Краткое содержание

Оглавление

Если вы интересуетесь системой Linux, книга «Внутреннее устройство Linux» подойдет вам лучше всего.

LinuxInsider

В книге множество информации практически по всем аспектам архитектуры Linux.

Everyday Linux User

Вы получите полноценное представление о том, что происходит внутри системы, не увязая во множестве деталей. Это отличное дополнение к литературе по Linux.

Фил Булл, соавтор книги Ubuntu Made Easy и член команды по составлению документации для Ubuntu

Автор погружается в глубину операционных систем на базе Linux и рассказывает, как все части системы сочетаются друг с другом.

DistroWatch

Книга заслуживает места на полке суперпользователя Linux.

Журнал The MagPi

Об авторе

Брайан Уорд работает с операционной системой Linux с 1993 года. Кроме этой, он написал книги *The Linux Kernel HOWTO*, *The Book of VMware* и *The Linux Problem Solver*.

О научных редакторах

Жорди Гутьеррес Эрмосо — профессиональный разработчик и пользователь GNU/Linux с почти двадцатилетним опытом работы, внесший вклад в развитие GNU Octave и Mercurial. В разное время он работал с криптографией, медицинской визуализацией и в сфере экологии — везде на Linux. Когда Жорди не сидит за компьютером, то занимается плаванием, математикой и вязанием.

Петрос Кутупис в настоящее время старший инженер по производительности программного обеспечения в компании HPE (ранее Cray Inc.) в подразделении Lustre High Performance File System. Он создатель RapidDisk Project (www.rapiddisk.org) и занимается его сопровождением. Петрос более десяти лет работает в индустрии хранения данных и помог внедрить многие современные технологии.

Благодарности

Вклад в эту книгу внесли не только те, кто участвовал в процессе ее создания, но и те, без кого я бы ничего не узнал о Linux, а именно: Джеймс Дункан, Дуглас Н. Арнольд, Билл Феннер, Кен Хорнстайн, Скотт Диксон, Дэн Эрлих, Феликс Ли и Грегори П. Смит. За помощь в работе над предыдущими изданиями благодарю Кароля Хурадо, Лорел Чун, Серену Янг, Элисон Лоу, Райли Хоффмана, Скотта Шварца, Дэна Салли, Доминика Пулена, Дональда Кейрона и Джину Стил.

В подготовке третьего издания участвовали Барбара Куин, Рэйчел Монаган, Джилл Франклин, Ларри Уэйда, Хорди Гутьерреса Эрмосо и Петрос Кутуписа. Билл Поллок, основатель издательства No Starch Press, сыграл особую роль в создании этой книги с ее первого издания. И еще раз благодарю Синьцзю Сие за то, что вытерпел меня и в этот раз.

Предисловие

Операционная система не должна быть загадкой. Любому специалисту хочется использовать свое программное обеспечение по желанию, без магических заклинаний или ритуалов. Чтобы этого добиться, необходимо понимать, что делает программное обеспечение и как оно работает, и именно этому посвящена данная книга. Больше вам никогда не придется сражаться со своим компьютером.

Linux — отличная платформа для обучения, так как она ничего не пытается скрыть от пользователя. Большинство сведений о конфигурации системы можно найти в легкочитаемых текстовых файлах. Единственная сложность — выяснить, какие части за что отвечают и как они все сочетаются друг с другом.

Для кого эта книга

Интерес к устройству Linux затрагивает разные сферы жизни. Профессионалы в сфере DevOps и разработчики должны знать почти всю информацию, которая рассматривается в этой книге. Архитекторы и разработчики программного обеспечения Linux также должны знать это, чтобы пользоваться операционной системой наилучшим образом. Для исследователей и студентов, часто работающих в своих собственных системах Linux, будет полезно узнать, почему в системе все устроено именно так, а не иначе, что и рассказывается в книге. Кроме того, есть еще и любители — люди, которые просто проводят время за своими компьютерами ради развлечения, выгоды или и того и другого сразу.

Хотите знать, почему одни вещи работают, а другие нет? Вам интересно, что произойдет, если что-либо изменить? Если вы ответили «Да!», то вы, скорее всего, любитель и найдете ответы на свои вопросы в этой книге.

Требуемый уровень знаний

Вам необязательно быть программистом, чтобы прочитать эту книгу. Понадобятся только базовые навыки пользователя компьютера: вы должны ориентироваться в графическом интерфейсе (особенно в интерфейсе установки и настроек для дистрибутива Linux) и знать, что такое файлы и каталоги (папки). Кроме того, будьте готовы искать и проверять дополнительную документацию в вашей системе и в интернете. И самое главное — быть готовыми исследовать свой компьютер.

Как читать книгу

Когда речь идет о технических предметах, донести все необходимые знания, — непростая задача. С одной стороны, читатель увязает в излишних подробностях и с трудом усваивает суть, поскольку человеческий разум просто не может одновременно обработать большое количество новых понятий. С другой — отсутствие подробностей приводит к тому, что читатель получает лишь смутное представление о предмете и не готов к усвоению дальнейшего материала.

В этой книге я упростил изложение и структурировал материал. В большинстве глав важная информация, которая необходима для дальнейшей работы, предлагается в первую очередь. По мере чтения главы вы встретите в ней и дополнительный материал. Надо ли вам сразу усваивать эти частности? В большинстве случаев полагаю, что нет. Если ваши глаза начинают тускнеть при виде большого количества подробностей, относящихся к только что изученному материалу, не раздумывая переходите к следующей главе или сделайте перерыв. Вас ожидают другие важные вещи.

Практический подход

Для работы вам понадобится компьютер с операционной системой Linux. Возможно, вы предпочтете виртуальную установку — для проверки большей части материала данной книги я пользовался приложением VirtualBox. Вы должны обладать правами доступа superuser (root), хотя основную часть времени следует использовать учетную запись обычного пользователя. Вы будете работать главным образом в командной строке, окне терминала или в удаленном сеансе. Если вы нечасто работали в этой среде, ничего страшного, в главе 2 вы узнаете об этом подробнее.

Как правило, команды будут выглядеть следующим образом:

```
$ ls /
[some output]
```

Вводите текст, который выделен жирным шрифтом; обычным шрифтом показан ответный текст, который выдаст машина. Символ `$` является приглашением для пользователя с обычной учетной записью. Если вы увидите в качестве приглашения символ `#`, следует работать в учетной записи superuser (подробнее об этом в главе 2).

Как устроена эта книга

Я сгруппировал главы книги в три основные части. Первая часть — вводная, дает представление о системе в целом, а затем предлагается ряд практических заданий с инструментами, которые понадобятся вам для дальнейшей работы в системе. Далее вы детально изучите каждую часть системы, начиная с управления оборудованием и заканчивая конфигурацией сети, следуя обычному порядку запуска системы.

И наконец, вы познакомитесь с частями работающей системы, освоите необходимые навыки и рассмотрите инструменты, которые используют программисты.

В большинстве первых глав (кроме главы 2) активно задействовано ядро системы Linux, но по мере продвижения по книге вы будете работать и в своем пространстве пользователя. Если вы не понимаете, о чем я сейчас говорю, не беспокойтесь, объяснения будут даны в главе 1.

Материал излагается по возможности без привязки к какому-либо дистрибутиву системы. Было бы скучно описывать все варианты системы, поэтому я попытался рассказать о двух основных семействах дистрибутивов: Debian (включая Ubuntu) и RHEL/Fedora/CentOS. Кроме того, я описал настольные и серверные установки. Значительный объем материала можно использовать во встроенных системах, таких как Android и OpenWRT, но поиск различий между ними предоставляется вам.

Новое в третьем издании

Второе издание вышло в переходный период для всех систем Linux. Часть традиционных компонентов постепенно изменялась, что затрудняло изучение, потому что читатели сталкивались с большим разнообразием настроек. Однако сейчас новые элементы (в частности, `systemd`) надежно заняли свои места в системах, поэтому я смог значительно упростить и сократить материал по этой теме.

Я все так же сохранил акцент на роли ядра в системе Linux. Именно эта информация была наиболее интересна читателям, и вы сами, скорее всего, взаимодействуете с ядром чаще, чем думаете.

В новом издании появилась глава о виртуализации. Несмотря на то что Linux часто устанавливают именно через виртуальные машины (например, через облачные серверы), подобный способ виртуализации выходит за рамки данной книги. Все потому, что система работает на виртуальной машине практически так же, как на «голом» железе. Таким образом, информация об этом отличается в основном лишь терминологией. Кроме того, со времени выхода второго издания контейнеризация стала более популярной, поэтому я добавил информацию и о ней, ведь контейнеры в основном состоят из множества функций Linux, которые описаны в книге. В контейнерах часто используются контрольные группы `cgroups`, о которых я также рассказываю в третьем издании. В книгу в том числе добавлены сведения о менеджере логических томов, системе ведения журнала `journald` и протоколе Ipv6 (в главе 9).

Мне хотелось снабдить вас сведениями, которые понадобятся для быстрого начала работы. Их усвоение потребует некоторых усилий, однако я не намереваюсь делать из вас «тяжелоатлетов», чтобы вы смогли одолеть эту книгу. Когда вы будете понимать важнейшие моменты, изложенные здесь, для вас не составит труда отыскать подробности и разобраться в них.

Я удалил кое-какие исторические детали, которые были в первом издании. Если вы интересуетесь системой Linux и ее отношением к истории системы Unix, обратитесь к книге Питера Салуса *The Daemon, the Gnu, and the Penguin* (Reed Media Services, 2008) — в ней рассказано о том, как развивалось используемое нами программное обеспечение.

Немного о терминологии

До сих пор ведутся споры о наименованиях некоторых элементов операционных систем. Они затрагивают даже само название системы Linux — как она должна называться: Linux или же GNU/Linux (чтобы отразить применение некоторых элементов проекта GNU)? В книге я старался использовать самые распространенные и по возможности наименее громоздкие названия.

От издательства

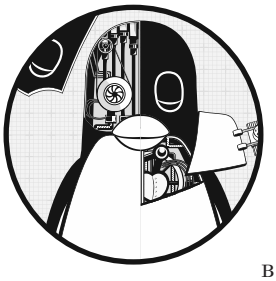
Ваши замечания, предложения, вопросы отправляйте по адресу comp@piter.com (издательство «Питер», компьютерная редакция).

Мы будем рады узнать ваше мнение!

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

1

Общая картина



На первый взгляд такая современная операционная система, как Linux, очень сложна и состоит из невероятного количества инструментов, которые работают и взаимодействуют друг с другом одновременно. Например, веб-сервер может взаимодействовать с сервером базы данных, который, в свою очередь, может применять разделяемую библиотеку, используемую многими другими программами. Как же все это работает и какой во всем этом смысл?

Самый эффективный способ понять, как работает операционная система, — абстрагироваться от большинства деталей, составляющих часть, которую вы изучаете, и сосредоточиться на ее основной цели и функциональности. Например, когда вы едете в автомобиле, вам не нужно думать о таких деталях, как крепежные болты, которые удерживают двигатель внутри него, или о людях, которые строят и ремонтируют дорогу, по которой он едет. Все, что вам действительно нужно знать, — это цель автомобиля (перевезти вас куда-то) и основные знания о том, как им пользоваться (например, как открывать дверь и пристегивать ремень безопасности).

Подобный уровень абстрагирования сработает, если вы просто пассажир в машине. Но если вы управляете автомобилем, вам нужно копнуть глубже и разбить свою абстракцию на несколько частей. Вы должны расширить свои знания в трех областях: сам автомобиль (например, его размер и возможности), манипуляции его частями (рулевое колесо, педаль акселератора и т. д.) и особенности дороги.

Абстрагирование от деталей может помочь при попытке найти и устранить проблемы. Предположим, что вы ведете машину и поездка дается тяжело. Вы можете

быстро оценить три основные упомянутые абстракции, связанные с автомобилем, чтобы определить источник проблемы. Если с первыми двумя абстракциями все в порядке (ваш автомобиль и способ вождения) и ни одна из них не является проблемой, можете сузить проблему до самой дороги. В таком случае выясняется, что дорога ухабистая. Теперь, если нужно, вы можете копнуть глубже в абстракцию дороги и подумать, почему ее состояние ухудшилось или, если она новая, почему строители так паршиво сделали свою работу.

Разработчики программного обеспечения используют абстракцию в качестве инструмента при создании операционной системы и ее приложений. В программном обеспечении существует множество терминов для абстрактного разделения, включая *«подсистему»*, *«модуль»* и *«пакет»*, но в этой главе мы будем применять термин *«компонент»*, потому что так проще. При создании программного компонента разработчики обычно не задумываются о внутренней структуре других компонентов, но рассматривают те из них, которые могут задействовать (чтобы не приходилось писать дополнительное ненужное программное обеспечение), и размышляют о том, как их использовать.

В этой главе мы подробно рассмотрим компоненты, составляющие систему Linux. Хотя каждый из них имеет огромное количество технических деталей, сейчас не будем брать их в расчет и сосредоточимся на том, что именно компоненты делают по отношению ко всей системе. А детали рассмотрим в последующих главах.

1.1. Уровни абстракции в системе Linux

Использование абстракции для разделения вычислительных систем на компоненты облегчает понимание, но не работает без организации процессов. Мы объединяем компоненты в слои или уровни классификации (или группы) в соответствии с тем, где они находятся между пользователем и оборудованием. Веб-браузеры, игры и т. п. находятся на верхнем уровне, на нижнем располагается память в компьютерном оборудовании — нули и единицы. Операционная система занимает множество промежуточных уровней.

Система Linux имеет три основных уровня. На рис. 1.1 показаны эти уровни и компоненты внутри каждого из них. Аппаратное оборудование компьютера (hardware) находится на базовом уровне. Оно включает в себя память и один или несколько процессоров (CPU) для выполнения вычислений, а также для чтения из памяти и записи в нее. Такие устройства, как диски и сетевые интерфейсы, — тоже часть аппаратного оборудования.

Следующий уровень — это *ядро (kernel)*, основа операционной системы. Ядро — это программное обеспечение, содержащееся в памяти. Оно сообщает процессору, где находится его следующая задача. Выступая в качестве посредника, ядро управляет оборудованием (особенно оперативной памятью) и является основным интерфейсом между оборудованием и любой запущенной программой.

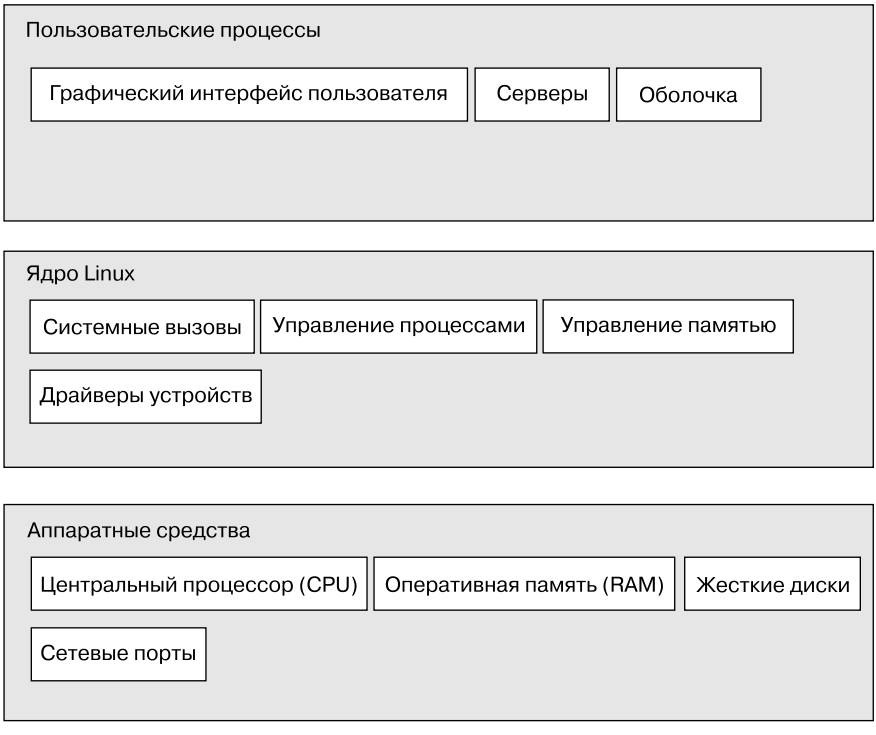


Рис. 1.1. Основные компоненты системы Linux

Процессы (processes) — запущенные программы, которыми управляет ядро, — в совокупности составляют верхний уровень системы, называемый *пользовательским пространством (user space)*. (Более конкретный термин для процесса — *«пользовательский процесс» (user process)*, независимо от того, взаимодействует ли пользователь непосредственно с процессом. Например, все веб-серверы работают как пользовательские процессы.)

Существует огромная разница между тем, как работает ядро и пользовательские процессы: ядро работает в *режиме ядра (kernel mode)*, а пользовательские процессы — *в пользовательском режиме*. Код, работающий в режиме ядра, имеет неограниченный доступ к процессору и оперативной памяти. Такая мощная, но опасная привилегия позволяет легко повредить ядро и вывести из строя всю систему. Область памяти, доступ к которой может получить только ядро, называется *пространством ядра (kernel space)*.

Пользовательский же режим ограничен доступом к подмножеству памяти (обычно довольно небольшому) и безопасным операциям процессора. *Пользовательское пространство* — это части основной памяти, к которым могут получить доступ пользовательские процессы. Если процесс совершает ошибку и выходит из строя,

последствия локальны и устраняются с помощью ядра. Это означает, что если ваш веб-браузер выйдет из строя, он не будет продолжать научные вычисления, которые выполнялись в фоновом режиме в течение нескольких дней.

Теоретически, пользовательский процесс, вышедший из строя, не может нанести серьезного ущерба остальной части системы. На самом деле это зависит от того, что вы считаете серьезным ущербом, а также от особых привилегий процесса, ведь некоторым процессам разрешено делать больше, чем другим. Например, может ли пользовательский процесс полностью уничтожить данные на диске? Если имеет необходимые привилегии — да, и это довольно опасно. Однако существуют меры предосторожности, и большинству процессов просто не разрешается сеять хаос в системе.

ПРИМЕЧАНИЕ

Ядро Linux может запускать потоки ядра (kernel threads), очень похожие на процессы, но имеющие доступ к пространству ядра, например kthreadd и kblockd.

1.2. Оборудование: оперативная память

Оперативная память — это, пожалуй, самый важный элемент оборудования компьютерной системы. В своей первоначальной форме оперативная память — это просто огромная область хранения для группы нулей и единиц. Каждый слот для нуля или единицы называется битом. Именно в оперативной памяти хранятся запущенные ядро и процессы — по сути, тоже большие наборы битов. Все входные и выходные данные с периферийных устройств проходят через оперативную память также в виде набора битов. Процессор — это оператор памяти, он считывает свои инструкции и данные из памяти и записывает данные обратно в память.

В отношении памяти, процессов, ядра и других частей компьютерной системы часто используется термин *«состояние» (state)*. Строго говоря, состояние — это особое расположение битов. Например, если у вас есть четыре бита в памяти, то биты 0110, 0001 и 1011 представляют три разных состояния системы.

Если учесть, что один процесс может легко состоять из миллионов битов в памяти, то когда речь идет о состояниях, проще использовать абстрактные термины. Вместо того чтобы описывать состояние с помощью битов, процесс описывают как завершенный или происходящий в данный момент. Например, вы можете сказать: «Процесс ожидает ввода» или «Процесс находится на втором этапе запуска».

ПРИМЕЧАНИЕ

Поскольку принято ссылаться на состояние в абстрактных терминах, а не на фактические биты, термин «образ» (image) относится к определенному физическому расположению битов.

1.3. Ядро

Почему мы говорим об оперативной памяти и состояниях? Почти все, что делает ядро, связано с оперативной памятью. Одной из задач ядра является разделение памяти на множество областей, и оно должно постоянно отслеживать определенную информацию об их состоянии. Каждый процесс получает некоторую часть памяти, и ядро должно обеспечивать необходимое количество памяти для каждого из процессов. Ядро отвечает за управление задачами в четырех основных областях системы.

- **Процессы.** Ядро определяет, каким процессам разрешено использовать процессор.
- **Память.** Ядро должно отслеживать распределение памяти: сколько в данный момент выделено конкретному процессу, сколько можно разделить между другими процессами и сколько свободно.
- **Драйверы устройств.** Ядро действует как интерфейс между оборудованием (например, диском) и процессами. Обычно оно управляет подключенным оборудованием.
- **Системные вызовы и поддержка.** Процессы обычно используют системные вызовы для связи с ядром.

Далее мы кратко изучим каждую из этих областей.

ПРИМЕЧАНИЕ

Если вы хотите изучить работу ядра более подробно, то я советую прочитать две отличные книги на эту тему: десятое издание книги Ави Зильбершаца Operating System Concepts (Wiley, 2018) и четвертое издание книги Эндрю Таненбаума Modern Operating Systems (Prentice Hall, 2014).

1.3.1. Управление процессами

Управление процессами описывает запуск, приостановку, возобновление, планирование и завершение процессов. Концепции, лежащие в основе запуска и завершения процессов, довольно просты, но описание того, как процессом используется процессор, немного сложнее.

В любой современной операционной системе многие процессы выполняются параллельно. Например, у вас могут быть открыты веб-браузер и электронная таблица одновременно. Однако все не так, как кажется: процессы, стоящие за этими приложениями, обычно не выполняются в одно и то же время.

Рассмотрим систему с одноядерным процессором. Многие процессы могут задействовать процессор, но только один из них фактически использует процессор

в любой момент времени. На практике процесс работает с процессором в течение небольшой доли секунды и делает паузу, затем другой процесс занимает процессор в течение еще одной небольшой доли секунды, потом в дело вступает еще один процесс и т. д. Акт передачи одним процессом управления процессором другому процессу называется *переключением контекста*.

Каждый отрезок времени, называемый *квантом времени*, позволяет процессу выполнить значительные вычисления (и действительно, процесс часто завершает свою задачу в течение одного кванта времени). Однако из-за того, что кванты очень малы, пользователи их не воспринимают, и возникает ощущение, что система выполняет несколько процессов одновременно (режим многозадачности).

Ядро отвечает за переключение контекста. Чтобы понять, как это работает, рассмотрим ситуацию, в которой процесс работает в пользовательском режиме, но его временной квант истек. Вот что происходит.

1. Процессор (фактическое оборудование) прерывает текущий процесс на основе внутреннего таймера, переключается в режим ядра и передает управление обратно ядру.
2. Ядро записывает текущее состояние процессора и памяти, что необходимо для возобновления только что прерванного процесса.
3. Ядро выполняет любые задачи, которые могли возникнуть в течение предыдущего временного кванта (например, сбор данных из ввода-вывода).
4. Теперь ядро готово к запуску другого процесса. Оно анализирует список процессов, готовых к запуску, и выбирает один из них.
5. Ядро подготавливает память для нового процесса, а затем готовит к нему процессор.
6. Ядро сообщает процессору длительность временного кванта для нового процесса.
7. Ядро переключает процессор в пользовательский режим и передает управление процессором процессу.

Переключение контекста позволяет понять, когда именно запускается ядро. Суть заключается в том, что ядро запускается между временными квантами процесса во время переключения контекста.

В случае многопроцессорной системы, как и в большинстве современных машин, все немного сложнее, потому что ядро не перестает управлять текущим процессором, чтобы позволить процессу работать на другом процессоре, и одновременно могут выполняться несколько процессов. Но чтобы максимально задействовать все доступные процессоры, ядро в любом случае выполняет необходимые шаги (и может использовать определенные лазейки, чтобы занять больше времени процессора для себя).

1.3.2. Управление памятью

Ядро должно управлять памятью во время переключения контекста, а это довольно сложная задача. Должны выполняться следующие условия.

- Ядро должно иметь в памяти выделенную область, к которой пользовательские процессы не могут получить доступ.
- Каждому пользовательскому процессу необходима собственная область памяти.
- Один пользовательский процесс не может получить доступ к области памяти, выделенной другому процессу.
- Пользовательские процессы могут совместно работать с памятью.
- Часть памяти пользовательских процессов может быть доступна только для чтения.
- Система может использовать больше памяти, чем ее существует физически, задействуя дисковое пространство в качестве вспомогательного механизма.

К счастью для ядра, оно не одно выполняет всю работу. Современные процессоры включают в себя блок управления памятью (memory management unit, MMU), который обеспечивает схему доступа к памяти, называемую *виртуальной памятью*. При использовании виртуальной памяти процесс не получает прямого доступа к памяти через ее физическое местоположение в компьютерной системе. Вместо этого ядро настраивает каждый процесс, чтобы он действовал так, будто ему доступна вся система. Когда процесс обращается к части своей памяти, MMU перехватывает обращение и с помощью таблицы соответствий преобразует адрес памяти с точки зрения процесса в фактическое физическое местоположение памяти в системе. Ядро по-прежнему должно инициализировать, постоянно поддерживать и изменять таблицу соответствий адресов памяти. Например, во время переключения контекста ядро должно заменить таблицу соответствий исходного процесса на таблицу последующего процесса.

ПРИМЕЧАНИЕ

Реализация таблицы соответствий адресов памяти называется таблицей страниц.

Подробнее о том, как отслеживать производительность памяти, описано в главе 8.

1.3.3. Управления драйверами устройств

Роль ядра в работе с устройствами относительно проста. Устройство обычно доступно только в режиме ядра, поскольку неправильный доступ (например, пользовательский процесс, запрашивающий отключение питания) может привести к сбою системы. Значительная проблема заключается в том, что различные устройства редко имеют один и тот же интерфейс программирования, даже если устройства