**6. Перенаправление**

В этом уроке мы познакомимся с самой крутой возможностью командной строки: перенаправлением ввода/вывода. Благодаря этой возможности мы сможем перенаправлять ввод и вывод команд из файлов и в файлы, а также составлять из команд целые конвейеры. Для демонстрации этой возможности введем в обиход следующие команды:

• cat — объединяет файлы.

• sort — сортирует строки текста.

• uniq — сообщает о повторяющихся строках или удаляет их.

• wc — выводит число символов перевода строки, слов и байтов в каждом указанном файле.

• grep — находит и выводит строки, соответствующие шаблону.

• head — выводит первые строки из файла.

• tail — выводит последние строки из файла.

• tee — читает данные со стандартного ввода и записывает в стандартный вывод и в файлы.

**Стандартный ввод, вывод и вывод ошибок**

Многие программы, которыми мы уже пользовались, что-нибудь выводят на консоль. Этот вывод часто делится на два типа. Первый — результаты работы программы, то есть данные, для получения которых создавалась программа. Второй — сообщения о состоянии или об ошибках, извещающие нас о самочувствии программы. Например, если взглянуть на вывод программы ls, можно увидеть, что она выводит на экран результаты своей работы и иногда сообщения об ошибках.

Согласно центральной идее Unix, что «все сущее есть файл», такие программы, как ls, в действительности выводят свои результаты в специальный файл, который называется *стандартным выводом* (standard output, часто обозначается как *stdout*), а сообщения о состоянии — в специальный файл *стандартный вывод ошибок* (standard error, *stderr*). По умолчанию оба файла, стандартный вывод и стандартный вывод ошибок, связаны с экраном и не сохраняются на диске.

Кроме того, многие программы принимают ввод из специального файла с названием *стандартный ввод* (standard input, *stdin*), который по умолчанию связан с клавиатурой.

Механизм перенаправления ввода/вывода позволяет изменять направление вывода и ввода. Обычно вывод осуществляется на экран, а ввод — с клавиатуры, но механизм перенаправления ввода/вывода позволяет изменить этот порядок вещей.

**Перенаправление стандартного вывода**

Механизм перенаправления ввода/вывода позволяет явно указать, куда должен осуществляться стандартный вывод. Чтобы перенаправить стандартный вывод в другой файл вместо экрана, нужно добавить в команду оператор перенаправления > и имя файла. Где это может пригодиться? Иногда полезно сохранить вывод команды в файл. Например, можно сообщить командной оболочке, что она должна направить вывод команды ls в файл *ls-output.txt* вместо экрана:

[me@linuxbox ~]$ **ls -l /usr/bin > ls-output.txt**

Здесь мы создали длинный список содержимого файла */usr/bin* и отправили результаты в файл *ls-output.txt*. Давайте исследуем перенаправленный вывод команды:

[me@linuxbox ~]$ **ls -l ls-output.txt**

-rw-rw-r-- 1 me  me  167878 2012-02-01 15:07 ls-output.txt

Неплохой файл получился. Если вывести содержимое *ls-output.txt* с помощью коман­ды less, можно увидеть, что он действительно содержит результаты работы команды ls:

[me@linuxbox ~]$ **less ls-output.txt**

Давайте теперь повторим эксперимент с перенаправлением, но с небольшим усложнением: укажем имя несуществующего каталога:

[me@linuxbox ~]$ **ls -l /bin/usr > ls-output.txt**

ls: невозможно получить доступ к '/bin/usr': Нет такого файла или каталога

Мы получили сообщение об ошибке. Все логично — мы указали несуществующий каталог*/bin/usr*, но почему же сообщение появилось на экране, а не было перенаправлено в файл *ls-output.txt*? Дело в том, что программа ls не выводит сообщения об ошибках в стандартный вывод. Как и многие добропорядочные программы для Unix, она выводит сообщения об ошибках в стандартный поток вывода ошибок. Поскольку мы перенаправили только стандартный вывод, а стандартный вывод ошибок — нет, сообщение об ошибке появилось на экране. Как перенаправить стандартный вывод ошибок, будет показано чуть ниже, но перед этим посмотрим, что произошло с нашим файлом:

[me@linuxbox ~]$ **ls -l ls-output.txt**

-rw-rw-r-- 1 me me 0 2012-02-01 15:08 ls-output.txt

Файл очистился! Это объясняется тем, что при перенаправлении вывода с помощью оператора > файл назначения всегда перезаписывается с самого начала. Поскольку команда ls не вывела никаких результатов, а только сообщение об ошибке, оператор перенаправления перезаписал файл, а затем остановился из-за ошибки, что привело к его очистке. Получается, что если вам понадобится очистить какой-нибудь файл (или создать новый, пустой файл), это можно сделать с помощью следующего трюка:

[me@linuxbox ~]$ **> ls-output.txt**

Простой оператор перенаправления, без предшествующей ему команды, очистит существующий файл или создаст новый, пустой файл.

Так как же добавить вывод в конец существующего файлa, не затерев его? Для этого используем оператор перенаправления >>:

[me@linuxbox ~]$ **ls -l /usr/bin >> ls-output.txt**

Оператор >> просто добавит результаты в конец файла. Если файл не существует, он будет создан, как при использовании оператора >. Давайте протестируем его:

[me@linuxbox ~]$ **ls -l /usr/bin >> ls-output.txt**

[me@linuxbox ~]$ **ls -l /usr/bin >> ls-output.txt**

[me@linuxbox ~]$ **ls -l /usr/bin >> ls-output.txt**

[me@linuxbox ~]$ **ls -l ls-output.txt**

-rw-rw-r-- 1 me  me 503634 2012-02-01 15:45 ls-output.txt

Мы повторили команду трижды и получили файл втрое большего размера.

**Перенаправление стандартного вывода ошибок**

Перенаправление стандартного вывода ошибок осуществляется не так просто, как стандартного вывода. Чтобы перенаправить стандартный вывод ошибок, нужно указать его *дескриптор файла*. Программа может производить вывод в любой из нескольких нумерованных файловых потоков. Первые три из них мы упомянули как стандартный ввод, вывод и вывод ошибок. Командная оболочка ссылается на них как на файловые дескрипторы 0, 1 и 2 соответственно. Командная оболочка поддерживает синтаксис перенаправления файлов с использованием номеров файловых дескрипторов. Так как стандартному выводу ошибок соответствует файловый дескриптор 2, мы можем перенаправить его, как показано ниже:

[me@linuxbox ~]$ **ls -l /bin/usr 2> ls-error.txt**

Номер файлового дескриптора 2 помещается непосредственно перед оператором перенаправления, чтобы перенаправить стандартный вывод ошибок в файл *ls-error.txt*.

**Перенаправление стандартного вывода и стандартного вывода ошибок в один файл**

Иногда необходимо сохранить весь вывод команды в один файл. Для этого перенаправьте сразу два потока, стандартный вывод и стандартный вывод ошибок. Сделать это можно двумя способами. Первый  — традиционный — работает в старых версиях командной оболочки:

[me@linuxbox ~]$ **ls -l /bin/usr > ls-output.txt 2>&1**

Здесь выполняется два перенаправления. Сначала — перенаправление стандартного вывода в файл *ls-output.txt*, а затем, с использованием нотации 2>&1, — перенаправление файлового дескриптора 2 (стандартный вывод ошибок) в файловый дескриптор 1 (стандартный вывод).

**ПРИМЕЧАНИЕ**

Имейте в виду, что порядок перенаправления играет важную роль. Перенаправление стандартного вывода ошибок всегда должно производиться после перенаправления стандартного вывода, иначе этот трюк не сработает. В примере, приведенном выше, последовательность > ls-output.txt 2>&1 перенаправит стандартный вывод ошибок в файл *ls-output.txt*, но если порядок перенаправления изменить на 2>&1 > ls-output.txt, стандартный вывод ошибок будет перенаправлен на экран.

Современные версии bash поддерживают второй, более простой метод выполнения перенаправления этого вида:

[me@linuxbox ~]$ **ls -l /bin/usr &> ls-output.txt**

В данном примере используется единственный оператор &>, перенаправляющий стандартный вывод и стандартный вывод ошибок в файл *ls-output.txt*.

**Удаление нежелательного вывода**

Иногда молчание действительно золото, и вывод команды нужно отбросить. В особенности это касается служебных сообщений и сообщений об ошибках. Система дает такую возможность, предоставляя специальный файл */dev/null*, куда можно перенаправить вывод. Этот файл представляет системное устройство, называемое битоприемником (bit bucket), или мусорной корзиной, которое принимает любой ввод и ничего с ним не делает. Чтобы подавить вывод сообщений об ошибках, достаточно проделать следующее:

[me@linuxbox ~]$ **ls -l /bin/usr 2> /dev/null**

**/DEV/NULL в культуре unix**

«Битоприемник» — старое понятие в Unix, благодаря своему универсализму широко используется в культуре Unix. Так, когда кто-то скажет, что посылает ваши комментарии в «dev null», вы теперь будете знать, что это означает. Еще больше примеров вы найдете в статье Википедии [*https://ru.wikipedia.org/wiki//dev/null*](http://https/ru.wikipedia.org/wiki/dev/null).

**Перенаправление стандартного ввода**

До сих пор нам не встречались команды, использующие стандартный ввод (на самом деле они встречались, но мы подробнее обсудим их чуть ниже), поэтому нам нужно познакомиться с ними.

**cat — объединение файлов**

Команда cat читает содержимое одного или нескольких файлов и копирует его в стандартный вывод:

cat [*файл*...]

Часто команду cat можно считать аналогом команды TYPE в DOS. Она используется для вывода содержимого файлов без возможности постраничного просмотра. Например,

[me@linuxbox ~]$ **cat ls-output.txt**

выведет содержимое файла *ls-output.txt*. Команда cat часто используется для вывода коротких текстовых файлов. Поскольку cat способна принимать сразу несколько файлов, она используется для их объединения. Представьте, что вы загрузили большой файл, разбитый на множество частей (в Usenet мультимедийные файлы часто разбиваются таким способом), и требуется объединить их в один файл. Если файлы имеют имена, такие как

movie.mpeg.001 movie.mpeg.002 ... movie.mpeg.099

их можно объединить следующей командой:

[me@linuxbox ~]$ **cat movie.mpeg.0\* > movie.mpeg**

Поскольку подстановка фактических имен взамен групповых символов всегда выполняется в порядке сортировки, аргументы окажутся расположенными в правильном порядке.

Все это прекрасно, но при чем здесь стандартный ввод? Пока ни при чем, но давайте попробуем кое-что еще. Что получится, если вызвать cat без аргументов?

[me@linuxbox ~]$ **cat**

Ничего не произошло — такое ощущение, что команда зависла. Однако в действительности команда делает именно то, что и предполагалось.

Если вызвать cat без аргументов, она начнет читать данные со стандартного ввода, а поскольку стандартный ввод по умолчанию подключен к клавиатуре, получается, что команда ждет, пока вы что-нибудь напечатаете!

Попробуйте так:

[me@linuxbox ~]$ **cat**

**Съешь ещё этих мягких французских булок, да выпей чаю.**

Затем нажмите комбинацию **CTRL-D** (то есть, удерживая нажатой клавишу **CTRL**, нажмите клавишу **D**), чтобы сообщить команде cat, что достигнут *конец файла* (*end-of-file*,*EOF*) на стандартном вводе:

[me@linuxbox ~]$ **cat**

**Съешь ещё этих мягких французских булок, да выпей чаю.**

Съешь ещё этих мягких французских булок, да выпей чаю.

В отсутствие аргументов с именами файлов cat копирует содержимое стандартного ввода в стандартный вывод, поэтому-то мы и увидели, как она повторила введенную нами строку. Эту ее особенность можно использовать для создания коротких текстовых файлов. Представьте, что вам потребовалось создать файл с именем *eat\_more.txt*, содержащий текст из примера, приведенного выше. Сделать это можно было бы так:

[me@linuxbox ~]$ **cat > eat\_more.txt**

**Съешь ещё этих мягких французских булок, да выпей чаю.**

Введите команду, затем текст, который нужно поместить в файл, и не забудьте нажать комбинацию **CTRL-D** в конце. Используя командную строку, мы реализовали самый простой в мире текстовый процессор! Чтобы увидеть результат, воспользуемся командой cat и скопируем файл в стандартный вывод:

[me@linuxbox ~]$ **cat eat\_more.txt**

Съешь ещё этих мягких французских булок, да выпей чаю.

Теперь, когда мы знаем, что команда cat может принимать данные не только из файлов, указанных в аргументах, но и со стандартного ввода, попробуем выполнить перенаправление стандартного ввода:

[me@linuxbox ~]$ **cat < eat\_more.txt**

Съешь ещё этих мягких французских булок, да выпей чаю.

Используя оператор перенаправления <, мы изменили источник данных для стандартного ввода с клавиатуры на файл *eat\_more.txt*. Как видите, результат получился тот же, как если бы мы просто передали единственный аргумент с именем файла. Этот способ не имеет никаких преимуществ в сравнении с передачей простого аргумента, но он демонстрирует, как можно использовать файлы в роли источника данных для стандартного ввода. Другие команды находят лучшее применение стандартному вводу, в чем мы вскоре убедимся.

Прежде чем двинуться дальше, прочитайте страницу справочного руководства (man) для команды cat, так как она имеет несколько очень интересных параметров.

**Конвейеры**

«Умение» команд читать данные со стандартного ввода и выводить результаты в стандартный вывод используется механизмом командной оболочки, который называется *конвейером*. C помощью оператора конвейера[6](http://10.207.106.8/os/pr/978549602303-9.xhtml#footnote-343439-6) | (вертикальная черта) стандартный вывод одной команды можно связать со стандартным вводом другой.

*команда1* | *команда2*

Для демонстрации этого механизма нам понадобится несколько команд. Мы уже упоминали команду, которая может получать данные со стандартного ввода. Это команда less. Теперь используем less для постраничного отображения вывода любой команды, которая посылает свои результаты в стандартный вывод:

[me@linuxbox ~]$ **ls -l /usr/bin | less**

Это очень удобно! С помощью этого приема можно со всем комфортом исследовать вывод любой команды, посылающей результаты на стандартный вывод.

**Фильтры**

Конвейеры часто используются для выполнения сложных операций с данными. Они позволяют объединить вместе несколько команд. Часто команды, используемые таким способом, называют *фильтрами*. Фильтры принимают ввод, изменяют его определенным образом и выводят результат. Первый из таких фильтров, который мы опробуем, — команда sort. Представьте, что нам необходимо составить список всех выполняемых программ в каталогах */bin* и */usr/bin*, расположив их по алфавиту, и затем вывести его:

[me@linuxbox ~]$ **ls /bin /usr/bin | sort | less**

Поскольку в команде указаны два каталога (*/bin* и */usr/bin*), вывод команды ls будет состоять из двух сортированных списков, по одному для каждого каталога. Добавив команду sort в конвейер, мы изменили данные, чтобы получить единый сортированный список.

**uniq — поиск или удаление повторяющихся строк**

Команда uniq часто используется в комбинации с командой sort. uniq принимает сортированный список данных либо со стандартного ввода, либо из файла, имя которого можно передать в единственном аргументе (за подробностями обращайтесь к странице справочного руководства (man) для команды uniq), и по умолчанию удаляет повторяющиеся строки из списка. Поэтому, чтобы гарантировать отсутствие дубликатов в нашем списке (то есть любых программ с одинаковыми именами в каталогах */bin* и */usr/bin*), добавим uniq в конвейер:

[me@linuxbox ~]$ **ls /bin /usr/bin | sort | uniq | less**

В этом примере мы использовали uniq для удаления любых повторяющихся строк в выводе команды sort. Если бы нам потребовалось, наоборот, получить список дубликатов, мы добавили бы в команду uniq параметр -d:

[me@linuxbox ~]$ **ls /bin /usr/bin | sort | uniq -d | less**

**wc — вывод числа строк, слов и байтов**

Команда wc (word count — счетчик слов) используется для подсчета числа строк, слов и байтов в файлах. Например:

[me@linuxbox ~]$ **wc ls-output.txt**

7902 64566 503634 ls-output.txt

В данном случае команда вывела три числа: число строк, число слов и число байтов в файле *ls-output.txt*. Подобно предыдущим командам, она может вызываться без аргументов, и в этом случае wc будет принимать данные со стандартного ввода. Параметр -l ограничивает вывод результатов только числом строк. Команду удобно использовать в конвейерах для подсчета: например, подсчитать число элементов в нашем сортированном списке можно так:

[me@linuxbox ~]$ **ls /bin /usr/bin | sort | uniq | wc -l**

2728

**grep — поиск строк, соответствующих шаблону**

grep — очень мощная программа, она часто используется для поиска в файлах текста по шаблону:

grep *шаблон* [*файл*...]

Когда grep находит в файле совпадение с «шаблоном», она выводит строку с найденным совпадением. Шаблоны, используемые командой grep для поиска, могут быть очень сложными, но сейчас мы рассмотрим только поиск прямого совпадения с текстом. Более сложные шаблоны, которые называют *регулярными выражениями*, мы рассмотрим в главе 19.

Допустим, что нам нужно найти все файлы в списке программ, которые имеют в своем имени последовательность символов zip. Результаты такого поиска могут подсказать нам, какие программы в системе имеют отношение к сжатию файлов. Сделать это можно так:

[me@linuxbox ~]$ **ls /bin /usr/bin | sort | uniq | grep zip**

bunzip2

bzip2

gunzip

gzip

unzip

zip

zipcloak

zipgrep

zipinfo

zipnote

zipsplit

Команда grep имеет пару удобных параметров: -i требует от grep игнорировать регистр символов в процессе поиска (обычно поиск выполняется с учетом регистра символов), -v требует от grep выводить только строки, где совпадение с шаблоном не найдено.

**head/tail — вывод первых/последних строк из файлов**

Иногда требуется выводить не все результаты работы команды, а только несколько первых или несколько последних строк. Команда head выводит первые 10 строк из файла, а tail — последние 10 строк. По умолчанию обе команды выводят 10 строк текста, но это число можно изменить с помощью параметра -n:

[me@linuxbox ~]$ **head -n 5 ls-output.txt**

total 343496

-rwxr-xr-x 1 root root       31316 2011-12-05 08:58 [

-rwxr-xr-x 1 root root        8240 2011-12-09 13:39 411toppm

-rwxr-xr-x 1 root root      111276 2011-11-26 14:27 a2p

-rwxr-xr-x 1 root root       25368 2010-10-06 20:16 a52dec

[me@linuxbox ~]$ **tail -n 5 ls-output.txt**

-rwxr-xr-x 1 root root        5234 2011-06-27 10:56 znew

-rwxr-xr-x 1 root root         691 2009-09-10 04:21 zonetab2pot.py

-rw-r--r-- 1 root root         930 2011-11-01 12:23 zonetab2pot.pyc

-rw-r--r-- 1 root root         930 2011-11-01 12:23 zonetab2pot.pyo

lrwxrwxrwx 1 root root           6 2012-01-31 05:22 zsoelim -> soelim

Их также можно использовать в конвейерах:

[me@linuxbox ~]$ **ls /usr/bin | tail -n 5**

znew

zonetab2pot.py

zonetab2pot.pyc

zonetab2pot.pyo

zsoelim

Команда tail позволяет наблюдать, как изменяется содержимое файла в режиме реального времени. Эту ее особенность удобно использовать для наблюдения за появлением новых записей в файлах журналов. В следующем примере демонстрируется наблюдение за файлом *messages* в каталоге */var/log*. В некоторых дистрибутивах Linux для этого требуется обладать привилегиями суперпользователя, поскольку файл */var/log/messages* может содержать секретную информацию.

[me@linuxbox ~]$ **tail -f /var/log/messages**

Feb 8 13:40:05 twin4 dhclient: DHCPACK from 192.168.1.1

Feb 8 13:40:05 twin4 dhclient: bound to 192.168.1.4 -- renewal in 1652 seconds.

Feb 8 13:55:32 twin4 mountd[3953]: /var/NFSv4/musicbox exported to both 192.168.1.0/24 and twin7.localdomain in 192.168.1.0/24,twin7.localdomain

Feb 8 14:07:37 twin4 dhclient: DHCPREQUEST on eth0 to 192.168.1.1 port 67

Feb 8 14:07:37 twin4 dhclient: DHCPACK from 192.168.1.1

Feb 8 14:07:37 twin4 dhclient: bound to 192.168.1.4 -- renewal in 1771 seconds.

Feb 8 14:09:56 twin4 smartd[3468]: Device: /dev/hda, SMART Prefailure Attribute: 8 Seek\_Time\_Performance changed from 237 to 236

Feb 8 14:10:37 twin4 mountd[3953]: /var/NFSv4/musicbox exported to both 192.168.1.0/24 and twin7.localdomain in 192.168.1.0/24,twin7.localdomain

Feb 8 14:25:07 twin4 sshd(pam\_unix)[29234]: session opened for user me by (uid=0)

Feb 8 14:25:36 twin4 su(pam\_unix)[29279]: session opened for user root by me(uid=500)

При вызове с параметром -f команда tail продолжает следить за файлом и при добавлении в конец этого файла новых строк немедленно выводит их. Так продолжается до тех пор, пока пользователь не нажмет комбинацию клавиш **CTRL-C**.

**tee — чтение со стандартного ввода и запись в стандартный вывод и в файлы**

Linux предоставляет команду tee, которая создает Т-образное разветвление в конвейере. Программа tee читает данные со стандартного ввода и копирует их в стандартный вывод (чтобы дать возможность передать их дальше по конвейеру) и в один или несколько файлов. Это может пригодиться для сохранения промежуточных результатов обработки в конвейере. Ниже, продолжая один из предыдущих примеров, мы сохраним полный список файлов в каталогах в файле *ls.txt*, перед тем как он будет отфильтрован командой grep:

[me@linuxbox ~]$ **ls /usr/bin | tee ls.txt | grep zip**

bunzip2

bzip2

gunzip

gzip

unzip

zip

zipcloak

zipgrep

zipinfo

zipnote

zipsplit

**linux развивает воображение**

Когда меня просят объяснить разницу между Windows и Linux, я часто привожу аналогию с игрушками.

Windows — это как игровая приставка Game Boy. Вы идете в магазин и покупаете новенькую сияющую приставку с игрой в комплекте. Приносите ее домой, включаете и играете. Отличная графика, чудные звуки. Но спустя некоторое время игра надоедает. Вы опять идете в магазин и покупаете другую игру. Так повторяется снова и снова. Наконец, вы возвращаетесь в магазин и говорите человеку за прилавком: «Я хочу игру, которая делает это!» — а в ответ слышите, что такой игры не существует, потому что на нее нет спроса. Тогда вы говорите: «Но мне нужно всего лишь изменить вот это!» А продавец за прилавком говорит, что это невозможно. Игры продаются зашитыми в картриджи. И тут вы понимаете, что ваша приставка ограничена кругом игр, при создании которых кто-то другой решил за вас, что вам нужно, а что нет.

Linux, напротив, можно сравнить с самым большим в мире конструктором. Вы открываете коробку и видите необозримую коллекцию деталей — огромное число железных планочек, болтиков, гаечек, шестеренок, колесиков и моторчиков и несколько рекомендаций по сборке. Вы начинаете играть. Сначала создаете один предлагаемый образец, затем другой. Затем вы обнаруживаете, что у вас появились собственные идеи новых конструкций и механизмов. И вам не нужно возвращаться в магазин, потому что у вас уже есть все, что требуется. Конструктор формирует ваше воображение. Он позволяет создать то, что вы хотите.

Выбор игрушки, конечно же, дело глубоко личное, но признайтесь честно: какая игрушка принесла бы вам большее удовлетворение?

**Заключительное замечание**

Как обычно, загляните в документацию с описанием каждой команды, представленной в этой главе. Здесь были показаны только самые общие примеры их использования, и все они имеют множество интересных параметров. По мере накопления опыта работы в Linux мы увидим, что поддержка перенаправления в командной оболочке чрезвычайно полезна для решения специализированных задач. Многие команды используют стандартный ввод и стандартный вывод, и почти все программы командной строки используют стандартный вывод ошибок для отображения информационных сообщений.

[6](http://10.207.106.8/os/pr/978549602303-9.xhtml#footnote-343439-6-backlink) Часто этот оператор называют также оператором канала. — *Примеч. пер.*