Input/Output (I/O). Linux

Перенаправление ввода/вывода.

Хотя обычно ввод/вывод программы связаны со стандартными потоками, в оболочке существуют специальные средства для перенаправления ввода/вывода.

Операторы >, < и >>

Для обозначения перенаправления используются символы "**>**", "**<**" и "**>>**". Чаще всего используется перенаправление вывода команды в файл. Вот соответствующий пример:

$ ls -l > /home/user/dir.txt

По этой команде в файле /home/user/dir.txt будет сохранен перечень файлов и подкаталогов того каталога, который был текущим на момент выполнения команды ls. При этом если указанного файла не существовало, то он будет создан, если он существовал, то будет перезаписан, если же нужно, чтобы вывод команды был дописан в конец существующего файла, то надо вместо символа > использовать >>. При этом наличие пробелов до или после символов **>** или **>>** служит только для удобства.

Ты можешь направить вывод не только в файл, но и на вход другой команды или на устройство (*например, принтер*). Так, для подсчета числа слов в файле /tmp/report.txt можно использовать следующую команду:

$ cat /tmp/report.txt | wc -w

а для вывода файла на печать - команду:

$ cat /tmp/report.txt > lpr

Как видишь, оператор **>** служит для перенаправления выходного потока. Аналогично для перенаправления входного потока применяется оператор <. Пример выше можно переписать так (*внимание на отсутствие команды cat*):

$ wc -w < /tmp/report.txt

Этот вариант перенаправления часто используется в различных скриптах, для команд, которые обычно воспринимают ввод (*или ожидают ввода*) с клавиатуры. В скрипте же, можно дать команде необходимую информацию из файла, в который заранее записано то, что нужно ввести для выполнения этой команды.

В силу того, что символы <, > и >> действуют на стандартные потоки, их можно использовать не только тем привычным образом, как это делается обычно, но и несколько по-другому. Так, следующие команды эквивалентны:

$ cat > file

$ cat>file

$ >file cat

$ > file cat

Однако сам по себе (*без какой-либо команды, для которой определены стандартные потоки*) символ перенаправления не может использоваться, так что нельзя, например, введя в командной строке

$ file1 > file2

получить копию какого-то файла, так как не выполняется никакой команды. Зато стандартные потоки определены для любой команды. А перенаправить можно не только стандартный ввод и вывод, но и другие потоки. Для этого надо указать перед символом перенаправления номер перенаправляемого потока. Стандартный ввод *(stdin)* имеет номер 0, стандартный вывод (*stdout*) — номер 1, стандартный поток сообщений об ошибках (*stderr*) — номер 2. То есть полный формат команды перенаправления имеет вид (напомним, что пробелы возле > не обязательны):

command N > M

где N и M — номера стандартных потоков (0,1,2) или имена файлов. Употребление в некоторых случаях символов <, > и >> без указания номера канала или имени файла возможно только потому, что вместо отсутствующего номера по умолчанию подставляется 1, т. е. стандартный вывод. Так, оператор > без указания номера интерпретируется как 1 >.

Кроме простого перенаправления стандартных потоков существует еще возможность не просто перенаправить поток в тот или иной канал, а сделать копию содержимого стандартного потока. Для этого служит специальный символ &, который ставится перед номером канала, на который перенаправляется поток:

command N > &M

Такая команда означает, что выход канала с номером N направляется как на стандартный вывод, так и дублируется в канал с номером M. Например, для того, чтобы сообщения об ошибках дублировались на стандартный вывод, надо дать команду 2>&1, в то время как 1>&2 дублирует stdout в stderr. Такая возможность особенно полезна при перенаправлении вывода в файл, так как мы тогда одновременно и видим сообщения на экране, и сохраняем их в файле.

Оператор |

Особым вариантом перенаправления вывода является организация программного канала (PIPE). Для этого две или несколько команд, таких, что вывод предыдущей служит вводом для следующей, соединяются (*или разделяются, если так будет понятнее*) символом вертикальной черты - "|". При этом стандартный выходной поток команды, расположенной слева от символа |, направляется на стандартный ввод программы, расположенной справа от символа |. Например:

$ cat myfile | grep Linux | wc -l

Эта строка означает, что вывод команды cat, т. е. текст из файла myfile, будет направлен на вход команды grep, которая выделит только строки, содержащие слово "Linux". Вывод команды grep будет, в свою очередь, направлен на вход команды wc -l, которая подсчитает число таких строк.

Программные каналы используются для того, чтобы скомбинировать несколько маленьких программ, каждая из которых выполняет только определенные преобразования над своим входным потоком, для создания обобщенной команды, результатом которой будет какое-то более сложное преобразование.

Надо отметить, что оболочка одновременно вызывает на выполнение все команды, включенные в конвейер, запуская для каждой из команд отдельный экземпляр оболочки, так что как только первая программа начинает что-либо выдавать в свой

выходной поток, следующая команда начинает его обрабатывать. Точно так же каждая следующая команда выполняет свою операцию, ожидая данных от предыдущей команды и выдавая свои результаты на вход последующей. Если тебе нужно, чтобы какая-то команда полностью завершилась до начала выполнения последующей, можешь использовать в одной строке как символ конвейера |, так и точку с запятой **;**. Перед каждой точкой с запятой оболочка будет останавливаться и ожидать, пока завершится выполнение всех предыдущих команд, включенных в конвейер.

Статус выхода (*логическое значение, возвращаемое после завершения работы программы*) из канала совпадает со статусом выхода, возвращаемым последней командой конвейера. Перед первой командой конвейера можно поставить символ "!", тогда статус выхода из конвейера будет логическим отрицанием статуса выхода из последней команды. Оболочка ожидает завершения всех команд конвейера, прежде чем установить возвращаемое значение.

Фильтры

Последний из приведенных выше примеров (*с командой grep*) можно использовать для иллюстрации еще одного важного понятия, а именно, программы-фильтра. Фильтры - это команды (*или программы*), которые воспринимают входной поток данных, производят над ним некоторые преобразования и выдают результат на стандартный вывод (*откуда его можно перенаправить куда-то еще*). К числу команд-фильтров относятся команды cat, more, less, wc, cmp, diff**,** а также следующие команды.

- grep, fgrep, egrep - Ищут во входном файле или данных со стандартного ввода строки, содержащие указанный шаблон, и выдают их на стандартный вывод.
- tr - Заменяет во входном потоке все встречающиеся символы, перечисленные в заданном перечне, на соответствующие символы из второго заданного перечня.
- comm - Сравнивает два файла по строкам и выдает на стандартный вывод 3 колонки: в одной - строки, которые встречаются только в 1 файле, во второй - строки, которые встречаются только во 2-ом файле: и в третьей - строки, имеющиеся в обоих файлах.
- pr - Форматирует для печати текстовый файл или содержимое стандартного ввода.
- sed - Строковый редактор, использующийся для выполнения некоторых преобразований над входным потоком данных (*берется из файла или со стандартного ввода*).

Особым фильтром является команда tee, которая "раздваивает" входной поток, с одной стороны направляя его на стандартный вывод, а с другой — в файл (*имя которого нужно задать*). Легко видеть, что по своему действию команда tee аналогична оператору перенаправления 1>&file.

Возможности фильтров можно существенно расширить за счет использования регулярных выражений, позволяющих организовать, например, поиск по различным шаблонам.