Занятие №8. Поиск данных

Одной из самых часто используемых операций при работе в Linux является поиск файлов и каталогов. Существует несколько команд, которые позволяют искать файлы, из которых сейчас мы рассмотрим locate и find.

locate

Команда `locate` выполняет быстрый поиск в базе данных имен файлов и выводит все имена, соответствующие заданной строке.

```
user@linux-pc:~$ locate /etc/bash.bashrc
/etc/skel/.bashrc
/home/user/.bashrc
/usr/share/base-files/dot.bashrc
/usr/share/doc/adduser/examples/adduser.local.conf.examples/bash.bashrc
/usr/share/doc/adduser/examples/adduser.local.conf.examples/skel/dot.bashrc
```

База данных, которую использует команда `locate`, обновляется с помощью команды `updatedb`. Делается это, обычно, автоматически с помощью системной службы cron раз в сутки. Базу данных можно обновить и вручную, запустив команду `updatedb` от имени суперпользователя.

Команда locate в качестве строки поиска может принимать шаблон, содержащий групповые символы:

```
user@linux-pc:~$ locate /home/user/ls_errors.txt
/home/user/ls_output.txt
/home/user/my_name.txt
```

find

В том числе по материалам статьи

В отличие от команды locate, выполняющей поиск по именам, команда find ищет файлы согласно заданным атрибутам и в указанном каталоге (и во всех вложенных подкаталогах).

В общей форме команда find записывается следующим образом:

```
find [каталог] [критерии_поиска] [действие]
```

Критерии поиска

Команда find поддерживает большое количество различных критериев поиска, среди которых можно выделить

- поиск по типу;
- поиск по имени;
- поиск по размеру;
- и другие (man find).

```
user@linux-pc:~$ find . -type f | wc -l
40541
user@linux-pc:~$ find . -type f -name "*.txt" | wc -l
2403
user@linux-pc:~$ find . -type f -name "*.txt" -size +1k | wc -l
133
```

Первая команда показывает количество различных объектов файловой системы в текущем каталоге; вторая — количество текстовых файлов; третья — количество текстовых файлов, размер которых превышает 1 Кб.

Критерии поиска можно объединять с помощью логических операторов

- -and
- -or
- -not

По умолчанию критерии поиска «соединены» логическим оператором -and.

```
user@linux-pc:~$ find . | wc -l
48142
user@linux-pc:~$ find . -type f -or -type d | wc -l
44272
user@linux-pc:~$ find . -type l | wc -l
3870
user@linux-pc:~$ find . -not \( -type f -or -type d \) | wc -l
3870
```

Как и в прошлый раз первая команда показывает количество различных объектов файлов системы в текущем каталоге; вторая — количество каталогов и файлов среди них; третья и четвертая — количество символьных ссылок.

Третья и четвертая команда не эквивалентны, потому что команда find к типам файлов, которые отличны от каталогов и обычных файлов, кроме символьных ссылок относит также ряд других специальных файлов. Просто последние в текущем каталоге примера отсутствуют.

Круглые скобки, которые вы видите в четвертой команде, группируют критерии поиска и логические операторы для формирования сложных выражений и используются для управления порядком проверок. По умолчанию проверки выполняются слева направо. Для вычисления логического выражения используется сокращенная схема.

Круглые скобки имеют специальное значение для командной оболочки, поэтому их нужно экранировать, чтобы они передавались команде find как аргументы. Экранирование выполняют с помощью символа обратного слеша.

Действия

К команде find можно добавить действия, которые будет применены к результатам поиска. В таблице указаны некоторые из таких действий:

Операции	**Описание**
-delete	Удаляет текущий найденный файл.
-ls	Выводит более подробные результаты поиска.
-print	Выводит полный путь к найденному файлу. Это действие выполняется по умолчанию.
-quit	Завершает выполнение команды после обнаружения первого совпадения.
-exec	Выполняет указанную команду.

Например, вот так можно удалить все пустые каталоги и файлы

```
user@linux-pc:~/test$ find . -empty -delete
```

Перед удалением лучше лишний раз проверить, что именно будет удалено, и запустить команду с действием -print.

Действие -exec имеет следующий синтаксис

```
-exec команда {} \;
```

где команда — это имя команды, {} - обозначает результат поиска, ; - обязательный разделитель, обозначающий конец команды.

Вот так выглядит альтернатива действию -delete

```
user@linux-pc:~/test$ find . -empty -exec rm {} \;
```

Простые регулярные выражения (Basic Regular Expression)

Полезно щёлкать сюда

Регулярное выражение – символьная форма записи, используемая для идентификации шаблонов в тексте. Различают базовые регулярные выражения (Basic Regular Expressions) и расширенные регулярные выражения (Extended Regular Expressions). Базовые регулярные выражения являются менее мощными, но гарантировано поддерживаются всеми терминалами.

Для знакомства с регулярными выражениями воспользуемся командой grep

```
grep параметры регулярное_выражение файл...
```

Подготовим несколько текстовых файлов:

```
user@linux-pc:~/test$ ls /bin > ls_bin.txt
user@linux-pc:~/test$ ls /usr/bin > ls_usr_bin.txt
user@linux-pc:~/test$ ls /sbin > ls_sbin.txt
user@linux-pc:~/test$ ls /usr/sbin > ls_usr_sbin.txt
user@linux-pc:~/test$ ls
ls_bin.txt  ls_sbin.txt  ls_usr_bin.txt  ls_usr_sbin.txt
```

Символы, из которых состоит регулярное выражение, делятся на литеральные символы и метасимволы. Литеральные символы соответствуют сами себе. Метасимволы используются для определения более сложных критериев совпадения. К метасимволам регулярных выражений относятся следующие символы:

```
^ $ . [ ] - * \
```

Символ обратного слеша может использоваться для экранирования метасимволов, чтобы они могли использоваться как литеральные символы.

Метасимвол “.”

Символ “.” соответствует любому символу. В регулярном выражении он соответствует любому символу в данной позиции

```
user@linux-pc:~/test$ grep '.zip' *.txt
ls_bin.txt:bunzip2
ls_bin.txt:bzip2
ls_bin.txt:bzip2recover
ls_bin.txt:funzip
ls_bin.txt:gpg-zip
ls_bin.txt:gunzip
ls_bin.txt:gzip
ls_bin.txt:unzip
ls_bin.txt:unzipsfx
ls_usr_bin.txt:bunzip2
ls_usr_bin.txt:bzip2
ls_usr_bin.txt:bzip2recover
ls_usr_bin.txt:funzip
ls_usr_bin.txt:gpg-zip
ls_usr_bin.txt:gunzip
ls_usr_bin.txt:gzip
ls_usr_bin.txt:unzip
ls_usr_bin.txt:unzipsfx
```

Обратите внимание, что программа zip не была найдена, потому что метасимвол . увеличивает длину обязательного совпадения до четырех символов.

Метасимволы ^ и $ (якоря)

Для определения шаблона, который должен совпадать с началом строки текста, служит символ ^. Если шаблон обнаруживается в любом другом месте строки, считается, что совпадение отсутствует. Для поиска шаблона в конце строки текста используется символ $.

```
user@linux-pc:~/test$ grep '^zip' *.txt
ls_bin.txt:zipdetails
ls_bin.txt:zipgrep
ls_bin.txt:zipinfo
ls_usr_bin.txt:zipdetails
ls_usr_bin.txt:zipgrep
ls_usr_bin.txt:zipinfo
user@linux-pc:~/test$ grep 'zip$' *.txt
ls_bin.txt:funzip
ls_bin.txt:gpg-zip
ls_bin.txt:gunzip
ls_bin.txt:gzip
ls_bin.txt:unzip
ls_usr_bin.txt:funzip
ls_usr_bin.txt:gpg-zip
```

```
ls_usr_bin.txt:gunzip
ls_usr_bin.txt:gzip
ls_usr_bin.txt:unzip
user@linux-pc:~/test$ grep '^gzip$' *.txt
ls_bin.txt:gzip
ls_usr_bin.txt:gzip
```

Регулярное выражение ‹^$› будет соответствовать пустым строкам.

Выражения в квадратных скобках

Выражение в квадратных скобках позволяет определить множество символов, один из которых должен находиться в данной позиции

```
user@linux-pc:~/test$ grep '[bg]zip' *.txt
ls_bin.txt:bzip2
ls_bin.txt:bzip2recover
ls_bin.txt:gzip
ls_usr_bin.txt:bzip2
ls_usr_bin.txt:bzip2recover
ls_usr_bin.txt:gzip
```

Метасимволы, заключенные в квадратные скобки, теряют свое специальное значение. При этом символ ^ используется для обозначения отрицания, а символ - (дефис) – для обозначения диапазона. Символ ^ обозначает отрицание только если он является первым символом в квадратных скобках. В противном случае он превращается в обычный символ. Если требуется включить символ - в выражение в квадратных скобках, он должен быть первым символом в скобках.

```
user@linux-pc:~/test$ grep '[^bg]zip' *.txt
ls_bin.txt:bunzip2
ls_bin.txt:funzip
ls_bin.txt:gpg-zip
ls_bin.txt:gunzip
ls_bin.txt:unzip
ls_bin.txt:unzipsfx
ls_usr_bin.txt:bunzip2
ls_usr_bin.txt:funzip
ls_usr_bin.txt:gpg-zip
ls_usr_bin.txt:gunzip
ls_usr_bin.txt:unzip
ls_usr_bin.txt:unzipsfx
```

Регулярное выражение ‹[\^bg]zip› используется для поиска строк, которые содержат подстроку ‹zip›, которой предшествует любой символ кроме b и g.

```
user@linux-pc:~/test$ grep '^[ABCDEFGHIJKLMNOPQRSTUVWXYZ]' *.txt
ls_bin.txt:NF
ls_bin.txt:VGAuthService
ls_bin.txt:X11
ls_usr_bin.txt:NF
ls_usr_bin.txt:VGAuthService
ls_usr_bin.txt:X11
user@linux-pc:~/test$ grep '^[A-Z]' *.txt
ls_bin.txt:NF
ls_bin.txt:VGAuthService
ls_bin.txt:X11
ls_usr_bin.txt:NF
ls_usr_bin.txt:VGAuthService
ls_usr_bin.txt:X11
```

С помощью регулярных выражений ‹^[ABCDEFGHIJKLMNOPQRSTUVWXYZ]› и ‹^[A-Z]› ищутся имена каталогов и файлов, которые начинаются с заглавной буквы.

Классы символов POSIX

Диапазоны символов — простой способ определения множества символов. К сожалению, этот способ может использоваться не со всеми программами из-за языковых настроек.

Для решения этой проблемы стандарт POSIX предусматривает несколько классов символов, описывающих диапазоны символов.

Класс символов	**Описание**
[:alnum:]	Алфавитно-цифровые символы (эквивалент диапазона [A-Za-z0-9] в ASCII)
[:word:]	То же, что [:alnum:] плюс символ подчеркивания
[:alpha:]	Алфавитные символы (эквивалент диапазона [A-Za-z] в ASCII)
[:blank:]	Символы пробела и табуляции
[:ctrl:]	Управляющие символы ASCII с кодами от 0 до 31 плюс код 127
[:digit:]	Цифры от 0 до 9
[:graph:]	Отображаемые символы (включает символы ASCII с кодами от 33 до 126)

Класс символов	Описание
[:lower:]	Символы нижнего регистра
[:punct:]	Знаки пунктуации (эквивалент класса ['^$'\-!“#$%&'()*+,./:;<=>?@[\]_`{|}~] в ASCII)
[:print:]	Печатные символы (все символы из класса [:graph:] плюс пробел)
[:space:]	Пробельные символы (эквивалент класс [\t\r\n\v\f] в ASCII)
[:upper:]	Символы верхнего регистра
[:xdigit:]	Символы, используемые для представления шестнадцатеричных цифр (эквивалент класса [0-9A-Fa-f] в ASCII)

Используя классы символов, можно повторить пример, приведенный выше, следующим образом

```
user@linux-pc:~/test$ grep '^[[:upper:]]' *.txt
ls_bin.txt:NF
ls_bin.txt:VGAuthService
ls_bin.txt:X11
ls_usr_bin.txt:NF
ls_usr_bin.txt:VGAuthService
ls_usr_bin.txt:X11
```

*Метасимвол * (квантификатор)*

Чтобы указать, что некоторый символ в строке должен появиться нуль или большее количество раз при сопоставлении с шаблоном, нужно поместить после символа ‹*›

```
user@linux-pc:~/test$ grep 'z*ip' *.txt
ls_bin.txt:bunzip2
ls_bin.txt:bzip2
ls_bin.txt:bzip2recover
ls_bin.txt:dpkg-maintscript-helper
ls_bin.txt:funzip
ls_bin.txt:gpg-zip
ls_bin.txt:gunzip
ls_bin.txt:gzip
ls_bin.txt:ip
ls_bin.txt:ipcmk
ls_bin.txt:ipcrm
ls_bin.txt:ipcs
ls_bin.txt:iptables-xml
```

```
ls_bin.txt:lesspipe
ls_bin.txt:lsipc
ls_bin.txt:ntfswipe
ls_bin.txt:psfstriptable
ls_bin.txt:script
ls_bin.txt:scriptreplay
...
```

Для обозначения любого количеств любых символов в регулярных выражениях использую комбинацию символа . и символа “*”.

```
user@linux-pc:~/test$ locate -r '.*\.txt$' | wc -l
9402
user@linux-pc:~/test$ locate '*.txt' | wc -l
9402
```

Звездочку можно применять не только к отдельному символу, но и к классу символов. Например, с помощью регулярного выражения ‹b[ae]*t› находятся строки, в которых между символами b и t появляются символы a и e в любых сочетаниях.

Литература

1. Шотсс У. “Командная строка Linux.”, глава 17, глава 19
2. Блум Р., Бреснахэн К. “Командная строка Linux и сценарии оболочки.”, глава 19