

Оглавление

Предисловие	10
Чему вы научитесь	11
Чем эта книга не является	11
Для кого эта книга	12
Ваша командная оболочка	13
Условные обозначения	13
Использование исходного кода примеров	14
Благодарности	15
От издательства	16

ЧАСТЬ 1 ОСНОВНЫЕ ПОНЯТИЯ

Глава 1. Объединение команд	18
Ввод, вывод и каналы	19
Шесть команд для начала	21
Команда #1: <code>wc</code>	21
Команда #2: <code>head</code>	24
Команда #3: <code>cut</code>	24
Команда #4: <code>grep</code>	26
Команда #5: <code>sort</code>	28
Команда #6: <code>uniq</code>	30
Обнаружение дубликатов файлов	32
Резюме	34
Глава 2. Знакомство с командной оболочкой	35
Терминология командной оболочки	36
Сопоставление шаблонов имен файлов	37
Вычисление переменных	40
Откуда берутся значения переменных	40
Переменные и заблуждения	41
Шаблоны vs переменные	42
Сокращение команд с помощью псевдонимов	43
Перенаправление ввода и вывода	44
Отключение вычисления с помощью кавычек и экранирования	47

Расположение исполняемых программ	49
Окружение и файлы инициализации, краткая версия.....	50
Резюме.....	52
Глава 3. Повторный запуск команд	53
Просмотр истории команд	54
Повторный вызов команд из истории	55
Перемещение курсора по истории команд	55
Расширение истории команд.....	57
Забудьте об ошибочном удалении файлов (спасибо расширению истории)	60
Инкрементальный поиск по истории команд.....	62
Редактирование командной строки.....	64
Перемещение курсора внутри команды	65
Расширение истории с помощью знака вставки	65
Редактирование командной строки в стилях Emacs или Vim	67
Резюме.....	69
Глава 4. Перемещение по файловой системе	70
Лучшие способы перехода в нужный каталог	71
Переход в домашний каталог	71
Перемещайтесь быстрее с автозавершением командной строки	72
Переход к часто посещаемым каталогам с использованием псевдонимов или переменных	73
Уменьшите пространство поиска с помощью CDPATH	75
Организуйте свой домашний каталог для быстрой навигации	77
Лучшие способы вернуться в каталог	79
Переключение между двумя каталогами с помощью «cd -»	79
Переключение между несколькими каталогами с помощью pushd и popd.....	80
Резюме.....	86
 ЧАСТЬ 2 ПРОДВИНУТЫЕ НАВЫКИ	
Глава 5. Расширяем ваш инструментарий.....	88
Создание текста.....	89
Команда date.....	89
Команда seq.....	90
Расширение команд с помощью фигурных скобок	91
Команда find.....	93

Команда <code>yes</code>	94
Извлечение текста	95
Команда <code>grep</code> . Более глубокий взгляд.....	96
Команда <code>tail</code>	100
Команда <code>awk {print}</code>	101
Объединение текста	103
Команда <code>tac</code>	104
Команда <code>paste</code>	104
Команда <code>diff</code>	105
Преобразование текста	106
Команда <code>tr</code>	107
Команда <code>rev</code>	107
Команды <code>awk</code> и <code>sed</code>	108
Как расширить инструментарий	117
Резюме	119
Глава 6. Родители, потомки и окружение	120
Оболочки — это исполняемые файлы	121
Родительский и дочерний процессы	123
Переменные окружения	125
Создание переменных окружения	126
Предупреждение о мифе: «глобальные» переменные	127
Дочерние оболочки vs подоболочки.....	128
Настройка окружения	129
Повторное считывание файла конфигурации	132
Путешествие с вашим окружением.....	132
Резюме	133
Глава 7. Еще 11 способов запуска команды	134
Способы, использующие списки	134
Способ #1. Условные списки.....	135
Способ #2. Безусловные списки	137
Способы, использующие подстановку	137
Способ #3. Подстановка команд.....	138
Способ #4. Подстановка процесса.....	140
Команда как строка	143
Способ #5. Передача команды в <code>bash</code> в качестве аргумента.....	144
Способ #6. Передача команды в <code>bash</code> через стандартный ввод.....	145
Способ #7. Удаленное выполнение однострочника с помощью <code>ssh</code>	147
Способ #8. Запуск списка команд с помощью <code>xargs</code>	148

Способы, использующие управление процессами	153
Способ #9. Фоновое выполнение команды.....	153
Способ #10. Явные подболочки.....	159
Способ #11. Замена процесса	161
Резюме.....	162
Глава 8. Создание дерзких однострочников.....	164
Приготовьтесь быть дерзкими	166
Будьте гибкими	166
Подумайте, с чего начать	167
Изучите инструменты тестирования	169
Вставка имени файла в последовательность	169
Проверка совпадающих пар файлов.....	172
Создание CDPATH из вашего домашнего каталога	175
Создание тестовых файлов.....	177
Создание пустых файлов	180
Резюме.....	181
Глава 9. Использование текстовых файлов	182
Первый пример: поиск файлов.....	184
Проверка срока действия домена.....	185
Создание базы данных телефонных кодов.....	188
Создание менеджера паролей.....	190
Резюме	197

ЧАСТЬ 3

ДОПОЛНИТЕЛЬНЫЕ ПЛЮСЫ

Глава 10. Эффективное использование клавиатуры	200
Работа с окнами.....	200
Мгновенный запуск оболочек и браузера.....	201
Одноразовые окна	202
Горячие клавиши в браузере	202
Переключение окон и рабочих столов	203
Доступ в интернет из командной строки	204
Запуск окон браузера из командной строки	204
Получение HTML-страниц с помощью curl и wget	207
Обработка кода HTML с помощью HTML-XML-utils	208
Получение и отображение содержимого веб-сайтов с помощью текстового браузера.....	212

Управление буфером обмена из командной строки	213
Подключение буферов обмена к stdin и stdout.....	214
Улучшение работы менеджера паролей	216
Резюме.....	219
Глава 11. Финальные советы по экономии времени	220
Способы решения задач легко и быстро.....	220
Переход в текстовый редактор напрямую из команды less	220
Редактирование файлов, содержащих заданную строку	221
Смиритесь с опечатками	222
Быстрое создание пустых файлов	222
Обработка файла построчно.....	223
Список команд, поддерживающих рекурсию	223
Читайте справочные страницы.....	224
Способы решения задач, требующие затрат времени на изучение	224
Прочтите справочную страницу команды bash	224
Изучите команды cron, crontab и at.....	225
Изучите команду rsync	226
Изучите другой язык для написания сценариев.....	227
Используйте make для задач, не связанных с программированием.....	228
Применяйте контроль версий к повседневным файлам	230
Прощание.....	231
Приложение А. Памятка по Linux.....	233
Команды, аргументы и параметры.....	233
Файловая система, каталоги и пути	234
Перемещение по каталогам	236
Создание и редактирование файлов	236
Работа с файлами и каталогами	237
Просмотр файлов.....	239
Права доступа к файлам.....	239
Процессы.....	240
Просмотр документации	241
Сценарии оболочки	242
Получение привилегий суперпользователя	243
Дополнительная литература.....	244
Приложение Б. Если вы используете не bash.....	245
Об авторе	251
Иллюстрация на обложке.....	252

Предисловие

Эта книга позволит перейти на новый уровень использования командной строки Linux, чтобы вы могли работать быстрее, умнее и эффективнее.

Если вы похожи на большинство пользователей Linux, то приобрели первоначальные навыки использования командной строки на работе, или прочитав вводную книгу, или установив Linux дома и просто попробовав. Моя книга должна помочь вам сделать следующий шаг — развить навыки работы с командной строкой Linux от среднего до продвинутого уровня. Она наполнена методами и концепциями, которые, я надеюсь, изменят ваше взаимодействие с Linux и повысят производительность. Воспринимайте ее как вторую книгу по использованию Linux, которая выведет вас за рамки азов.

Командная строка — самый простой и вместе с тем самый сложный из интерфейсов. Он прост, потому что содержит только приглашение командной строки, которое ждет ваших действий¹:

```
$
```

Он сложный, потому что за все, что выходит за рамки приглашения, отвечаете только вы. Нет понятных значков, кнопок или меню, которые могли бы помочь. Вместо этого каждая введенная команда является результатом ваших усилий. Это верно как для основных команд, таких, например, как вывод списка файлов:

```
$ ls
```

так и для более сложных, например:

```
$ paste <(echo {1..10}.jpg | sed 's/ /\n/g') \  
        <(echo {0..9}.jpg | sed 's/ /\n/g') \  
      | sed 's/^/mv /' \  
      | bash
```

Если вы смотрите на предыдущую команду и думаете: «*Что это*, черт возьми?» или «Мне бы никогда не понадобилась такая сложная команда», то эта книга для вас².

¹ Для отображения приглашения командной строки в этой книге используется символ \$. В вашей операционной системе этот символ может быть другим.

² Вы узнаете назначение этой таинственной команды в главе 8.

Чему вы научитесь

Эта книга поможет вам быстро и эффективно освоить три основных навыка:

- Выбор и создание команд для решения текущей задачи.
- Эффективный запуск команд.
- Удобная навигация по файловой системе Linux.

Прочитав книгу, вы будете понимать, что остается за кадром при запуске команды, и научитесь лучше предсказывать ее результаты (а не руководствоваться суевериями). Вы познакомитесь с дюжиной разных способов запуска команд и узнаете, когда лучше использовать каждый из них. Вы также узнаете приемы и хитрости, которые сделают вашу работу продуктивнее. Например:

- Построение сложных команд из простых, шаг за шагом, для решения практических задач, например управления паролями или создания тысяч тестовых файлов.
- Экономия времени за счет продуманной организации домашнего каталога, чтобы вам не приходилось искать нужные файлы.
- Преобразование текстовых файлов и получение из них данных для решения поставленных задач.
- Управление из командной строки теми операциями в Linux, которые обычно выполняются мышкой. Например, копирование и вставка с помощью буфера обмена или получение и обработка веб-данных. И все это, не отрывая рук от клавиатуры.

И самое главное, вы изучите лучшие практические методы работы в командной строке, поэтому независимо от того, какие команды вы запускаете, вы сможете добиться большего в повседневном использовании Linux и стать более конкурентоспособными на рынке труда. Я бы хотел иметь эту книгу, когда сам только начинал изучение Linux.

Чем эта книга не является

Эта книга не поможет оптимизировать и не сделает эффективнее ваш компьютер с установленной операционной системой Linux. Она лишь сделает *вас* более эффективными при использовании Linux.

Эта книга также не является исчерпывающим справочником по командной строке. Существуют сотни команд и функций, которые здесь не упомянуты. Эта книга об опыте. Она содержит тщательно отобранный набор практических знаний о командной строке для развития ваших навыков. В качестве справочного руководства используйте мою предыдущую книгу *Linux Pocket Guide* (издательство O'Reilly).

Для кого эта книга

Предполагается, что у вас есть опыт работы с Linux, поскольку *эта книга не знакомит с Linux*. Она предназначена, прежде всего, для пользователей, которые хотят улучшить свои навыки работы с командной строкой, таких как студенты, системные администраторы, разработчики программного обеспечения, IT-инженеры, тестировщики и энтузиасты операционной системы Linux. Опытные пользователи Linux также смогут найти для себя полезный материал, особенно если они учились методом проб и ошибок и хотят улучшить свое понимание командной строки.

Чтобы извлечь максимальную пользу из этой книги, вы уже должны быть знакомы со следующими темами (если нет, см. приложение А, где дается их краткий обзор):

- Создание и редактирование текстовых файлов с помощью редакторов `vim (vi)`, `emacs`, `nano`, `pico`.
- Основные команды для работы с файлами, такие как `cp` (копирование), `mv` (перемещение или переименование), `rm` (удаление), `chmod` (изменение прав доступа).
- Основные команды для просмотра файлов, такие как `cat` (вывод содержимого файла) и `less` (постраничный вывод текста).
- Основные команды для работы с каталогами, такие как `cd` (переход в другой каталог), `ls` (вывод списка файлов и каталогов), `mkdir` (создание каталога), `rmdir` (удаление каталога), `pwd` (вывод пути к текущему каталогу).
- Основы сценариев командной оболочки: хранение команд Linux в файле, создание исполняемого файла (с помощью `chmod 755` или `chmod +x`) и его запуск.
- Просмотр встроенной документации Linux, известной как ман-страницы, с помощью команды `man` (например, `man cat` отображает документацию по команде `cat`).
- Умение стать суперпользователем с помощью команды `sudo` для полного доступа к операционной системе (например, команда `sudo nano /etc/hosts` позволяет редактировать файл `/etc/hosts`, изменение которого запрещено обычному пользователю).

Если вы также знакомы с общими функциями командной строки, такими как сопоставление имен файлов с шаблоном (символы `*` и `?`), перенаправление ввода/вывода (`<` и `>`) и каналы (`|`), то вы уже хорошо подготовлены к изучению этой книги.

Ваша командная оболочка

Предполагается, что вашей командной оболочкой является `bash`, которая используется по умолчанию в большинстве дистрибутивов Linux. Под термином «оболочка» в тексте имеется в виду именно `bash`. Большинство идей, представленных в книге, применимы и к другим оболочкам, таким как `zsh` или `dash` (см. Приложение Б для перевода примеров в другие оболочки). Большая часть материала будет работать без изменений и в Apple Mac-терминале, который по умолчанию запускает `zsh`, но также может запускать и `bash`¹.

Условные обозначения

В этой книге используются следующие условные обозначения.

Курсив

Курсивом выделены новые термины и важные понятия, а также имена файлов и каталогов.

Моноширинный шрифт

Используется для листингов программ, а также внутри абзацев для обозначения таких элементов, как переменные и функции, базы данных, типы данных, переменные среды, операторы и ключевые слова, имена файлов и их расширения.

Моноширинный полужирный шрифт

Показывает команды или другой текст, который пользователь должен ввести самостоятельно.

Моноширинный курсив

Показывает текст, который должен быть заменен значениями, введенными пользователем или определяемыми контекстом. Кроме того, этот шрифт используется для кратких пояснений в правой части листингов.

Моноширинный шрифт в рамке

Используется для привлечения внимания к отдельным частям листингов.

Шрифт без засечек

Используется для обозначения URL, адресов электронной почты, названий кнопок и других элементов интерфейса.

¹ Версия `bash` для macOS устарела, в ней отсутствуют важные функции. Чтобы обновить `bash`, см. статью Дэниела Вейбеля (Daniel Weibel) *Upgrading Bash on macOS* (<https://oreil.ly/35jux>).



Этот рисунок указывает на совет или предложение.



Этот рисунок указывает на примечание.



Этот рисунок указывает на предупреждение.

Использование исходного кода примеров

Вспомогательные материалы (примеры кода, упражнения и т. д.) доступны для загрузки по адресу: <https://efficientlinux.com/examples>. Если у вас возникнут вопросы технического характера по использованию примеров кода, направляйте их по электронной почте на адрес bookquestions@oreilly.com.

В общем случае все примеры кода из книги вы можете использовать в своих программах и в документации. Вам не нужно обращаться в издательство за разрешением, если вы не собираетесь воспроизводить существенные части программного кода. Если вы разрабатываете программу и используете в ней несколько фрагментов кода из книги, вам не нужно обращаться за разрешением. Но для продажи или распространения примеров из книги вам потребуется разрешение от издательства O'Reilly. Вы можете отвечать на вопросы, цитируя данную книгу или примеры из нее, но для включения существенных объемов программного кода из книги в документацию вашего продукта потребуется разрешение.

Мы рекомендуем, но не требуем добавлять ссылку на первоисточник при цитировании. Под ссылкой на первоисточник мы подразумеваем указание авторов, издательства и ISBN.

За разрешением на использование значительных объемов программного кода из книги обращайтесь по адресу permissions@oreilly.com.

Благодарности

Эту книгу было приятно писать. Спасибо замечательным людям из издательства O'Reilly, особенно редакторам Вирджинии Уилсон (Virginia Wilson) и Джону Девинсу (John Devins), редакторам производства Кейтлин Геган (Caitlin Ghegan) и Грегори Хайману (Gregory Human), контент-менеджеру Кристен Браун (Kristen Brown), редактору Киму Уимпсетту (Kim Wimpsett), редактору предметного указателя Сюю Клефстад (Sue Klefstad) и всегда готовой помочь команде производственного отдела. Я также очень благодарен рецензентам книги Полу Байеру (Paul Bayer), Джону Бонезио (John Bonesio), Дэну Риттеру (Dan Ritter) и Карле Шредер (Carla Schroder) за ценные комментарии и критические замечания. Также спасибо Boston Linux Users Group за предложения по названию. Особая благодарность Мэгги Джонсон (Maggie Johnson) из Google за ее любезное разрешение написать книгу.

Я бы хотел выразить глубочайшую благодарность Чипу Эндрюсу (Chip Andrews), Мэтью Диазу (Matthew Diaz) и Роберту Странду (Robert Strandh), которые 35 лет назад учились со мной в Университете Джонса Хопкинса (The Johns Hopkins University). Они заметили мой растущий интерес к операционной системе Unix и, к моему крайнему удивлению, порекомендовали Департаменту компьютерных наук (Computer Science Department) нанять меня в качестве системного администратора. Их вера в меня изменила траекторию моей жизни (кстати, благодаря Роберту добавлен совет касательно навыков слепой печати в главе 3). Спасибо также тем, кто создает и поддерживает проекты Linux, GNU Emacs, Git, AsciiDoc и многие другие инструменты с открытым исходным кодом — без этих умных и щедрых людей моя карьера действительно была бы совсем другой.

Как всегда, спасибо моей замечательной семье, Лизе и Софии, за их любовь и терпение.

От издательства

Ваши замечания, предложения, вопросы отправляйте по адресу comp@piter.com (издательство «Питер», компьютерная редакция).

Мы будем рады узнать ваше мнение!

На веб-сайте издательства www.piter.com вы найдете подробную информацию о наших книгах.

ЧАСТЬ 1

Основные понятия

Первые четыре главы должны быстро повысить эффективность вашей работы в командной строке, ознакомив вас с принципами и методами, которые можно сразу применить на практике. Вы научитесь комбинировать команды с помощью каналов, поймете функции командной оболочки Linux, сможете повторно вызывать и редактировать команды, которые ранее уже запускали, и быстро перемещаться по файловой системе Linux.

ГЛАВА 1

Объединение команд

В Windows, macOS и большинстве других операционных систем вы тратите время на запуск таких приложений, как веб-браузеры, текстовые редакторы, программы для работы с таблицами и игры. Типичное приложение содержит множество функций: все, что, по мнению разработчиков, может понадобиться пользователям. Поэтому большинство приложений самодостаточно и не связаны с другими приложениями. Вы можете время от времени копировать и вставлять данные из одного приложения в другое, но по большей части они независимы друг от друга.

Командная строка в Linux работает по-другому. Вместо больших приложений с множеством функций Linux предоставляет тысячи небольших команд с ограниченным набором возможностей. Команда `cat`, например, выводит на экран содержимое файлов и больше ничего. `ls` отображает список файлов в каталоге, `mv` переименовывает файлы и т. д. Каждая команда имеет простую, четко определенную цель.

Что делать, если вам нужно произвести более сложные действия? Не волнуйтесь, Linux позволяет легко *объединять команды*, чтобы их функции работали вместе для достижения поставленной цели. Такой способ порождает совсем другое отношение к работе за компьютером. Вместо того чтобы спрашивать себя «Какое приложение мне запустить?» для достижения необходимого результата, возникает вопрос: «Какие команды мне объединить?»

В этой главе вы узнаете, как упорядочивать и запускать команды в различных комбинациях, чтобы решать поставленные задачи. Для простоты начнем с шести команд Linux и их основных способов применения. Это позволит сосредоточиться на более сложной и интересной части — их объединении — без долгого обучения. Это все равно, что учиться готовить, используя только шесть ингредиентов, или осваивать столярное дело с применением только молотка и пилы (дополнительные команды добавятся в ваш набор в главе 5).

Вы будете объединять команды, используя *канал (pipe)* — функцию Linux, которая соединяет вывод одной команды с вводом другой. При представлении каждой команды (`wc`, `head`, `cut`, `grep`, `sort` и `uniq`) мы сразу же будем рассматривать пример использования канала. Некоторые примеры будут полезны

для повседневной работы в Linux, в то время как другие являются искусственно выдуманными для демонстрации важной функции.

Ввод, вывод и каналы

Большинство команд Linux считают ввод с клавиатуры, отображают вывод на экране или делают и то, и другое. В Linux эти операции чтения и записи имеют собственные названия:

stdin (стандартный ввод, или стандартный входной поток)

Поток данных, который Linux считывает при вводе с клавиатуры. Когда вы вводите любую команду в терминале после приглашения командной строки, вы передаете данные в стандартный входной поток *stdin*.

stdout (стандартный вывод, или стандартный выходной поток)

Поток данных, который Linux выводит на ваш дисплей. Когда вы запускаете команду `ls` для печати имен файлов, результаты появляются в стандартном выходном потоке *stdout*.

Теперь самое интересное. Вы можете подключить стандартный вывод одной команды к стандартному вводу другой, чтобы первая команда передавала свои выходные данные во вторую. Давайте начнем со знакомой команды `ls -l` для просмотра большого каталога, такого как `/bin`, в длинном формате:

```
$ ls -l /bin
total 12104
-rwxr-xr-x 1 root root 1113504 Jun  6  2019 bash
-rwxr-xr-x 1 root root 170456 Sep 21  2019 BSD-csh
-rwxr-xr-x 1 root root 34888 Jul  4  2019 bunzip2
-rwxr-xr-x 1 root root 2062296 Sep 18  2020 busybox
-rwxr-xr-x 1 root root 34888 Jul  4  2019 bzip2
:
-rwxr-xr-x 1 root root 5047 Apr 27  2017 znew
```

Этот каталог содержит значительно больше файлов, чем, строк на вашем дисплее, поэтому вывод быстро прокручивается за пределы экрана. Жаль, что `ls` не может печатать информацию по одному экрану за раз, делая паузу, пока вы не нажмете клавишу для продолжения. Но постойте: у другой команды Linux есть такая возможность. Команда `less` отображает файл по одному экрану за раз:

```
$ less myfile
```

Вы можете соединить эти две команды, потому что `ls` передает данные в стандартный вывод, а `less` может считывать данные из стандартного ввода. Используем канал для отправки выходных данных `ls` на вход команды `less`:

```
$ ls -l /bin | less
```

ЧТО ТАКОЕ КОМАНДА?

Слово *команда* имеет три различных значения в Linux, представленные на рис. 1.1:

Программа

Исполняемая программа, названная и выполняемая одним словом, например `ls`, или аналогичная функция, встроенная в оболочку, например `cd` (*встроенная команда оболочки*)¹.

Простая команда

Имя программы или встроенной команды оболочки, за которым могут следовать аргументы, например `ls -l /bin`.

Комбинированная команда

Несколько простых команд, рассматриваемых как единое целое, например конвейер `ls -l /bin | less`

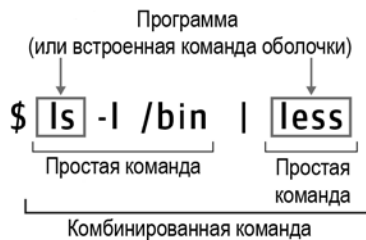


Рис. 1.1. Программы, простые команды и комбинированные команды называются «командами»

В этой книге слово «команда» используется во всех этих смыслах. Обычно из контекста понятно, что имеется в виду, но если нет, будем использовать один из более конкретных терминов.

Эта объединенная команда отображает содержимое каталога по одному экрану за раз. Вертикальная черта (|) между командами — это символ² канала в Linux. Он соединяет стандартный вывод первой команды со стандартным вводом следующей. Командная строка, содержащая каналы, называется *конвейером* (*pipeline*).

¹ Стандарт POSIX называет такие команды *утилитами*.

² На стандартной клавиатуре символ вертикальной черты находится на той же клавише, что и обратная косая черта (\), обычно расположенная между клавишами **Enter** и **Backspace** или между левой клавишей **Shift** и **Z**.

Команды обычно не знают, что они являются частью конвейера. `ls` считает, что выводит данные на дисплей, хотя на самом деле ее вывод был перенаправлен на `less`. А `less` верит, что читает данные с клавиатуры, когда на самом деле получает вывод `ls`.

Шесть команд для начала

Использование каналов — неотъемлемая часть работы с Linux. Давайте развивать ваши навыки работы с каналами с помощью небольшого набора команд Linux. Когда вы столкнетесь с какими-то из этих команд в будущем, то будете готовы их объединять.

Команды `wc`, `head`, `cut`, `grep`, `sort` и `uniq` имеют множество опций и режимов работы, про которые пока умолчим, чтобы сосредоточиться на каналах. Если хотите узнать больше о конкретной команде, запустите команду `man`, которая отобразит полную документацию. Например:

```
$ man wc
```

Для демонстрации этих шести команд в действии будем использовать файл с именем *animals.txt*, в котором перечислены некоторые сведения из книг издательства O'Reilly. Содержимое файла представлено в примере 1.1.

Пример 1.1. Содержимое файла animals.txt

python	Programming Python	2010	Lutz, Mark
snail	SSH, The Secure Shell	2005	Barrett, Daniel
alpaca	Intermediate Perl	2012	Schwartz, Randal
robin	MySQL High Availability	2014	Bell, Charles
horse	Linux in a Nutshell	2009	Siever, Ellen
donkey	Cisco IOS in a Nutshell	2005	Boney, James
oryx	Writing Word Macros	1999	Roman, Steven

Каждая строка содержит четыре факта о какой-либо книге издательства O'Reilly, разделенные одним символом табуляции: животное на обложке, название книги, год публикации и имя первого автора.

Команда #1: `wc`

Команда `wc` выводит количество строк, слов и символов в файле:

```
$ wc animals.txt
 7 51 325 animals.txt
```

`wc` сообщает, что в файле *animals.txt* 7 строк, 51 слово и 325 символов. Если вы посчитаете символы, включая пробелы и табуляции, то найдете только 318 символов, но `wc` также учитывает скрытые символы новой строки.

Опции `-l`, `-w` и `-c` указывают `wc` печатать только количество строк, слов и символов соответственно:

```
$ wc -l animals.txt
7 animals.txt
$ wc -w animals.txt
51 animals.txt
$ wc -c animals.txt
325 animals.txt
```

Вывод параметров текста — настолько полезная и часто встречающаяся задача, что авторы `wc` разработали свою команду специально для работы с каналами. Если вы не указываете имя файла, она читает данные из стандартного ввода и отправляет данные в стандартный вывод. Давайте воспользуемся `ls` для вывода содержимого текущего каталога и используем `wc` для подсчета строк. Этот конвейер отвечает на вопрос «Сколько файлов в моем текущем каталоге?»:

```
$ ls -l
animals.txt
myfile
myfile2
test.py
$ ls -l | wc -l
4
```

Опция `-l`, указывающая `ls` выводить результаты в один столбец, здесь не обязательна. Чтобы узнать, почему мы ее использовали, см. врезку «`ls` меняет свое поведение при перенаправлении вывода» на с. 23.

`wc` — это первая команда, с которой мы познакомились в этой главе, поэтому пока сложно использовать каналы. Ради интереса направим вывод `wc` самой себе, чтобы продемонстрировать, что одна и та же команда может появляться в конвейере более одного раза. Эта комбинированная команда сообщает, что количество слов в выводе `wc` равно четырем (три целых числа и имя файла):

```
$ wc animals.txt
 7 51 325 animals.txt
$ wc animals.txt | wc -w
4
```

Не будем останавливаться. Добавим третий вызов `wc` в конвейер и посчитаем количество строк, слов и символов в выводе 4:

```
$ wc animals.txt | wc -w | wc
 1      1      2
```

LS МЕНЯЕТ СВОЕ ПОВЕДЕНИЕ ПРИ ПЕРЕНАПРАВЛЕНИИ ВЫВОДА

В отличие от большинства других команд Linux, `ls` знает, выводит ли она данные на экран или перенаправляет в канал. Когда `stdout` — это экран, `ls` упорядочивает вывод в несколько колонок для удобства чтения:

```
$ ls /bin
bash      dir      kmod      networkctl  red      tar
bsd-csh   dmesg    less      nisdomainname  rm      tempfile
:
```

При перенаправлении `stdout` `ls` создает одну колонку. Убедимся в этом, передав вывод `ls` команде, которая воспроизводит свой ввод, такой как `cat`¹:

```
$ ls /bin | cat
bash
bsd-csh
bunzip2
busybox
:
```

Это может привести к неожиданным результатам, как в следующем примере:

```
$ ls
animals.txt myfile myfile2 test.py
$ ls | wc -l
4
```

Первая команда `ls` выводит все имена файлов в одну строку, а вторая сообщает, что выводятся четыре строки. Если вы не знаете об особых свойствах `ls`, это несоответствие может удивить.

`ls` позволяет переопределить свое поведение по умолчанию. Вы можете заставить `ls` вывести на экран одну колонку с помощью параметра `-1` или несколько колонок с параметром `-C`.

Вывод указывает на одну строку (содержащую число 4), одно слово (само число 4) и два символа. Почему два? Потому что строка 4 заканчивается скрытым символом новой строки.

Думаю, что достаточно простейших конвейеров с `wc`. По мере того как мы изучим больше команд, конвейеры станут более практичными.

¹ В зависимости от ваших настроек `ls` может использовать также другие функции форматирования, например изменение цвета при выводе на экране. Но не при перенаправлении вывода.

Команда #2: head

Команда `head` выводит первые строки файла. Выведем первые три строки файла *animals.txt*, используя `head` с параметром `-n`:

```
$ head -n3 animals.txt
python  Programming Python      2010    Lutz, Mark
snail   SSH, The Secure Shell     2005    Barrett, Daniel
alpaca  Intermediate Perl           2012    Schwartz, Randal
```

Если вы запрашиваете больше строк, чем содержится в файле, `head` выведет весь файл (как это делает `cat`). Если вы опустите параметр `-n`, заголовок по умолчанию будет состоять из 10 строк (`-n10`).

Сама по себе функция `head` удобна для просмотра начала файла, когда вас не интересует остальное содержимое. Эта команда работает быстро и эффективно даже с очень большими файлами, поскольку ей не нужно считывать весь файл. Кроме того, `head` умеет считывать данные из стандартного ввода и использует стандартный вывод, что делает ее полезной в конвейерах. Подсчитаем количество слов в первых трех строках файла *animals.txt*:

```
$ head -n3 animals.txt | wc -w
20
```

Обычное использование `head` заключается в сокращении вывода данных от другой команды, когда вам не нужно видеть полную информацию, например длинный список каталогов. Перечислим первые пять имен файлов в каталоге */bin*:

```
$ ls /bin | head -n5
bash
bsd-csh
bunzip2
busybox
bzipcat
```

Команда #3: cut

Команда `cut` выводит одну или несколько колонок из файла. Например, выведем все названия книг, которые расположены во второй колонке файла *animals.txt*:

```
$ cut -f2 animals.txt
Programming Python
SSH, The Secure Shell
Intermediate Perl
MySQL High Availability
Linux in a Nutshell
```