ВНУТРЕННЕЕ УСТРОЙСТВО LINUX

Дмитрий Кетов

bhv®

Дмитрий Кетов

ВНУТРЕННЕЕ УСТРОЙСТВО LINUX

Санкт-Петербург
«БХВ-Петербург»
2017

УДК 004.451
ББК 32.973.26-018.2
К37

Кетов Д. В.

К37 Внутреннее устройство Linux. — СПб.: БХВ-Петербург, 2017. — 320 с.: ил.

ISBN 978-5-9775-3580-9

Книга представляет собой введение во внутреннее устройство операционной системы Linux. Все положения наглядно проиллюстрированы примерами, разработанными автором и проверенными им на практике. Рассмотрены основные подсистемы ядра и их сущности — файлы и файловые системы, виртуальная память и отображаемые файлы, процессы, нити и средства межпроцессного взаимодействия, каналы, сокеты и разделяемая память. Раскрыты дискреционный и мандатный (принудительный) механизмы контроля доступа, а также привилегии процессов. Подробно описано пользовательское окружение и интерфейс командной строки CLI, оконная система X Window и графический интерфейс GUI, а также сетевая подсистема и служба SSH. Особое внимание уделено языку командного интерпретатора и его использованию для автоматизации задач эксплуатации операционной системы.

Для студентов, пользователей, программистов
и системных администраторов Linux

УДК 004.451
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Екатерина Капалыгина*
Редактор	*Анна Кузьмина*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн обложки	*Марины Дамбиевой*

Подписано в печать 30.09.16.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 25,8.
Тираж 1500 экз. Заказ №
"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

Первая Академическая типография "Наука"
199034, Санкт-Петербург, 9 линия, 12/28

ISBN 978-5-9775-3580-9

Оглавление

Введение

О чем эта книга?

Книга, которую вы держите в руках, адресована начинающим пользователям[1] операционной системы Linux и представляет собой ***иллюстрированное*** введение в ее ***внутреннее устройство*** — от ядра до сетевых служб и от утилит командной строки до графического интерфейса.

По моему скромному мнению, никакие книги не могут превратить читателя из новичка в эрудированного специалиста — ни обзорные, рассказывающие «ничего обо всем», ни узкоспециализированные, повествующие «все об одном». Без самостоятельного опыта, получаемого в результате исследования происходящих процессов и явлений, осмысление любого книжного знания практически невозможно.

Именно поэтому данная книга преследует всего одну важную цель — привить читателю ***навыки*** самостоятельного ***исследования*** постоянно ***эволюционирующей*** операционной системы Linux через ***умения*** пользоваться соответствующим инструментарием и на основе теоретических ***знаний*** о ее устройстве и философии[2].

Порядок и стиль изложения материала, иллюстративные изображения и листинги, приведенные в книге, являются результатом обобщения моего достаточно долгого опыта преподавания технических дисциплин, посвященных операционным системам в целом и операционной системе Linux в частности. Именно такой мне представляется картина «правильного» минимального набора необходимых знаний, умений и навыков, необходимых для построения качественной ***ментальной модели*** современной ОС Linux.

---

[1] Хотя термин «пользователь» зачастую ассоциируется с этаким домашним любителем или офисным работником, но никак не с IT-специалистом, книга адресована абсолютно всем, кому интересно понимать то, как устроена операционная система изнутри. Именно для ее компетентного применения в своей профессиональной деятельности такое понимание в первую очередь нужно программистам и системным администраторам — разработчикам и эксплуатационникам информационных систем на базе Linux.

[2] **W**:[Философия UNIX].

Основная задача книги — иллюстративно изложить внутреннее устройство операционной системы Linux, связав «***теорию***» и «***практику***». В отличие от многих изданий, скупо и сухо излагающих теоретические аспекты построения операционной системы или излишне концентрирующихся на деталях конкретной реализации того или иного ее программного обеспечения, в книге рассматриваются ***абстрактные*** концепты внутреннего устройства ОС, иллюстративно подкрепляемые примерами анализа (а иногда и синтеза) ее ***конкретных*** компонент и связей между ними.

Все части операционной системы рассматриваются в контексте типичных задач, решаемых при их помощи на практике, а сама иллюстрация проводится при помощи соответствующего инструментария пользователя, администратора и разработчика.

Многие концепты и компоненты, инструменты и задачи будут иметь ***общность***, характерную для всех **W**:[дистрибутивов Linux][1], но их реализации могут существенно ***различаться***. Поэтому, преследуя цель сохранить практическую значимость и жертвуя некоторой потерей общности излагаемого материала, повествование ведется в контексте отдельно взятой операционной системы, в качестве которой выступает дистрибутив **W**:[Ubuntu Linux]. Читатель может ***свободно*** скачать этот дистрибутив из Интернета и использовать в абсолютно любых целях, например, для проработки примеров, приведенных в книге.

Выбор в пользу дистрибутива **W**:[Ubuntu Linux], в частности, сделан еще и по совокупности присущих ему свойств, как то: большое сообщество его пользователей (включая русскоговорящих), громадная пакетная база, регулярность обновлений и их стабильность, дружественность к начинающим и, наконец, широкая распространенность технических решений на его основе — от бизнес-решений и до сертифицированных разработок в оборонной промышленности.

Кому адресована книга?

Понимание внутреннего устройства важно как в системном администрировании для направленной диагностики проблем эксплуатации информационных систем, построенных на платформе Linux, так и для разработки программного обеспечения, учитывающего особенности ее внутреннего устройства.

Поэтому книга адресуется абсолютно всем, кто не любит использовать непрозрачный «черный ящик», кому интересна механика всех этих шестеренок, что вращаются внутри Linux, ежедневно решая задачи миллионов пользователей по всему

[1] А зачастую и для всех операционных систем семейства **W**:[UNIX], к которому Linux принадлежит.

миру. Всем тем, кто в детстве разобрал не один игрушечный луноход в попытке узнать, как там чувствуют себя человечки в скафандрах.

Принятые соглашения и обозначения

Наиболее полезным навыком при исследовании любой незнакомой информационной системы является способность оперативно получать справочную информацию непосредственно изнутри ее, не прибегая к внешним источникам (вроде поиска по Google). Именно поэтому в данной книге *нигде* не встречаются даже частичные цитирования содержания справочных систем.

Классическим электронным справочником UNIX являются страницы руководства (**man**ual pages, подробнее *см. главу 2*), которые сгруппированы в тематические секции, а ссылки на страницу *page* в секции *N* принято записывать как **page(N)**, в том числе и в этой книге. Другая полезная информация, в том числе историческая, доступна из статей всемирной онлайн-энциклопедии Wikipedia, поэтому полезные ссылки на статью *useful article* изображаются как **W:[useful article]**.

Разнообразные значки [☛☝ ? 🗣 🗣 🗝 ❶...❾①...⑨ ♞ ⇆ ⇊ 🚶] на полях и в теле листингов позволяют акцентировать внимание на *нужном* и сопровождать *важное* дополнительными комментариями в тексте повествования.

Полужирный шрифт **command** в листингах идентифицирует вводимые пользователем команды или управляющие символы и последовательности, а нажатия на клавиши или их комбинации изображаются как Ctrl Alt F4. Курсив используется для выделения *понятий* и *терминов*, а полужирный курсив выделяет ***ключевые слова***, определяющие основное существо текста абзаца.

Значок указательного пальца ☛ и ☝ и вот такое «выделение маркером» укажет на те места в листингах, куда нужно обратить внимание. Номера ❶...❾①...⑨ и звездочка ★ позволят сослаться на конкретные места листингов в основном тексте.

Вопрос **?** отметит те места листингов, которые должны вызывать недоумение вида «а это еще что?», а знак восклицания ! укажет на места, которые должны сопровождаться возгласами «ох тыж! ничего себе!». На особенно эмоциональные места листингов указывает значок 🗣, а поднятый вверх большой палец ☝, как и обычно, одобряет определенные команды листингов, которые приводят к долгожданному результату.

Аналогично, закрытый замок 🔒 символизирует закрытость (заблокированность) или отсутствие доступа, а открытый замок 🔓, наоборот, открытость (разблокированность), наличие доступа. Ключ 🗝 указывает на операцию открытия/закрытия чего-либо (например, ввод пароля при аутентификации).

Кинжал 🗡 указывает на выполнение некоторой деструктивной операции, а значки часов 🕐🕑🕒🕓 символизируют ожидание завершения достаточно длительной операции.

Для удобства чтения листингов значки ⇆⇅ указывают на «замещающий» вывод (т. е. без скроллинга), значок ↻ обозначает очередную итерацию некоторого цикла, а значок ⊘ указывает на его окончание. Ну и, наконец, разнообразные стрелки ↵▸→←↑ связывают отдельные части листинга друг с другом.

Методические рекомендации

Многие иллюстративные листинги, приводящиеся в книге, нужно воспринимать не как «дополнение» к основному тексту, а как сам основной текст, просто на (пока) незнакомом и непонятном языке, а саму книгу — как учебник нового иностранного языка.

Получение читателем собственного отклика от взаимодействия с операционной системой представляется мне наиболее значимым результатом освоения содержимого этой книги. Поэтому чтение стоит непременно сопровождать самостоятельным выполнением команд листингов, т. к. успешное изучение нового иностранного языка никак невозможно без его непосредственного практического использования в процессе изучения.

Не менее важным для построения целостных взаимоотношений с системой является самостоятельное изучение первоисточников знаний[1] — ее внутренней документации, страниц руководства man(1).

Для выполнения команд листингов и доступа к внутренним справочникам потребуется сама операционная система, что несложно получить, например, инсталлировав настольную версию Ubuntu Linux в виртуальную машину W:[VirtualBox].

Дополнительную пользу принесет ознакомление и с дополнительными источниками информации в Интернете, например со статьями на http://help.ubuntu.ru, а также поиск ответов на возникающие вопросы на http://forum.ubunu.ru, http://askubuntu.com и даже на http://UNIX.stackexchange.com и http://stackoverflow.com.

Кроме текстовых источников информации в Интернете можно воспользоваться услугами таких очаровательных онлайн-сервисов, как http://explainshell.com для объяснений конструкций и http://www.shellcheck.net для проверки проблем в конструкциях командного интерпретатора. Для проверки примеров из книги и разработки собственных сценариев командного интерпретатора незаменимую помощь может оказать сервис http://www.tutorialspoint.com/execute_bash_online.php.

[1] Это обязательно потребует знания технического английского языка, но без него вообще трудно рассчитывать на сколько-нибудь значительные успехи в отрасли информационных технологий.

Что должен знать читатель

В операционной системе Linux программное обеспечение поставляется в виде так называемых пакетов (package) — специальным образом подготовленных архивов, содержащих само *программное* обеспечение, его *конфигурационные* файлы, его данные и *управляющую* информацию. Управляющая информация пакета включает контрольные суммы устанавливаемых файлов, зависимости устанавливаемого пакета от других пакетов, краткое описание пакета, сценарии установки, сценарии удаления пакета и прочие данные, необходимые *менеджеру* пакетов.

В примерах, приведенных в листингах, часто встречаются программы, которые, возможно, не будут установлены «по умолчанию», поэтому важно знать и понимать, как устроена подсистема управления установкой и удалением программного обеспечения — так называемые менеджеры пакетов и зависимостей[1].

Менеджер пакетов производит непосредственную установку и удаление пакетов программного обеспечения, а также ведет их учет в системе. Вспомогательная «грязная» работа по подбору зависящих друг от друга пакетов, получению их из *репозиториев*[2] (например, скачивание с FTP/HTTP-серверов), выбору правильных версий пакетов, определению правильного их порядка установки достается *менеджеру зависимостей*.

Листинг B1. Пакетный менеджер dpkg

```
bart@ubuntu:~$ which date
/bin/date
❶ bart@ubuntu:~$ dpkg -S /bin/date
coreutils: /bin/date
❷ bart@ubuntu:~$ dpkg -L coreutils
...                 ...             ...             ...             ...
/bin/ln
/bin/touch
/bin/date
/bin/mkdir
/bin/sty
...                 ...             ...             ...             ...
```

[1] Условно, можно выделить две ветви операционной системы Linux — ветвь debian, к которой относятся дистрибутивы **W**:[Debian] и **W**:[Ubuntu], и ветвь redhat, куда нужно отнести **W**:[RHEL], **W**:[CentOS] и **W**:[Fedora]. В debian-ветви используется пакетный менеджер **dpkg** и построенные над ним менеджеры зависимостей **apt**, **aptitude**, **synaptic** и **software-center**, а в ветви redhat — пакетный менеджер **rpm** и основной менеджер зависимостей **yum**.

[2] Хранилище пакетов у разработчиков дистрибутива Linux.

```
bart@ubuntu:~$ dpkg -s coreutils
Package: coreutils
Essential: yes
Status: install ok installed
...                  ...             ...             ...             ...
Version: 8.13-3ubuntu3.2
Replaces: mktemp, timeout
Depends: dpkg (>= 1.15.4) | install-info
Pre-Depends: libacl1 (>= 2.2.51-5), libattr1 (>= 1:2.4.46-5), libc6 (>= 2.15), libselinux1
(>= 1.32)
Conflicts: timeout
Description: GNU core utilities
 This package contains the basic file, shell and text manipulation
 utilities which are expected to exist on every operating system.
 .
 Specifically, this package includes:
 arch base64 basename cat chcon chgrp chmod chown chroot cksum comm cp
                      ...                        ...                        ...
 users vdir wc who whoami yes
Homepage: http://gnu.org/software/coreutils
Original-Maintainer: Michael Stone <mstone@debian.org>
```

При помощи менеджера пакетов (листинг B1) всегда можно узнать имя пакета ❶, в который входит та или иная компонента операционной системы (например, **/bin/date**), или, наоборот, узнать список компонент ❷, установленных из указанного пакета (например, **coreutils**).

Если при попытке выполнить ту или иную команду операционной системы Ubuntu Linux (это одна из причин, по которой иллюстрация в книге ведется именно с ее помощью) обнаружится, что нужный пакет с программным обеспечением не установлен, то при наличии доступа в Интернет можно тривиальным способом доустановить недостающие компоненты (листинг B2).

Листинг B2. Установка недостающего программного обеспечения

```
bart@ubuntu:~$ mail
Программа 'mail' на данный момент не установлена. Вы можете установить её, выполнив:
sudo apt-get install mailutils
bart@ubuntu:~$ sudo apt-get install mailutils
[sudo] password for bart:
Чтение списков пакетов... Готово
```

```
Построение дерева зависимостей
Чтение информации о состоянии... Готово
Предлагаемые пакеты:
  mailutils-mh
НОВЫЕ пакеты, которые будут установлены:
  mailutils
обновлено 0, установлено 1 новых пакетов, для удаления отмечено 0 пакетов, и 567 пакетов
не обновлено.
Необходимо скачать 401 kБ архивов.
После данной операции объём занятого дискового пространства возрастёт на 1 149 kB.
Получено:1 http://archive.ubuntu.com/ubuntu/ precise/universe mailutils i386 1:2.2+dfsg1-5
[401 kB]
Получено 401 kБ за 0с (846 kБ/с)
Выбор ранее не выбранного пакета mailutils.
(Чтение базы данных ... на данный момент установлено 281689 файлов и каталогов.)
Распаковывается пакет mailutils (из файла .../mailutils_1%3a2.2+dfsg1-5_i386.deb)...
Обрабатываются триггеры для man-db ...
Настраивается пакет mailutils (1:2.2+dfsg1-5) ...
                              ...                     ...                     ...
```

Совет для начинающих

И напоследок, самый важный совет для начинающих — начните!

Глава 1

Архитектура ОС Linux

1.1. Обзор внутреннего устройства

Операционная система (ОС), в общем, и **W**:[Linux], в частности (рис. 1.1), представляет собой набор специализированных *программных* средств, обеспечивающих *доступ* потребителей (пользователей) к ресурсам (устройствам) и *распределение* последних между потребителями наиболее *удобным* и *эффективным* способом. Под ресурсами в первую очередь понимают аппаратные средства, такие как центральные процессоры, память, устройства ввода-вывода, дисковые и прочие накопители и другие периферийные устройства. Во вторую очередь к ресурсам относят такие «виртуальные» сущности (на рис. 1.1 не показаны), как процессы, нити, файлы, каналы и сокеты, семафоры и мьютексы, окна и прочие артефакты самой операционной системы, которые не существуют вне ее границ.

Как и во многих других операционных системах, в Linux выделяют два главных режима работы ее программных средств — ядерный режим (kernel mode), он же пространство ядра (kernel space), и пользовательский, внеядерный режим (user mode), или же пользовательское пространство (user space). Основное отличие этих двух режимов состоит в привилегиях доступа к аппаратным средствам — памяти и устройствам ввода-вывода, к которым разрешен полный доступ из режима ядра и ограниченный доступ из режима пользователя.

Совокупность работающих в ядерном режиме программ называют *ядром*, которое в Linux состоит из основы (или же, остова) и присоединяемых к ней объектов — динамически загружаемых модулей (подробнее *см. разд. 4.1.1*). Несмотря на компонентность, **W**:[Ядро Linux] относят к классу *монолитных* в силу того, что все его компоненты выполняются с *одинаковыми* (ядерными) привилегиями. В классе *микроядерных* систем, наоборот, компоненты ядра работают с *разными* привилегиями: основная компонента — микроядро (планировщик процессов/нитей и менеджер памяти) — с наибольшими привилегиями, менеджер ввода-вывода, драйверы устройств, файловые системы, сетевые протоколы и пр. — с другими, обычно меньшими привилегиями (возможно даже, с привилегиями пользовательского режима).

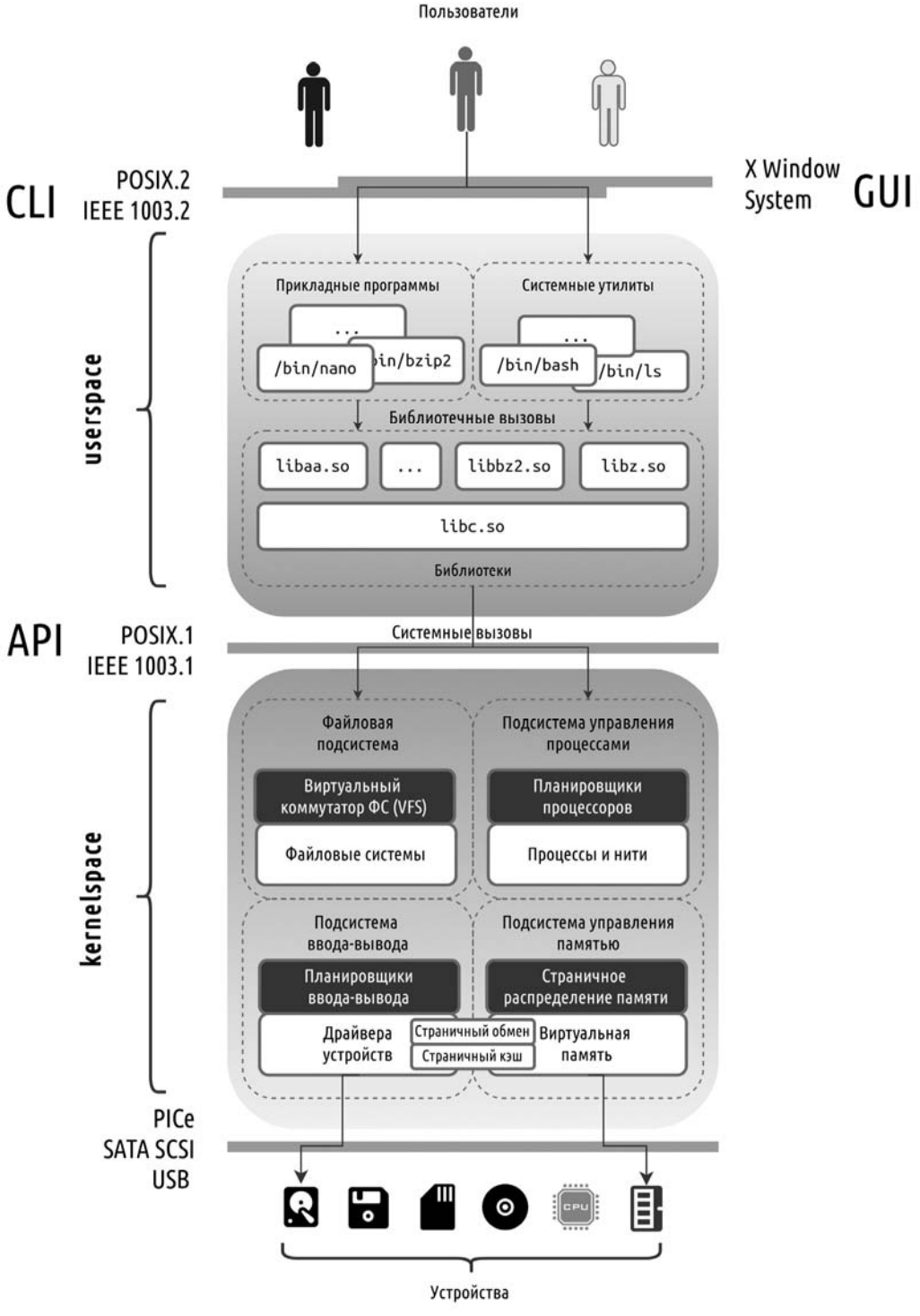

Рис. 1.1. Компоненты операционной системы Linux

1.2. Внеядерные компоненты: программы и библиотеки

Ядерные компоненты в основном обеспечивают решение задачи диспетчеризации (распределения) ресурсов между потребителями и предоставляют им базовый интерфейс доступа к ресурсам. Решение задачи обеспечения удобства доступа реализуется компонентами внеядерного режима — библиотеками динамической и статической компоновки (подробнее *см. разд. 4.1*).

Функции операционной системы, реализуемые ядерными компонентами, доступны внеядерным компонентам посредством *системных вызовов* — специализированных наборов обращений для получения услуг ядра. Системные вызовы выполняются в ядре, а вызываются при помощи основной внеядерной компоненты — библиотеки **libc.so** языка программирования **W**:[Си] (так исторически сложилось в силу того, что бóльшая часть системных программных средств написана на этом языке).

Функции, реализуемые внеядерными компонентами, доступны посредством *библиотечных вызовов* и вызываются/выполняются в самих библиотеках (например, алгоритмы сжатия информации в библиотеках **libz.so** и **libbz2.so**).

1.3. Ядерные компоненты: подсистемы управления процессами, памятью, вводом-выводом, файлами

Одна из основных задач ядра — распределение ресурсов между потребителями — вполне естественным образом приводит к тому, что среди ядерных компонент выделяют соответствующие (аппаратным ресурсам) менеджеры, так называемые подсистемы управления *процессами*, *памятью*, *вводом-выводом* и *файловую* подсистему.

Подсистема управления *процессами* распределяет время центральных процессоров (ЦП) между выполняющимися задачами (т. е. реализует **W**:[многозадачность]). Она создает и уничтожает такие сущности, как *процессы* и *нити (см. разд. 4.2)*, и организует одновременное (параллельное или псевдопараллельное) их выполнение при помощи планировщиков (англ. *scheduler*), реализующих алгоритмы распределения процессорного времени.

Подсистема *ввода-вывода* распределяет доступ к устройствам ввода-вывода (УВВ) между процессами и предоставляет им унифицированные интерфейсы *блочного, символьного (см. разд. 3.2.5)* и *пакетного* (сетевого) устройств *(см. разд. 6.1)*. Для устройств *внешней* памяти (дисковых или твердотельных накопителей, более медленных по сравнению с оперативной памятью) подсистема ввода-вывода организует **W**:[кэширование] при помощи подсистемы управления памятью.

Подсистема управления *памятью* (подробнее *см. разд. 4.7*) распределяет пространство оперативного запоминающего устройства (ОЗУ) между процессами при помо-

щи механизма страничного отображения — *выделяет* (и высвобождает) процессам страничные кадры физической памяти и *отображает* на страницы их адресного пространства. Кроме того, эта подсистема организует **W**:[виртуальную память] за счет механизма *страничного обмена* (**W**:[подкачка страниц]) — *вытесняет* неиспользуемые страницы процессов во внешнюю память и *загружает* их обратно по требованию при помощи подсистемы ввода-вывода.

Стоит отметить, что распределение ресурсов процессоров и устройств ввода-вывода происходит «во времени», т. е. в отдельные промежутки времени эти устройства выполняют операции только *одного* процесса или нити. Наоборот, распределение ресурсов запоминающих устройств происходит «в пространстве», т. к. информация *нескольких* процессов одновременно размещается в разных их *областях*.

Файловая подсистема ядра (подробнее *см. главу 3*) предоставляет процессам унифицированный интерфейс *файлового* доступа к внешней памяти (внешним запоминающим устройствам, ВЗУ — магнитным дисковым, твердотельным накопителям и т. п.) и распределяет между ними пространство ВЗУ при помощи *файлов* и *файловых систем*. Особенное назначение файловой подсистемы состоит еще и в том, что при помощи ее *файлового интерфейса* процессам предоставляется доступ и к другим подсистемам. Так, доступ к устройствам ввода-вывода организуется посредством специальных файлов устройств (блочных и символьных), например к CD/DVD-накопителю — через файл **/dev/sr0**, а к манипулятору мыши — через **/dev/input/mouse0**. Доступ к физической памяти и памяти ядра ОС организуется через файлы виртуальных устройств **/dev/mem** и **/dev/kmem**, а доступ процессов к страницам памяти друг друга — через файлы **/proc/PID/mem** псевдофайловой системы **procfs**. Даже доступ к внутренним параметрам и статистике ядра операционной системы возможен через файлы псевдофайловой системы **procfs**, а к списку обнаруженных ядром устройств ввода-вывода — через файлы псевдофайловой системы **sysfs**.

Кроме всех вышеперечисленных задач, файловая подсистема, подсистема ввода-вывода и подсистема управления процессами в совокупности предоставляют процессам средства межпроцессного взаимодействия, такие как сигналы *(см. разд. 4.8)*, каналы, сокеты и разделяемая память *(см. разд. 4.9)*.

1.4. Трассировка системных и библиотечных вызовов

Для наблюдения за обращениями программ к услугам ядерных компонент операционной системы, т. е. за системными вызовами, служит утилита strace(1), предназначенная для трассировки — построения трасс выполнения той или иной программы. В листинге 1.1 представлена трассировка программы date(1) относительно

системных вызовов времени time(2), gettimeofday(2), settimeofday(2) и clock_gettime(2), clock_settime(2), где оказывается, что для получения и установки значения системного времени программа date(1) использует системные вызовы clock_gettime(2) и clock_settime(2).

Листинг 1.1. Трассировщик системных вызовов strace

```
$ date
Пн. янв. 11 01:44:17 MSK 2016

$ strace -fe time,gettimeofday,clock_gettime date
clock_gettime(CLOCK_REALTIME, {1452465309, 213946474}) = 0
Пн. янв. 11 01:44:26 MSK 2016

$ strace -fe settimeofday,clock_settime date -s 10:00
clock_settime(CLOCK_REALTIME, {1452495600, 0}) = -1 EPERM (Operation not permitted)
date: невозможно установить дату: Операция не позволяется
Пн. янв. 11 10:00:00 MSK 2016
```

Листинг 1.2. Трассировщик системных вызовов ltrace

```
$ ltrace  -e localtime,asctime,ctime,strftime date
localtime(0xbf818448)                           = 0xb77c97e0
strftime(" \320\237\320\275.", 1024, " %a", 0xb77c97e0) = 6
strftime(" \321\217\320\275\320\262.", 1024, " %b", 0xb77c97e0) = 8
Пн. янв. 11 01:45:12 MSK 2016
+++ exited (status 0) +++
```

1.5. Интерфейсы прикладного программирования

Системные и библиотечные вызовы Linux, формирующие интерфейс прикладного программирования (API, **a**pplication **p**rogramming **i**nterface), соответствуют определенным промышленным спецификациям, в частности (практически идентичным друг другу) стандартам **W**:[POSIX][1], Portable Operating System Interface и SUS, **W**:[Single UNIX Specification], доставшимся «в наследство» от семейства операционных систем UNIX, членом которого Linux и является.

[1] Стандарт POSIX выпускается комитетом 1003 организации **W**:[IEEE], поэтому имеет формальное обозначение IEEE 1003, а части стандарта POSIX.1 и POSIX.2 формально обозначаются IEEE 1003.1 и IEEE 1003.2 соответственно.

Стандарт POSIX условно делится на две части: POSIX.1, программный интерфейс (API) операционной системы, и POSIX.2, интерфейс командной строки (CLI) пользователя *(см. главу 2)* и командный интерпретатор *(см. главу 5)*.

1.6. В заключение

Совершенно очевидно, что в современной медицине без понимания анатомии живых организмов невозможно представить ни их терапию, ни хирургию. В информационных технологиях абсолютно аналогичным образом разработка и эксплуатация программного обеспечения будут успешны только на основе понимания внутреннего устройства операционной системы.

Рисунок 1.1 изображает эдакий «рентгеновский снимок» внутренностей операционной системы Linux, подробная анатомия которой шаг за шагом и раскрывается в последующих главах.

Глава 2

Пользовательское окружение ОС Linux

2.1. Командный интерфейс

Основным интерфейсом взаимодействия между ЭВМ и человеком в классической операционной системе **W**:[UNIX] был единственно возможный, диктуемый аппаратными устройствами ее времени[1], *командный интерфейс*. Называемый сегодня интерфейсом командной строки (Command Line Interface, **W**:[CLI]), он в неизменном виде сохранил все свои элементы — понятие *терминала*, двусторонний попеременный диалог при помощи клавиши ↵ENTER, управляющие символы и клавишу CTRL для их набора.

Терминал является оконечным (терминальным) оборудованием, предназначенным для организации человеко-машинного интерфейса, и состоит из устройств вывода — принтера или дисплея, и устройств ввода — клавиатуры, манипулятора «мышь» и пр.

Алфавитно-цифровой терминал позволяет вводить и выводить символы из некоторого заданного набора (например, семибитной кодировки ascii(7)) или другого набора символов charsets(7)), состоящего из букв алфавита, цифр, знаков препинания, некоторых других значков, и символов специального назначения для управления самим терминалом — *управляющих символов*.

Ранние, *печатающие терминалы*, представляли собой телетайпы **W**:[телетайп] (телепринтеры **W**:[teleprinter]), которые печатали символы из фиксированного набора на ленте или рулоне бумаги — слева направо, сверху вниз. Управляющие символы использовались для управления перемещением печатающей головки справа налево (символ **BS**), возврата головки к началу строки (символ **CR**), прокрутки рулона бумаги (символ **LF**) и пр.

Дисплейный терминал (видеотерминал на основе электронно-лучевой трубки) в упрощенном своем режиме эмулирует поведение печатающего терминала: пово-

[1] ЭВМ **W**:[PDP-11] и терминалы **W**:[Teletype Model ASR-33].

рачивающийся рулон бумаги — при помощи *скроллинга* изображаемых строк снизу вверх, а перемещающуюся вдоль строки печатающую головку — при помощи *курсора*. В расширенном режиме видеотерминал используется как матрица символов, например в 24 строки по 80 символов в строке, и позволяет выводить символы в произвольное место матрицы, задавать символам стиль изображения, как то: мерцание, жирность, инвертирование, подчеркивание и цвет, и даже менять шрифты символов терминала. Для управления курсором, его позиционирования, смены стиля изображения символов и прочих возможностей видеотерминала применяются управляющие символы *(см. разд. 2.3)* и управляющие последовательности *(см. разд. 2.4)*.

Двусторонний попеременный диалог (рис. 2.1) командного интерфейса между пользователем и операционной системой представляет собой процесс ввода команд ❶ пользователем посредством клавиатуры и получения результата их выполнения ❷ на бумаге или дисплее алфавитно-цифрового терминала.

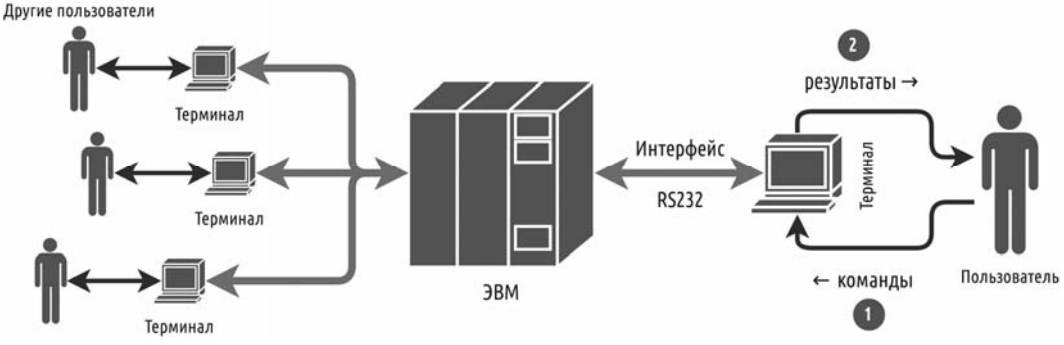

Рис. 2.1. Терминалы и командный интерфейс

В начале сеанса работы в многопользовательской среде операционной системы пользователь должен произвести регистрацию (**log**ging **in**) себя в системе (обычно говорят «произвести вход» в систему) при помощи предъявления имени своей учетной записи (login) и соответствующего и известного ему пароля (password, буквально — пропускное **pass** слово **word**) (рис. 2.2).

Процедура регистрации начинается с заставки операционной системы ❶ и приглашения к вводу имени учетной записи ❷, в ответ на которое пользователь вводит имя своей учетной записи ①. Затем, в ответ на приглашение к вводу пароля ❸, пользователь вводит пароль ② и при этом на алфавитно-цифровых терминалах никакие символы не изображаются. При положительном исходе регистрации пользователь получает сообщение ❹ о последней (**last**) успешной регистрации, сообщение дня и приглашение командного интерпретатора ❺.

Передача управления от пользователя к операционной системе на каждом шаге диалога происходит при помощи нажатия клавиши ↵ ENTER, а передача управления

в обратную сторону при помощи *приглашений* к вводу регистрационного имени, пароля, командного интерпретатора и пр. Приглашение командного интерпретатора исторически состояло из символа **$** или символа **%**, а при регистрации под учетной записью администратора — из символа **#**. Позднее приглашение развилось в **finn@ubuntu:~$** и состоит теперь из имени зарегистрировавшегося пользователя **finn**, собственного имени компьютера **ubuntu**, условного имени домашнего каталога пользователя, обозначенного символом **~**, и «классического» символа приглашения **$**.

Рис. 2.2. Регистрация пользователя в системе

Сеанс командного интерфейса пользователя продолжается двусторонним попеременным диалогом с командным интерпретатором, где пользователь вводит команды ③ и получает результаты их выполнения ❻.

2.2. Виртуальные терминалы

На текущий момент времени многопользовательские системы с настоящими физическими терминалами, подключенными посредством интерфейса RS232 и его драйвера ttyS(4) к *большой* ЭВМ, — экзотическая редкость. На *персональных* ЭВМ для взаимодействия с пользователем имеются стандартные клавиатура, видеоадаптер и монитор, формирующие так называемую **W**:[консоль], которая используется драйвером *виртуальных терминалов* для эмуляции нескольких физических терминалов (рис. 2.3).

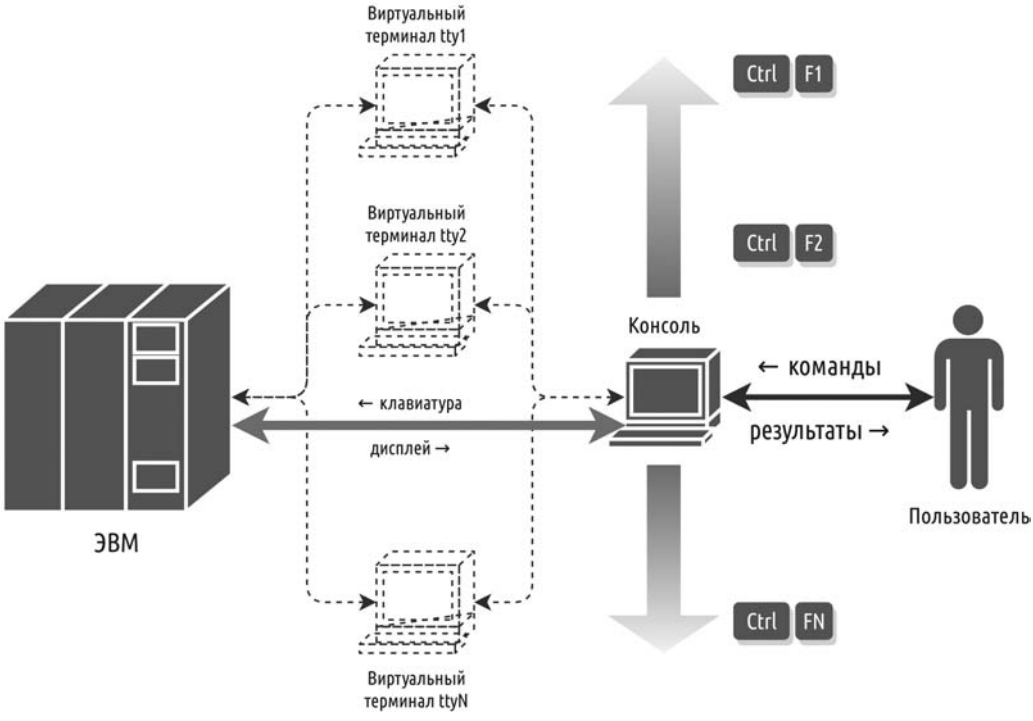

Рис. 2.3. Виртуальные терминалы

Узнать имя текущего терминала (а точнее — имя специального файла устройства[1] терминального драйвера, см. листинг 2.1), на котором выполнен вход в систему, позволяет команда tty(1), а список всех терминальных входов пользователей — команды users(1), who(1) и w(1).

Листинг 2.1. Утилиты tty, users, who и w

```
finn@ubuntu:~$ tty ↵
/dev/tty1
finn@ubuntu:~$ users ↵
bubblegum finn iceking jake jake marceline
finn@ubuntu:~$ who ↵
iceking  tty4         2015-03-27 10:46
bubblegum tty5        2015-03-27 10:46
marceline tty3        2015-03-27 10:47
finn     tty1         2015-03-27 10:46
jake     tty7         2015-03-24 23:46
```

[1] Подробнее о специальных файлах устройств *см. в разд. 3.2.5*.

```
    jake     pts/3        2015-03-27 10:47 (:0.0)
    finn@ubuntu:~$ w ↵
     10:47:52 up 2 days, 11:02,  9 users,  load average: 0,05, 0,18, 0,27
    USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
    iceking  tty4                     10:46    1:20   0.05s  0.00s -sh
    bubblegu tty5                     10:46    1:04   0.04s  0.00s -sh
    marcelin tty3                     10:47   32.00s  0.06s  0.00s -sh
❶   finn     tty1                     10:46    1:04   0.50s  0.44s w
❷   jake     tty7                     Tue23    2days 25:36   2.88s gnome-session -
❸   jake     pts/3    :0.0            10:47    0.00s  0.21s  0.00s -bash
```

Драйвер виртуального терминала позволяет переключаться между эмулируемыми терминалами при помощи сочетания клавиш Ctrl F1, ..., Ctrl F12 (первые двенадцать терминалов из 63 возможных), Alt ← для переключения на предыдущий, Alt → для переключения на следующий виртуальный терминал. При переключении из *графического* виртуального терминала на другой виртуальный терминал необходимо добавлять к сочетанию еще и клавишу Ctrl, т. к. сочетания с клавишей Alt востребованы самим графическим интерфейсом, например Alt F4 закрывает активное окно. Таким образом, для переключения из графического на третий виртуальный терминал используется сочетание Alt Ctrl F3. Также, драйвер виртуальных терминалов позволяет листать буфер вывода виртуального терминала при помощи сочетаний Ctrl PgUp и Ctrl PgDn (к сожалению, после переключения терминалов буфер пропадает).

Как и любым другим, драйвером виртуальных терминалов можно управлять при помощи специально предназначенных команд, например, программа chvt(1) позволяет переключаться на заданный терминал по его номеру, а команда программы setfont(1) — загружать шрифты, формирующие начертания алфавитно-цифровых знаков (см. листинг 2.12).

2.2.1. Псевдотерминалы

При работе в оконной системе X Window System *(см. главу 7)* используются графические терминалы, тогда как для командного интерфейса требуется алфавитно-цифровой терминал. В этом случае (рис. 2.4) он эмулируется при помощи драйвера *псевдотерминала* pty(4) (**p**seudo t**ty**) и приложения-посредника — эмулятора терминала (например, **xterm** или **gnome-terminal**), который связывает действительный обмен ① в графическом окне с мультиплексором псевдотерминалов ② ptmx(4) (**p**seudo**t**erminal **m**ultiple**x**er), а тот, в свою очередь, присоединен драйвером к подчиненному псевдотерминалу pts(4) (**p**seudo**t**erminal **s**lave) командного интерфейса.

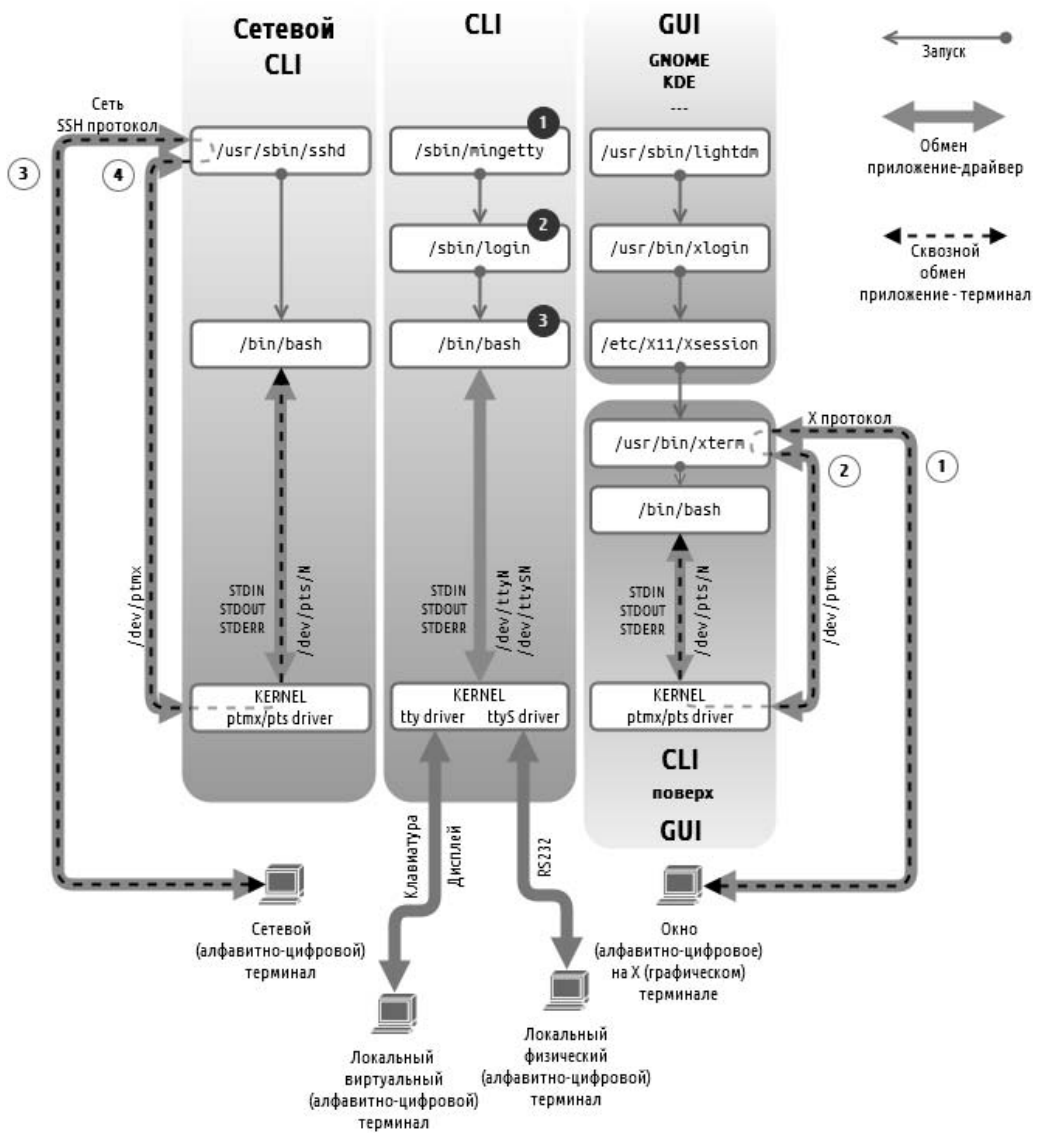

Рис. 2.4. Псевдотерминалы

Таким образом, командный интерпретатор и запускаемые им программы работают с воображаемым псевдотерминалом так, как будто окно графического приложения является настоящим физическим дисплеем и настоящей физической клавиатурой настоящего терминала. В примере из листинга 2.1 пользователь **finn** зарегистрирован в системе на первом виртуальном терминале **tty1**, а пользователь **jake** зарегистрирован на седьмом (графическом) виртуальном терминале **tty7** в оконной системе и работает с командным интерфейсом на третьем псевдотерминале **pts/3** в окне эмулятора терминала.

Аналогично, при подключении удаленного алфавитно-цифрового терминала посредством протоколов удаленного доступа (например, SSH) приложение-посредник (например, демон сетевой службы **sshd**, *см. разд. 6.4.1*) связывает действительный обмен ③ в сетевом соединении с мультиплексором псевдотерминалов ④.

Нужно заметить, что на этапе входа пользователя в систему посредством алфавитно-цифрового терминала (см. средний фрагмент с меткой **CLI** на рис. 2.4) последовательно запускаются: обработчик терминалов ❶, обработчик аутентификации и авторизации пользователей ❷, а затем командный интерпретатор ❸, например bash(1). Именно getty(1) предъявляет пользователю (см. рис. 2.2) заставку операционной системы и приглашение к вводу имени пользователя, а login(1) — приглашение к вводу пароля, сообщение о последнем успешном входе и сообщение дня.

Аналогичные процессы происходят при любом входе пользователя в систему, например через псевдотерминалы «графического» или «сетевого» доступа (см. левый и правый фрагменты на рис. 2.4). В любом случае после аутентификации и авторизации основной программой (и первой в сеансе пользователя), интерпретирующей вводимые пользователем команды, является командный интерпретатор.

2.3. Управляющие символы

При *вводе* с терминала управляющие символы (табл. 2.1) служат командами драйверу терминала и в большинстве своем генерируются при помощи сочетания клавиш Ctrl (отсюда ее название control — управление) с одной из алфавитно-цифровых клавиш. В отдельных случаях, управляющие символы генерируются специально предназначенными для этого клавишами, например ↵ Enter, ⇆ Tab или ⇤ Backspace.

Таблица 2.1. Управляющие символы терминала

Управляющий символ			Клавиши	Код символа	
Нотация	Стандартное действие драйвера				
	ввод символа	вывод символа			
^C	intr	—	Ctrl C	0x03	ETX
^\	quit	—	Ctrl \ или Ctrl 4	0x1C	FS
^Z	susp	—	Ctrl Z	0x1A	SUB
^D	eof	—	Ctrl D	0x04	EOT
^?	erase	—	⇤ Backspace или Ctrl ? или Ctrl 8	0x7F	DEL

Таблица 2.1 *(окончание)*

Управляющий символ			Клавиши	Код символа	
Нотация	Стандартное действие драйвера				
	ввод символа	вывод символа			
^H или \b	—	backspace	Ctrl H	0x08	BS
^W	werase	—	Ctrl W	0x17	ETB
^U	kill	—	Ctrl U	0x15	NAK
^I или \t	—	tab	⇆ Tab или Ctrl I	0x09	HT
^M или \r	eol	cr	↵ Enter или Ctrl M	0x0D	CR
^J или \n	eol	nl	Ctrl J	0x0A	LF
^S	stop	—	Ctrl S	0x13	DC3
^Q	start	—	Ctrl Q	0x11	DC1
^R	rprnt	—	Ctrl R	0x12	DC2
^V	lnext	—	Ctrl V	0x16	SYN
^N	—	so	Ctrl N	0x0E	SO
^O	—	si	Ctrl O	0x0F	SI
^[или \e	esc	esc	Esc или Ctrl [или Ctrl 3	0x1B	ESC

Так, например, нажатие клавиши ↵ Enter или эквивалентное сочетание Ctrl J, записывающееся как **^J**, генерирует управляющий символ **LF** (таким же действием обладает символ **CR**, **^M**), который сигнализирует драйверу терминала о завершении ввода строки (eol, **e**nd **o**f **l**ine) и необходимости «отдать команду на выполнение» (листинг 2.2).

Листинг 2.2. Управляющие символы ^J и ^M

```
finn@ubuntu:~$ date ↵
Вс. февр.  1 22:39:00 MSK 2015
finn@ubuntu:~$ hostname ^M
ubuntu
finn@ubuntu:~$ whoami ^J
finn
```