordings/11 - Terraplane Blues.mp3
```

ПОДСКАЗКА

В нашем примере мы использовали формат `s/`*`старое_имя`*`/`*`новое_имя`*, но вместо него часто применяется формат `s/`*`старое_имя`*`/`*`новое_имя`*`/g` (буква `g` означает global (глобальный)). Если каждое имя файла встречается и заменяется только один раз, как файл "`Terraplane`

Blues.mp3, то команда s/MP3/mp3 работает просто отлично. А что делать с песней Роберта Джонсона под названием Corrupted MP3 Blues.MP3". Для того чтобы изменить оба имени MP3 на mp3, необходимо использовать инструкции s/MP3/mp3/g. Благодаря этому команда rename будет знать, что мы хотим заменить MP3 на mp3 не только в первом файле (как это делает инструкция s/MP3/mp3/, но и во втором.

Итак, все расширения MP3 заменены на mp3. Применим тот же подход для решения другой задачи. В предыдущем разделе мы выяснили, что на диске содержатся файлы m3u. Так получилось, что в именах многих из них содержатся символы подчеркивания, которые хотелось бы заменить пробелами. Сгенерируем список файлов m3u, содержащих символы подчеркивания. Используем шаблон поиска * *_*m3u. В нем слева и справа от символа подчеркивания помещены символы групповой операции, т.е. таким образом выявляются файлы, в имени которых содержится хотя бы один символ подчеркивания.

```
$ find . -name "*_*m3u"
./Holiday_-_Christmas/Christmas_With_The_Rat_
↳Pack.m3u
./Holiday_-_Christmas/Boots_For_Your_Stockings.m3u
./Classical_-_Baroque/Handel/Chamber_Music.m3u
./Classical_-_Opera/Famous_Arias.m3u
./R&B_-_Doo_Wop/Doo_Wop_Box.m3u
./Electronica/Aphex_Twin/I_Care_Because_You_Do.m3u
```

Теперь к каждому найденному файлу применим команду rename. Символ для подстановки имеет вид "\ "; при этом команда rename получает информацию о символе подчеркивания, а оболочка не обрабатывает его нежелательным для нас образом. Знак подчеркивания заменяем пробелом.

```
$ find . -name "* *m3u"
↳-exec rename 's/\ /_/g' {} \;
$ find . -name "* *m3u"
$
```

Команда работает так, как мы и ожидали.

ЗАМЕЧАНИЕ

Обратите внимание на то, что, перед тем, как применять к файлам команду изменения имени, необходимо выяснить, какие файлы затронет эта операция. Не стоит пренебрегать такой проверкой, в противном случае можно получить нежелательные результаты.

Более эффективный поиск файлов

```
find +
find | xargs
```

Недавно я работал на клиентском сервере, который из-за неправильной конфигурации оказался заполненным 1,5 миллиона временных файлов. Да, полтора миллиона файлов. Можно предположить, что эта проблема решается простым сочетанием команды `find` и `rm`, но, попытавшись сделать это, вы получите следующее:

```
$ ls | less
sess_000000001.txt
sess_000000002.txt
sess_000000003.txt
[Результаты сокращены с 1.5 млн до трех]
$ find . -name "sess_*" -exec rm {} \;
/bin/rm Argument list too long
```

Из-за ограничений ядра системы Linux задать такое громадное количество аргументов команды физически невозможно. Необходимо как-то избавиться от всех этих файлов, но как?

Новомодное решение заключается в использовании команды `find` (а затем надо терпеливо ждать, пока не исчезнут 1,5 млн файлов).

```
$ find . -name "sess_*" -exec rm {} +
```

В отличие от предыдущего случая, опция `-exec` здесь используется двумя способами: отсутствуют символы `\;` и используется символ `+` (фактически именно символ `+` в конце позволяет отбросить символы `\;`). Обычно команда `find` применяет команду `rm` к каждому найденному файлу, что очень неэффективно и приводит к слишком длинному списку аргументов, но символ `+` указывает команде `find`, что указанную порцию файлов необходимо объединить в одно целое и применять команду `rm` к каждой порции, а не к отдельным файлам.

Почему бы не использовать символ `+` с командой `find` во всех случаях? Потому что эта возможность появилась в системе UNIX относительно недавно, в 2006 г., и не все версии команды `find` ее поддерживают. Для проверки можно выполнить безопасную команду вроде `find . -exec ls {} +`. Если вместо сообщения об ошибке вы получите результаты, то ваша версия команды `find` поддерживает символ `+` (и приготовьтесь нажать комбинацию клавиш <Ctrl+C>, чтобы остановить прокрутку огромного списка файлов!).

Классический принцип "что сработало один раз, должно работать постоянно", подсказывает нам, что следует использовать команду `xargs`. Эта команда считывает аргументы из потока STDIN, а не из командной строки. Успех такого подхода объясняется тем, что команда `xargs` обрабатывает данные порциями, а не по одному (напоминает сочетание символа `+` и команды `find`, не так ли?). По этой причине команда `xargs` выполняется реже и позволяет избежать сообщений о слишком длинном списке аргументов. Насколько большими должны быть порции? Выполните команду `xargs --show-limits` (чтобы остановить выполнение команды `xargs`, нажмите комбинацию клавиш <Ctrl+D>).

ЗАМЕЧАНИЕ

Конвейеры и команда `xargs` похожим образом обрабатывают и используют результаты команд (причем команда `xargs` почти всегда использует конвейеры!), но

между ними есть существенная разница. Конвейер получает результат первой команды и использует его как ввод для *второй*. В то же время команда `xargs` получает результаты первой команды (как в конвейере) и использует его как *аргументы* второй.

Например, команда `ls -1 | wc -l` получает результат команды `ls -1` и использует его как ввод для команды `wc -l`, потому что команда `wc` использует поток `STDIN`, а команда `ls -1 | echo` ничего не выводит, потому что команда `echo` ожидает аргументы, а не данные из потока `STDIN`. Поскольку команда `echo` получает аргументы, команда `ls -1 | xargs echo` работает, потому что команда `xargs` превращает результаты работы команды `ls -1` в аргументы команды `echo`. (Да, повторять, как эхо, результаты команды `ls` довольно глупо, но я это делаю только для иллюстрации!)

Вернемся к 1,5 млн файлов и применим действие `-exec` с командой `find`, затем передадим результаты команды `find` команде `xargs` и удалим эти файлы.

```
$ find . -name "sess_*" | xargs rm
$ ls
[Все! 1.5 млн файлов удалены!]
```

Удаление 1,5 млн файлов занимает довольно много времени, но все же оно выполняется, а это уже хорошо.

ПОДСКАЗКА

Несмотря на то что команда `xargs` часто образует конвейер с командой `find`, ее можно использовать с разными командами. Однако в 90% примеров, которые можно найти в веб, используется комбинация команд `xargs` и `find`!

Поиск файлов, имена которых содержат пробелы

```
find -print0 | xargs -0
```

В предыдущем разделе был описан процесс удаления 1,5 млн файлов, имена которых выглядели как `sess_000000001.txt`, поэтому мы использовали сочетание команд `find` и `xargs`, например:

```
$ find . -name "sess_*" | xargs rm
```

Этот вариант работает, так как имя `sess_000000001.txt` и ему подобные не содержат пробелов. Если бы имена файлов выглядели, как, например, `sess 000000001.txt`, то мы получили бы такие результаты:

```
$ find . -name "sess_*" | xargs rm
rm: cannot remove './sess': No such file or
↳directory
rm: cannot remove '000000001.txt': No such file or
↳directory
rm: cannot remove './sess': No such file or
↳directory
rm: cannot remove '000000002.txt': No such file or
↳directory
^C
```

Символы `^C` в конце позволяют прекратить вывод трех миллионов строк с помощью комбинации клавиш <Ctrl+C> (каждому файлу соответствуют две строки, поскольку имена файлов содержат один пробел). Очевидно, что пробелы в именах файлов создают проблему. Но почему?

Команда `xargs` использует пробелы для разделения результатов, поэтому, когда команда `find` передает ей имя `sess 000000001.txt`, команда `xargs` видит их как `rm sess` и `rm 000000001.txt`. Поскольку таких файлов не существует, команда `rm` выдает ошибку. Кстати, это может оказаться опасным — если файл с именем `000000001.txt` существовал, то он будет удален.

Проблемы создают не только пробелы в именах файлов. Такие символы, как кавычки, апострофы, символы перехода на новую строку и обратная косая черта, также приводят к неожиданных результатам и сбоям. Для того чтобы решить эти проблемы, необходимо выполнить команду

```
$ find . -name "sess_*" -print0 | xargs -0 rm
```

Как правило, команда `find` передает каждое имя файла с символом перехода на новую строку (именно поэтому вы видите все результаты команды `find` в отдельных строках). Действие `-print0` указывает команде `find` сопровождать каждое имя файла не символом перехода на новую строку, а нулевым символом. Между прочим, опция `-0` указывает команде `xargs`, что для разделения результатов она должна использовать нулевой символ вместо пробела. Поскольку команды `find` и `xargs` работают вместе, вы не получите никаких сообщений об ошибке. Кроме того, опция `-0` указывает команде `xargs` игнорировать кавычки и любые другие символы, которые могут вызвать сбой.

Выводы

Как вы уже знаете, возможности команды `find` огромны. Чтобы получить подробную информацию о них, выполните команду `man find` или прочитайте руководства, доступные в веб. Разнообразные опции `find` можно использовать для получения списка файлов и каталогов, а опция `-exec` делает данную программу поистине бесценной, позволяя применять произвольные команды к результатам поиска. Можно также передавать выходные данные `find` другим командам. Команда `find` — одна из самых полезных в системе Linux. Используйте ее!