ЧАСТЬ I

Низкоуровневая конфигурация системы

Глава 1

Настройка сетевых средств ядра

"Все дороги ведут в Рим" — гласит пословица. Нечто подобное можно сказать и о сетевых средствах Linux; в этом случае в роли Рима выступает ядро операционной системы. Рано или поздно весь сетевой трафик будет обработан ядром. Различные компьютеры и разные сети отличаются друг от друга, поэтому для ядра Linux предусмотрен ряд опций, изменяя значения которых вы можете оптимизировать систему с учетом задач, которые ей придется выполнять. Установку некоторых опций можно производить в процессе загрузки системы, передавая ядру необходимые параметры, либо впоследствии, после окончания загрузки. Подобные опции будут рассмотрены в последующих главах. Существуют также характеристики, для изменения которых надо перекомпилировать ядро системы.

В данной главе описаны опции, которые задают конфигурацию ядра. Сначала здесь рассматривается процедура настройки ядра. Затем обсуждаются опции, которые определяют свойства TCP/IP и других сетевых протоколов, а также сетевых фильтров. Далее речь пойдет о драйверах Linux, предназначенных для поддержки различных сетевых устройств. В конце главы кратко описывается процесс компиляции ядра системы.

В этой главе компиляция ядра рассматривается лишь в общих чертах, основное внимание уделяется опциям, определяющим характеристики сетевых средств системы. Если вы хотите получить более подробную информацию о конфигурации ядра системы, обратитесь к документу *Linux Kernel HOWTO* (`http://www.linuxdoc.org/HOWTO/Kernel-HOWTO.html`) либо к соответствующим книгам, представляющим собой введение в операционную систему Linux.

Конфигурация ядра

Для того чтобы установить опции, определяющие процесс компиляции ядра, необходимо иметь в наличии исходный код ядра. Исходный код входит в состав всех дистрибутивных пакетов, но при установке системы можно либо разрешить, либо запретить копирование исходного кода на жесткий диск компьютера. Следует заметить, что в некоторых случаях исходный код, поставляемый в составе дистрибутивного пакета, может быть изменен по сравнению со стандартным кодом ядра (так, например, в состав кода могут быть включены специальные драйверы). Целесообразно вначале инсталлировать стандартное ядро, а затем, по мере необходимости, установить дополнительные модули (не исключено, что для выполнения ваших задач никакие дополнения не потребуются). Спи-

сок основных узлов, содержащих архивы Linux, находится по адресу `http://www.kernel.org`. В частности, там вы найдете ссылку на `ftp://sunsite.unc.edu` и адреса других узлов, содержащих последние варианты исходного кода ядра Linux. (Конечно, вы можете работать с исходным кодом ядра, который входит в состав дистрибутивного пакета, но, как было сказано выше, в нем могут быть установлены дополнительные модули. Если в процессе работы возникнут проблемы, то устранить их будет легче, если у вас инсталлировано стандартное ядро.)

Номер версии ядра системы состоит из трех чисел, разделенных точками. Если второе число четное (например, 2.4.17), то ядро называется *стабильным*, или *рабочим*. Нечетное второе число в номере версии (например, 2.5.2) указывает на то, что ядро находится *в процессе разработки*. Стабильное ядро обеспечивает более высокую надежность. Используя ядро, находящееся в процессе разработки, вы получаете возможность ознакомиться с новыми техническими решениями. Чаще всего в ядре с нечетным вторым числом номера версии используются новые драйверы, реализованы новые варианты интерфейса или применяются другие подобные новшества. Устанавливая систему для практического использования, желательно использовать ядро с четным вторым числом номера версии. Исключением является ситуация, когда необходимый вам драйвер присутствует только в версии с нечетным вторым числом. В этом случае можно также использовать *обратный перенос* (back-port) драйвера в одну из предыдущих стабильных версий.

Обычно исходный код ядра содержится в каталоге `/usr/src/linux` либо в одном из подкаталогов `/usr/src` (при этом в имени каталога присутствует номер версии ядра, например `/usr/src/linux-2.4.17`). В последнем случае желательно создать ссылку `/usr/src/linux`, указывающую на каталог с исходным кодом ядра. Если вы поступите так, то обеспечите нормальную работу программ, которые предполагают, что исходный код ядра содержится в каталоге `/usr/src/linux`. Таким образом, удобно работать с несколькими версиями исходного кода ядра, а если надо перейти от одной версии к другой, достаточно лишь изменить символьную ссылку.

Разархивировав исходный код ядра в каталог `/usr/src/linux`, надо сделать это каталог рабочим в используемой вами оболочке. После этого можно задать одну из описанных ниже команд конфигурирования ядра.

- `make config`. Данное средство конфигурирования является базовым. При этом у вас поочередно будут запрашиваться значения опций ядра. Отвечать на вопросы утомительно и при этом легко допустить ошибку. В случае ошибки придется начать всю процедуру сначала. Данная команда в настоящее время используется крайне редко.
- `make menuconfig`. Это средство конфигурирования предоставляет меню, позволяющее просматривать опции и задавать новые значения. Меню отображается в алфавитно-цифровом режиме. В этом случае изменить придется только те опции, значения которых не устраивают вас. При работе в текстовом режиме данное средство применяется чаще других.
- `make xconfig`. Данный способ установки конфигурации аналогичен `make menuconfig`, за исключением того, что меню отображается средствами графического интерфейса. В этом случае выбор опций и установку их значений можно выполнять с помощью мыши. Это средство установки конфигурации применяется при работе в среде X Window (X Window иногда называют X).

Все три способа позволяют работать с одними и теми же опциями. Опции объединены в несколько категорий; некоторые из категорий содержат подкатегории. Если вы используете `make menuconfig` или `make xconfig`, то для каждой категории отображается отдельное меню (пример работы с окном, отображаемым по команде `make xconfig`, показан на рис. 1.1). При настройке сетевых средств в основном используются категории Networking Options и Network Device Support, которые подробно рассматриваются в двух последующих разделах.

Для большинства опций предусмотрены переключатели. Примерами таких переключателей могут служить Y, M и N, показанные на рис. 1.1. Y и N указывают на присутствие или отсутствие

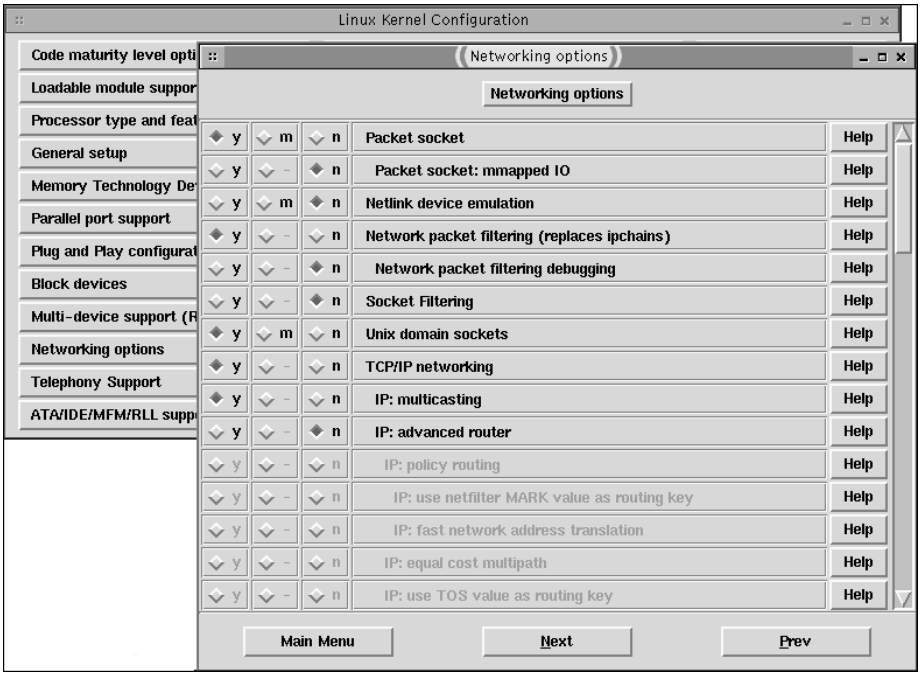

Рис. 1.1. Опции ядра Linux объединяются в категории и подкатегории, для каждой из которых предусмотрено отдельное меню

опции в составе ядра, а M (сокращение от modular compilation — модульная компиляция) указывает на то, что соответствующие средства должны быть скомпилированы как отдельный модуль, которые можно загружать и выгружать независимо от других компонентов ядра. Боле подробно о настройке опций рассказывается ниже.

Данная глава посвящена опциям версии 2.4.x ядра Linux, в частности, материал главы ориентирован на ядро 2.4.17. Опции, относящиеся к сетевым средствам, модифицировались раньше и, по-видимому, будут изменяться и в будущем. В версиях 2.2.x ядра опции в основном совпадают; различаются они лишь в деталях. В состав разрабатываемого ядра 2.5.x включено инструментальное средство CML2, предназначенное для настройки. Дополнительную информацию об этом инструменте можно получить по адресу `http://tuxedo.org/~esr/cml2/`.

Поддержка сетевых протоколов

Меню Networking Options содержит опции, влияющие на работу сетевых протоколов. Вы можете включить или исключить средства поддержки стека протоколов либо отдельных протоколов (в основном данные опции касаются семейства протоколов TCP/IP). Опции из этого меню позволяют также оптимизировать ядро для выполнения конкретных функций, например маршрутизации или фильтрации пакетов.

Опции для работы с пакетами и гнездами

Низкоуровневые сетевые средства Linux позволяют программам передавать и принимать фрагменты данных, называемые *пакетами*, посредством специальных структур, которые называются *гнездами* (socket). В большинстве случаев обмен данными через гнездо осуществляется по тому же принципу, что и обмен данными с файлом. Стек сетевых протоколов обеспечивает передачу информации по адресу назначения, где происходит ее интерпретация.

В некоторых случаях желательно и даже необходимо изменить принцип обработки данных; иногда приходится расширять стандартный набор операций над пакетами. Сделать это позволяют специальные опции, рассмотрению наиболее важных из них посвящены разделы данной главы. Некоторые из опций кратко описаны ниже.

- Packet Socket. Эта опция позволяет приложениям непосредственно обращаться к требуемому протоколу, минуя некоторые уровни стека протоколов. Для большинства программ такая возможность не нужна; ее используют лишь инструментальные средства сетевой диагностики и специальные утилиты, действующие на нижнем уровне. В качестве примера подобных программ можно привести утилиту `tcpdump`, которая выводит информацию о пакетах TCP и IP. Данная опция не обязательна. Она несколько увеличивает размер ядра и дает возможность злоумышленникам воспользоваться утилитами сетевой диагностики. С другой стороны, отключив данную опцию, вы не сможете воспользоваться целым рядом утилит.
- Packet Socket: Mapped IO. Если данная подопция Packet Socket включена, производительность инструментальных средств, использующих низкоуровневые соединения, повышается.
- Unix Domain Sockets. Некоторые важные программы Linux используют сетевые протоколы для обмена данными даже в том случае, если они выполняются на одном и том же компьютере. В качестве примеров можно привести средство протоколирования `syslogd` и программы, выполняющиеся в среде X Window (X-программы используют сетевой протокол для взаимодействия с X-сервером, выполняющим отображение данных). Опция Unix Domain Sockets допускает взаимодействие в пределах одной системы даже в тех случаях, когда на компьютере не установлено сетевое оборудование. Даже если средства поддержки сетевого обмена присутствуют, опция Unix Domain Sockets обеспечивает более высокую скорость обмена по сравнению с обычными TCP-гнездами. Обычно данная опция устанавливается; без нее обходятся лишь системы, предназначенные для выполнения на специализированных устройствах.

По умолчанию все три указанные опции устанавливаются. При необходимости вы можете запретить Packet Socket.

Опции сетевой фильтрации

Опции сетевой фильтрации блокируют или преобразуют пакеты, поступающие на компьютер или покидающие его. Данные опции используются при создании брандмауэров и выполнении IP-маскировки (подробно эти вопросы будут обсуждаться в главе 25). Брандмауэры блокируют нежелательные обращения к компьютеру или сети, а IP-маскировка позволяет организовать работу в Internet пользователей всей локальной сети при наличии одного IP-адреса. Опции ядра системы, предназначенные для фильтрации, перечислены ниже.

- Socket Filtering. В обычных условиях ядро направляет все пакеты, полученные через некоторое гнездо, программе, которая создала это гнездо. Опция Socket Filtering позволяет указать ядру на то, что принятые пакеты должны быть сначала переданы небольшой программе (которая называется фильтром). Эта программа способна блокировать некоторые из пакетов. Как правило, программы могут работать без данной опции. Исключение составляют последние варианты серверов DHCP и клиентов DHCP. Если в вашей сети используются средства DHCP (Dynamic Host Configuration Protocol — протокол динамической конфигурации узла), данная опция должна быть установлена.

- Network Packet Filtering. Данная опция является наиболее важным средством фильтрации, так как именно она делает возможной работу брандмауэра и IP-маскировку. Обычно опция Network Packet Filtering устанавливается; при этом становится доступной опция Network Packet Filtering Debugging, которую можно использовать для решения возникающих проблем. Кроме того, становится также доступным подменю IP: Netfilter Configuration. В этом подменю отображаются описанные ниже опции.
- Connection Tracking. Эта опция обеспечивает более высокую степень контроля над сетевыми соединениями, чем это возможно в обычных условиях. Как правило, маршрутизаторы ограничиваются пересылкой информационных пакетов между сетевыми интерфейсами. Если опция Connection Tracking активна, система запоминает IP-адрес источника, IP-адрес назначения и порты для дальнейшего использования. Эта возможность необходима для реализации IP-маскировки. В других случаях опцию Connection Tracking можно отключить. Если данная опция установлена, доступны опции поддержки FTP, что позволяет обеспечить работу данного протокола при наличии IP-маскировки.
- IP Tables Support. Данная опция включает поддержку ядром утилиты `iptables`, используемой для реализации брундмауэтов и осуществления IP-маскировки (эти вопросы будут подробно обсуждаться в главе 25). При установленной опции IP Tables Support становятся доступны подопции, позволяющие настроить средства поддержки `iptables` для выполнения конкретных задач. Многие из этих подопций задают соответствие ядра определенному типу, и их имена имеют вид *Тип* Match Support. Из них очень важна опция Connection State Match Support, которая позволяет осуществлять *проверку пакетов с учетом состояния* (stateful packet inspection). Эта операция применяется в брундмауэрах и подробно рассматривается в главе 25. Также важны опции Packet Filtering, Full NAT и LOG Target Support и их подопции. Установив данные опции, вы можете использовать ваш компьютер как брандмауэр или осуществлять IP-маскировку. Для независимой рабочей станции или сервера опцию Full NAT можно не указывать.
- ipchains (2.2-Style) Support. В некоторых случаях бывает необходимо обеспечить работу сценариев брандмауэра, ориентированных на использование утилиты `ipchains` (эта утилита применялась при работе с версиями ядра 2.2.x). Поддержку `ipchains` можно включить в том случае, если средства IP Tables Support не были скомпилированы непосредственно в ядро системы. (Средства `iptables` и `ipchains` выполняют приблизительно одинаковые действия, но они не совместимы друг с другом.) Если вы создаете брандмауэр с нуля, можете смело отключить поддержку `ipchains`.
- ipfwadm (2.0-Style) Support. При работе с версиями 2.0.x ядра для создания брандмауэров использовалось инструментальное средство `ipfwadm`. Чтобы использовать сценарии брандмауэра, ориентированные на `ipfwadm`, надо установить данную опцию. Следует помнить, что средства поддержки `ipfwadm` не совместимы ни с `iptables`, ни с `ipchains`. Если вы не используете `ipfwadm`-сценарии либо твердо решили преобразовать их для работы с `iptables`, можете отказаться от установки данной опции.

По мере перехода от версий 2.0.x к версиям 2.4.x ядра Linux средства поддержки фильтрации пакетов становились все сложнее. В ядре 2.4.x предусмотрены многие дополнительные возможности; создавая брандмауэр, важно активизировать те опции, которые необходимы для решения конкретной задачи. Если вы сомневаетесь, нужна ли та или иная опция из меню IP: Netfilter Configuration, рекомендую вам установить ее. В этом случае объем ядра несколько возрастет, но вы получите возможность использовать различные правила брандмауэра.

Вам может показаться, что использовать правила брандмауэра на машине под управлением Linux не обязательно, особенно если она находится в сети, которая защищена выделенным брандмауэром. К сожалению, в системе защиты многих сетей есть недостатки, поэтому дополнительные меры предосторожности не помешают. Возможно, вам потребуется установить на своем компьютере дополнительный простой брандмауэр.

Опции маршрутизации TCP/IP

Маршрутизатор — это компьютер, который непосредственно передает данные из одной сети в другую. Маршрутизаторы также часто называют *шлюзами*. Так, например, маршрутизатор может понадобиться для связи сети, принадлежащей отделу большой корпорации, с корпоративной сетью. Корпорация, в свою очередь, использует маршрутизатор для обеспечения связи своей сети с Internet. Рассмотрению опций маршрутизации посвящена глава 24. Сейчас вам достаточно знать лишь то, что для ядра Linux предусмотрен ряд опций, являющихся подопциями IP: Advanced Router.

Опции поддержки IPv6

Работа Internet обеспечивается за счет протоколов семейства TCP/IP, в частности, для передачи пакетов используется протокол IP (IPv4). К сожалению, на сегодняшний день уже невозможно игнорировать тот факт, что версия IPv4 устарела. Для представления IP-адреса в IPv4 используется 32-разрядное число, т. е. общее число адресов равно 2^{32}, или 4294967296. Вследствие неэффективности механизма распределения адресов реальное их количество оказывается намного меньшим. В результате возникла проблема нехватки IP-адресов. Кроме того, недостатки в защите IPv4 позволяют злоумышленникам вмешиваться в сеансы сетевого взаимодействия. На момент написания данной книги, т. е. в 2002 г., с проблемами, связанными с использованием IPv4, еще можно мириться, но их придется решить в течение ближайшего десятилетия.

В настоящее время разрабатывается версия IPv6, призванная заменить IPv4. В IPv6 поддерживаются 128-разрядные IP-адреса. Общее число IP-адресов равно 2^{128}, или $3{,}4\times10^{38}$ — приблизительно $2{,}2\times10^{18}$ адресов на квадратный миллиметр поверхности Земли. IPv6 также обеспечивает дополнительные средства защиты. В настоящее время число сетей, в которых используется IPv6, очень мало. Если ваш компьютер подключен к такой сети или если вы собираетесь в качестве эксперимента организовать обмен данными во внутренней сети предприятия посредством IPv6, вам надо активизировать средства поддержки IPv6, установив для этого опцию IPv6 Protocol (Experimental) в меню Networking Options. После установки данной опции вам станут доступны дополнительные опции, объединенные в подменю IPv6: Netfiler Configuration. В этом подменю также находятся описанные ранее опции фильтрации, но они ориентированы на работу с протоколом IPv6.

Чтобы активизировать средства поддержки IPv6, надо установить значения Yes опции Prompt for Development или Incomplete Code/Drivers в меню Code Maturity Level Options. То же самое надо сделать при работе с любыми "экспериментальными" драйверами. Со временем эксперименты с IPv6 закончатся, и опция, включающая поддержку IPv6, будет относиться к числу основных опций. Пока это не произошло, при работе с IPv6, как и при использовании других "экспериментальных" средств, следует соблюдать осторожность.

Опции QoS

Предположим, что компьютер под управлением Linux действует как маршрутизатор в сети с напряженным трафиком или выполняет роль сервера и обрабатывает при этом большой объем данных. При этом может возникнуть ситуация, когда система будет в течение некоторого времени получать большее число пакетов, чем она может обработать. Очевидно, что в этом случае необходимы специальные средства планировки, которые устанавливали бы очередность передачи пакетов.

Как правило, в системе Linux используется стратегия FIFO (first in/first out — "первый пришел — первый вышел"), согласно которой пакет, предназначенный для передачи, находится в очереди до тех пор, пока не будут переданы все пакеты, поставленные в очередь раньше него. Но в некоторых случаях необходимо предоставить пакетам определенного типа некоторые преимущества. Это могут быть пакеты, адресованные в конкретную сеть, или пакеты, которые содержат информацию, соответствующую определенному протоколу. Так, например, пакеты, содержащие информацию реального времени, например данные Internet-телефонии, целесообразно передавать вне очереди. Назначать приоритеты пакетам позволяют опции QoS (quality of service — качество сервиса). Эти опции доступны посредством подменю QoS and/or Fair Queueing меню Networking Options.

Для того чтобы реализовать систему QoS, необходимо выбрать опцию QoS and/or Fair Queueing в одноименном меню. В результате автоматически устанавливается ряд опций этого меню. Другие опции задаются отдельно. Основными из них являются опции планирования передачи пакетов и организации очереди, такие как CBQ Packet Scheduler и SFQ Queue. Эти опции позволяют ядру выполнять более сложную обработку пакетов по сравнению с традиционно используемым принципом FIFO. Опции QoS Support и Packet Classifier API, а также их подопции позволяют использовать Differentiated Services и Resource Reservation Protocol. При этом появляется возможность обмена QoS-приоритетами с другими маршрутизаторами. Если все маршрутизаторы на пути от одного узла к другому поддерживают совместимые между собой протоколы QoS, скорость передачи важных данных может быть увеличена за счет задержки информации, время доставки которой некритично.

Если система не выполняет функции маршрутизатора, опции QoS в ней, как правило, не используются. Если же вы создаете маршрутизатор, а в особенности, если он планируется для использования в сети с интенсивным обменом данными, желательно установить эти опции. Активизировав одну опцию, целесообразно активизировать и все остальные, в противном случае система не будет обладать должной гибкостью. Так, например, если вы не установите опцию U32 Classifier, то не сможете задавать приоритеты исходя из адресов назначения пакетов.

На практике использование средств QoS предполагает применение расширенных средств маршрутизации, таких как `ip` и `tc`. Об этих инструментах речь пойдет в главе 24, однако они слишком сложны, чтобы привести их исчерпывающее описание в рамках одной главы. Дополнительную информацию об `ip` и о `tc` можно найти в документах *iproute2 + tc Notes* (`http://snafu.freedom.org/linux2.2/iproute-notes.html`) и *Differentiated Services on Linux* (`http://diffserv.sourceforge.net`).

Поддержка протоколов высокого уровня

В ядре Linux предусмотрена поддержка нескольких протоколов высокого уровня. Благодаря этому коды, отвечающие за работу с этими протоколами, выполняются намного быстрее, чем соответствующие коды обычных пользовательских программ. Кроме того, поддержка высокоуровневых протоколов в ядре обеспечивает более тесную интеграцию этих протоколов с остальными компонентами операционной системы. Например, включение в состав ядра средств поддержки NFS позволяет монтировать удаленные ресурсы и использовать их так же как и компоненты локальной файловой системы. В версиях 2.4.x ядра реализована поддержка трех важных высокоуровневых протоколов: HTTP, NFS и SBM/CIFS.

В приведенном перечне содержатся не все протоколы, поддерживаемые ядром. Кроме указанных выше, ядро Linux позволяет работать и с другими протоколами, в частности, с различными протоколами, обеспечивающими разделение сетевых ресурсов.

Ускорение HTTP-обмена

Работа World Wide Web в основном базируется на использовании протокола HTTP (Hypertext Transfer Protocol — протокол передачи гипертекстовой информации). По сути, в ядре Linux

реализован простой сервер HTTP, который включается при установке опции Kernel HTTPd Acceleration. Для настройки и активизации этого сервера в псевдофайлы, находящиеся в каталоге `/proc/sys/net/khttpd`, записываются специальные значения. Вопросы работы со встроенным сервером HTTP подробно рассматриваются в главе 20.

Реализовать сервер HTTP в составе ядра оказалось сравнительно не сложно, так как передача клиенту статических Web-страниц (документов, содержимое которых не изменяется при различных обращениях клиентов) мало отличается от копирования файлов с диска на удаленные компьютеры. Ядро может выполнять эту операцию гораздо эффективнее, чем пользовательские программы. Для обслуживания запросов, связанных с предоставлением динамических Web-страниц, а также запросов, предполагающих сложную обработку статических документов, ядро обращается к обычному Web-серверу, например Apache. При этом нет необходимости в специальных настройках Apache; этот сервер попросту "не видит" запросов на получение статических Web-страниц.

Опции для работы с NFS

NFS (Network Filesystem — сетевая файловая система), разработанная Sun, предназначена для организации совместного использования файлов несколькими компьютерами. Благодаря NFS обращение к удаленным файлам осуществляется так же, как и обращение к файлам на локальной машине. Поддержка NFS реализована в ядре Linux. Подробно вопросы работы с NFS рассматриваются в главе 8. Для того чтобы иметь возможность монтировать каталоги, экспортируемые другими компьютерами, надо установить опции, отвечающие за поддержку NFS. Опции, включающие средства клиента и сервера NFS, содержатся в подменю Network File Systems меню File Systems (а не в меню Networking Options, как это можно было бы ожидать). Опции для работы с NFS перечислены ниже.

- NFS File System Support. Эта опция включает базовые средства поддержки клиента NFS (т. е. средства, позволяющие монтировать удаленные каталоги NFS и пользоваться ими так же, как и фрагментами локальной файловой системы).
- Provide NFSv3 Client Support. За время своего развития система NFS претерпела многочисленные изменения. Последней была разработана версия 3 (NFSv3). Поддержку этой версии необходимо задавать явно, так как стандартные средства, включаемые с помощью NFS File System Support, не обеспечивают надежную работу NFSv3. Для поддержки NFSv3 опция NFS File System Support также должна быть установлена.
- Root File System on NFS. Чтобы эта опция стала доступной, надо установить опцию IP: Kernel Level Autoconfiguration в меню Networking Options. Опция Root File System on NFS позволяет монтировать внешний каталог как корневую файловую систему Linux. Эта возможность обычно используется только на бездисковых рабочих станциях.
- NFS Server Support. Чтобы компьютер под управлением Linux мог работать как сервер NFS (т. е. мог предоставлять свои каталоги другим компьютерам), желательно установить данную опцию. В большинстве случаев она позволяет ускорить работу сервера NFS.
- Provide NFSv3 Server Support. Если вы хотите обеспечить работу сервера NFS, ориентированного на использование средств ядра, установите данную опцию. Как и в случае NFSv3-клиента, для поддержки NFSv3 должны также быть включены базовые средства NFS.

Чаще всего средства NFS используются в Linux и различных версиях UNIX. Обмен файлами с прочими системами реализуется другими средствами, одно из которых рассматривается ниже.

Опции для работы с SMB/CIFS

NFS — не единственный протокол, обеспечивающий разделение файлов. В Macintosh для этой цели используется AppleTalk; большой популярностью пользуются также протоколы IPX/SPX, разработанные Novell. В системе Linux, помимо NFS, часто применяется система Samba, которая

реализует протокол SMB (Server Message Block — блок сообщений сервера). Эти средства известны также под названием CIFS (Common Internet Filesystem — общая межсетевая файловая система). Подробно настройка и использование Samba рассматриваются в главе 7.

Samba предоставляет средства, необходимые для того, чтобы система Linux функционировала как SMB/CIFS сервер; при этом специальная настройка ядра не требуется. Если вы хотите монтировать в Linux разделяемые каталоги SMB/CIFS, вам надо установить опцию SMB File System Support, которая действует подобно опции NFS File System Support. Две подопции (Use a Default NLS и Default Remote NLS Option) обеспечивают преобразование имен файлов на основе NLS (National Language Support — поддержка национальных языков). Эти опции полезны при использовании алфавитов, отличных от латинского, например кириллицы, а также символов с дополнительными элементами.

Чтобы обеспечить работу Linux как клиента SMB/CIFS, не обязательно устанавливать опции ядра SMB/CIFS. Это также позволяет сделать программа `smbclient`. Данная программа не монтирует ресурсы; для передачи разделяемых файлов предоставляется интерфейс, подобный интерфейсу клиентской программы FTP.

Поддержка альтернативных сетевых протоколов

Несмотря на то что семейство протоколов TCP/IP, обеспечивающее работу глобальной сети Internet, пользуется широкой популярностью, в системе Linux также реализованы другие стеки протоколов. Опции, включающие поддержку различных сетевых протоколов, представлены в меню Networking Options. Многие из опций, содержащихся в этом меню, на самом деле являются подопциями TCP/IP Networking. За ними следуют опции, соответствующие другим протоколам; некоторые из них перечислены ниже.

- Asynchronous Transfer Mode (ATM). Этот набор экспериментальных опций предназначен для поддержки аппаратных средств и протоколов ATM. Опции ATM в равной мере относятся как к аппаратным средствам, так и к сетевым протоколам, но в версии ядра 2.4.x они, как и другие опции поддержки сетевых протоколов, сосредоточены в меню Networking Options.
- The IPX Protocol. Семейство протоколов IPX (Internetwork Packet Exchange — межсетевой обмен пакетами), разработанное Novell, применяется во многих сетях, в частности в Netware. Для работы с этими протоколами вам потребуется дополнительное программное обеспечение, например Mars_nwe (дополнительную информацию о нем вы можете получить по адресу `http://www.redhat.com/support/docs/tips/Netware/netware.html`). Опция NCP File System Support, расположенная в подменю Network File Systems меню File Systems, дает возможность монтировать тома Netware, подобно тому, как опции NFS и SMB/CIFS позволяют монтировать фрагменты файловой системы Windows.
- AppleTalk Protocol Support. Компания Apple разработала стек протоколов AppleTalk, предназначенный для разделения файлов и принтеров на компьютерах Macintosh. Для поддержки AppleTalk в Linux используются средства ядра, а также пакет Netatalk (`http://netatalk,sourceforge.net/`).
- DECnet Support. Корпорация DEC (Digital Equipment Corporation) разработала для своих компьютеров сетевую технологию под названием DECnet. Система Linux включает средства поддержки DECnet, но для работы с данным стеком протоколов необходимо установить специальный программный пакет. Дополнительную информацию об использовании DECnet можно получить, обратившись по адресу `http://linux-decnet.sourceforge.net`.

В системе Linux также предусмотрены опции поддержки менее популярных протоколов, например Acorn Econet. В большинстве случаев при установке системы достаточно включить поддержку TCP/IP и одного-двух дополнительных стеков протоколов. Вследствие бурного развития Internet производители, ранее использовавшие собственные стеки протоколов, модернизировали свои инструментальные средства для работы с TCP/IP. Так, например, несмотря на то, что Apple в течение

длительного времени применяла AppleTalk, средства разделения файлов на компьютерах Macintosh сейчас используют как AppleTalk, так и TCP/IP.

В ядре Linux отсутствуют средства поддержки стека протоколов NetBEUI. Для работы с разделяемыми файлами Windows в настоящее время с успехом применяются средства SMB/CIFS.

В главе 3 детально рассматриваются стеки сетевых протоколов и их использование.

Опции для работы с аппаратными средствами

В меню Network Device Support содержатся опции, определяющие взаимодействие системы с различными сетевыми устройствами. Самые важные из этих опций управляют использованием драйверов сетевых карт. Несмотря на то что в настоящее время наиболее распространены Ethernet-карты, в меню Network Device Support содержатся опции для работы и с другими устройствами, предназначенными для создания локальных сетей. Кроме того, Linux предоставляет опции, включающие драйверы устройств дальней связи и драйверы беспроводных устройств. Устройства PC Card (применяющиеся в портативных компьютерах) описаны в отдельном подменю, которое входит в состав меню Network Device Support. Вы также можете выбрать устройства, позволяющие устанавливать соединения по телефонным линиям через модемы, и другие аппаратные средства.

Для того чтобы получить доступ к описанным выше опциям, надо установить опцию Network Device Support, которая находится в начале меню Network Device Support. Если вы не сделаете это, опции в данном меню будет не доступны.

Устройства Ethernet

На момент написания данной книги, т. е. в 2002 г., подавляющее большинство локальных сетей строились на базе Ethernet. Беспроводные технологии также пользуются определенной популярностью, но сети, созданные на их основе, проигрывают сетям Ethernet в скорости обмена. Одной из проблем, усложняющих процесс администрирования операционных систем, является тот факт, что число разновидностей Ethernet-карт исчисляется сотнями и даже тысячами.

К счастью, при построении Ethernet-карт применяется ограниченный набор микросхем, поэтому для поддержки подавляющего большинства таких устройств достаточно шестидесяти драйверов. Опции, управляющие использованием этих драйверов, сосредоточены в двух подменю: Ethernet (10 or 100Mbit) и Ethernet (1000 Mbit). Большинство опций находится в первом меню. Соответствующие им драйверы, как следует из названия подменю, ориентированы на устройства, обеспечивающие скорость обмена 10 и 100 Мбод. На момент написания этой книги наибольшей популярностью пользуются Ethernet-платы 100 Мбод (100-мегабитовая Ethernet), а в некоторых случаях применяются более новые платы 1000 Мбод (или гигабитовая Ethernet). Разрабатываются также Ethernet-карты 10 Гбод.

Ethernet-сети различаются не только по скорости обмена данными, но и по типу кабеля. Для соединения устройств применяются коаксиальные кабели (в некоторых типах 10 мегабитовых Ethernet-сетей), витые пары (во всех 100-мегабитовых Ethernet-сетях, а также в некоторых типах 10-мегабитовых и гигабитовых Ethernet-сетей) и волоконно-оптические кабели (в некоторых типах гигабитовых Ethernet-сетей). Витые пары обеспечивают соединение на расстоянии до 100 метров (обычно такое соединение устанавливается между компьютером и концентратором либо коммутатором). Волоконно-оптические соединения допускают обмен данными на расстоянии до 5 километров.

Структура меню Ethernet (10 or 100Mbit) далека от совершенства. В начале меню перечислены сетевые карты 3Com, SMC, Racal-Interlan и некоторых других производителей. За ними расположены устройства ISA (Industry Standard Architecture), затем следуют устройства EISA (Extended

ISA), VLB (VESA Local Bus) и PCI (Peripheral Component Interconnect). Завершается список группой параллельных Ethernet-адаптеров. В результате для поиска требуемого устройства приходится просматривать две-три группы опций.

Существуют также Ethernet-устройства, драйверы которых не устанавливаются посредством опций меню Network Device Support или его подменю. Так, например, для устройств PC Card используются специальные драйверы, а адаптеры USB — Ethernet описаны в меню USB Support. Для того чтобы использовать устройство USB, вам надо, в зависимости от контроллера, присутствующего на материнской плате, установить либо опцию UHCI Support, либо OHCI Support, а также включить опцию, которая соответствует требуемому драйверу, например USB ADMtek Pegasus-Based Ethernet Device Support.

Альтернативные средства для создания локальных сетей

Несмотря на свою популярность, Ethernet — отнюдь не единственное устройство, позволяющее создать локальную сеть. В ядре Linux предусмотрена поддержка различных типов сетей. Число драйверов для устройств, отличных от Ethernet невелико, но это не означает, что средства для организации соответствующих сетей разработаны недостаточно хорошо. Ниже приведены некоторые опции, присутствующие в меню Network Device Support.

- Token Ring. В течение многих лет технология Token Ring, разработанная IBM, была главным конкурентом Ethernet, однако начиная с 1990 г. преимущество Ethernet стало очевидным. Большинство карт Token Ring поддерживают скорость обмена до 16 Мбод, но в настоящее время появились модели 100 Мбод. Максимальное расстояние между устройствами в сети Token Ring составляет 150–300 метров. Средства поддержки устройств Token Ring сосредоточены в подменю Token Ring Devices меню Network Device Support.
- LocalTalk. Для компьютеров Macintosh компания Apple разработала сетевую технологию, включающую протоколы как аппаратного (LocalTalk), так и программного (AppleTalk) уровня. Для взаимодействия с сетями LocalTalk были разработаны устройства x86; некоторые из них поддерживает система Linux. Соответствующие опции находятся в меню AppleTalk Devices. Как ни странно, версия Linux, разработанная для Macintosh, не поддерживает LocalTalk. На момент написания данной книги максимальная скорость обмена данными в сети LocalTalk составляла 2 Мбод.
- ARCnet. Это сетевая технология, которая в основном используется в специальных целях, например, для подключения охранных устройств или для сбора результатов научных экспериментов. Устройства ARCnet обеспечивают скорость обмена от 19 Кбод до 10 Мбод. Соединение устройств осуществляется с помощью коаксиального кабеля, витой пары или волоконно-оптического кабеля. Опции поддержки ARCnet находятся в подменю ARCnet Devices. Помимо драйвера устройства, вам необходимо включить драйвер, предназначенный для поддержки формата ARCnet-пакетов (RFC 1051 или RFC 1201).
- FDDI и CDDI. FDDI (Fiber Distributed Data Interface — волоконно-оптический интерфейс распределенных данных) и CDDI (Copper Distributed Data Interface — "медный" интерфейс распределенных данных) предназначены для создания сетей со скоростью обмена информацией порядка 100 Мбод. Преимущество FDDI перед 10 мегабитовой Ethernet состоит в том, что данная технология обеспечивает связь на расстоянии до 2 километров. Следует заметить, что гигабитовая Ethernet с передачей данных по волоконно-оптическому кабелю обеспечивает дальность до 5 километров. Для того чтобы опции ядра 2.4.17, предназначенные для поддержки FDDI/CDDI, стали доступны, надо установить опцию FDDI Driver Support.
- HIPPI. HIPPI (High Performance Parallel Interface — высокопроизводительный параллельный интерфейс) обеспечивает скорость обмена данными 800 или 1600 Кбод. При соединении

с помощью витой пары максимальная дальность составляет до 25 метров, многорежимное волоконно-оптическое соединение обеспечивает дальность до 300, а однорежимное волоконно-оптическое соединение — до 10 километров. Ядро 2.4.17 поддерживает единственное устройство HIPPI Essential RoadRunner. Заметьте, что драйвер данного устройства считается экспериментальным.

- Fiber Channel. Данный тип сетевого интерфейса поддерживает как волоконно-оптическое соединение, так и соединение с помощью обычного кабеля и обеспечивает скорость передачи данных 133–1062 Мбод. При использовании волоконно-оптического кабеля максимальная дальность составляет до 10 километров. Ядро 2.4.17 поддерживает единственное устройство Fiber Channel Interphase 5526 Tachyon.

Некоторые из описанных выше сетевых сред, например Token Ring, используются для создания локальных сетей, компоненты которых размещаются в одном здании либо в нескольких зданиях, расположенных рядом. Другие, например FDDI и HIPPI, чаще применяются для организации соединения между компьютерами, расположенными на большом расстоянии, например, находящимися в различных помещениях на территории университетского городка. Поддержка этих технологий системой Linux означает, что компьютер под управлением Linux может выступать в роли маршрутизатора, связывающего между собой различные типы сетей.

Далее в этой книге мы будем предполагать, что локальная сеть, к которой подключены компьютеры под управлением Linux, создана на базе технологии Ethernet. Если при решении конкретной задачи вам потребуется организовать поддержку других сетей, то единственное, что вам придется для этого сделать, — это изменить имя сетевого интерфейса. Устройствам Ethernet имена присваиваются следующим образом. Первое устройство имеет имя `eth0`, второе — `eth1` и т. д. Аналогично именуются другие устройства, например, первому устройству Token Ring присваивается имя `tr0`, а второму устройству FDDI — имя `fddi1`.

Устройства с широкой полосой пропускания и устройства, обеспечивающие связь на большой дальности

Термин “устройства с широкой полосой пропускания” имеет несколько значений. Во-первых, этот термин обозначает устройства, позволяющие одновременно передавать различные типы информации, например, видео, аудио и обычные цифровые данные. Во-вторых, устройствами с широкой полосой пропускания называют устройства с большой пропускной способностью, позволяющие увеличить скорость передачи данных по коммутируемым линиям (например, реализовать скорость обмена до 200 Кбод). Конечно, величина 200 Кбод выглядит более чем скромно по сравнению со скоростями, которые обеспечивает технология Ethernet, но все же 200 Кбод — это значительно больше, чем скорость 56 Кбод, которой позволяют добиться обычные телефонные линии.

Устройства с широкой полосой пропускания часто применяются в небольших компаниях для связи с серверами Internet-провайдеров или для организации взаимодействия компьютеров, расположенных далеко друг от друга. В большинстве случаев посредством устройств с широкой полосой пропускания пользователи подключают свои компьютеры к Internet. Этим данные устройства отличаются от других сетевых устройств, которые связывают несколько компьютеров в одну локальную сеть. Часто, подключая компьютеры пользователей через устройства с широкой полосой пропускания, провайдеры накладывают ограничения на их действия. Так, например, провайдер может запретить пользователю устанавливать на своем компьютере серверы.

На момент написания данной книги наиболее популярными из устройств с широкой полосой пропускания были DSL (Digital Subscribe Line — цифровая линия подписки) и кабельные модемы. Существует несколько разновидностей DSL, например ADSL (Asymmetric DSL — асимметричная

DSL) и SDSL (Single-Line, или Symmetric DSL — одиночная линия, или асимметричная DSL). В процессе работы DSL-устройства передают высокочастотные сигналы по обычным телефонным линиям. Кабельные модемы пересылают данные по кабельным телевизионным сетям и используют полосу частот свободного телевизионного калана. Некоторые из устройств с широкой полосой пропускания передают данные через спутниковые системы, по радиоканалам и по волоконно-оптическим кабелям.

Большинство соединений с широкой полосой пропускания используют специальные внешние модемы, а подключение к локальным компьютерам производится через Ethernet-порты. Поэтому, для того, чтобы работа через такое соединение стала возможной, необходим Ethernet-адаптер, кроме того, надо обеспечить поддержку этого адаптера с помощью стандартного драйвера Linux. Сам по себе широкополосный модем может работать без специального драйвера, но некоторые провайдеры используют PPPoE (Point-to-Point Protocol over Ethernet — протокол межузлового взаимодействия через Ethernet). В этом случае необходимо обеспечить поддержку данного протокола в системе Linux, для чего надо установить опцию PPP over Ethernet в меню Network Device Support (эта опция считается экспериментальной). Для того чтобы опция PPP over Ethernet была доступна, необходимо включить опцию PPP Support. Средства PPPoE понадобятся также в том случае, если вы собираетесь запускать на вашем компьютере пакет Roaring Penguin PPPoE (этот пакет находится по адресу `http://www.roaringpenguin.com/pppoe/`).

Некоторые широкополосные модемы используют вместо Ethernet интерфейс USB. В ядре 2.4.17 поддержка данных устройств не предусмотрена, но Alcatel предоставляет Linux-драйверы для своего модема Speed Touch USB DSL (`http://www.alcatel.com/consumer/dsl/supuser.htm`). Информацию о других USB-продуктах можно найти по адресу `http://www.linux-usb.org`.

Ряд широкополосных модемов, особенно, предназначенных для установления низкоуровневых ADSL-соединений, поставляются вместе с внутренними PCI-картами. Лишь немногие из этих устройств поддерживаются в системе Linux. В частности, в ядре 2.4.17 предусмотрена поддержка General Instruments Surfboard 1000 и некоторых односторонних модемов. Односторонними (one-way) называются модемы, которые могут только принимать данные. При работе с такими модемами для передачи данных используются обычные модемы, взаимодействующие по телефонным линиям. В настоящее время односторонние модемы встречаются очень редко. Драйверы для модема Diamond 1MM DSL можно найти по адресу `http://www.rodsbooks.com/network/network-dsl.html`, однако они являются модернизацией существующих Ethernet-драйверов и при использовании их в 2.4.x и более старших версиях ядра могут возникать проблемы.

Компьютеры, расположенные на большом расстоянии друг от друга, часто соединяют с помощью *выделенных линий*, которые часто называют *арендованными линиями*. Роль выделенной линии чаще всего выполняет обычная телефонная линия, арендованная у телефонной компании. По такой линии не передается тональный сигнал, все меры по организации взаимодействия модемов должны принять специалисты, занимающиеся созданием сети. Сети, созданные на базе выделенных линий, обычно называют *региональными сетями*, или WAN (Wide-Area Network). При организации региональных сетей часто используются специальные внешние устройства, называемые *WAN-маршрутизаторами*. Для создания региональных сетей могут также применяться специальные интерфейсные карты. В системе Linux предусмотрена поддержка таких устройств; соответствующие опции находятся в подменю Wan Interfaces меню Network Device Support. Как и при работе со многими другими подменю, для того, чтобы получить доступ к опциям, которые соответствуют конкретным устройствам, необходимо включить опцию Wan Interfaces Support, расположенную в начале данного подменю.

Беспроводные устройства

В течение последних лет сети, созданные на базе беспроводных устройств, становятся все более популярными. Беспроводные технологии позволяют компьютерам обмениваться данными,

даже если они не подключены к сети с помощью сетевых кабелей. Беспроводные сети очень удобно устанавливать, если прокладка кабеля по каким-либо причинам затруднена или в том случае, если пользователь, работающий на портативном компьютере, вынужден часто перемещаться и не имеет возможности подключиться к сетевому кабелю.

К сожалению, на момент написания данной книги беспроводные сети по многим характеристикам уступают Ethernet-сетям. Беспроводные сети гораздо дороже Ethernet-сетей, скорость передачи данных относительно мала, а разработка стандартов, регламентирующих структуру и работу таких сетей, еще продолжается. В настоящее время основными стандартами являются 802.11 и 802.11b. Стандарт 802.11 определяет скорость обмена 2 Мбод со снижением до 1 Мбод. *Снижением* называется повторный обмен инициализационными параметрами в случае, если уровень сигнала слишком низкий или если помехи искажают сигнал. Стандарт 802.11b определяет скорость 11 Мбод со снижением до 5,5, 2 и 1 Мбод. Существует также беспроводная технология Bluetooth, которая поддерживает скорость обмена до 1 Мбод. Основное направление развития беспроводных сетей связано с увеличением скорости передачи данных. Планируется разработать беспроводные версии ATM со скоростью обмена до 155 Мбод.

Как правило, беспроводные локальные сети создаются на базе беспроводных устройств PC Card, используемых в портативных компьютерах. В одних случаях эти устройства могут непосредственно обмениваться данными друг с другом, в других случаях для взаимодействия необходима базовая станция, которая может выполнять роль шлюза к сети, созданной традиционными средствами. С помощью базовой станции можно также организовать подключение к Internet. Существуют беспроводные ISA и PCI-карты, которые позволяют подключать к беспроводным сетям настольные компьютеры и рабочие станции. Для поддержки устройств PC Card, ISA и PCI в системе Linux необходимо установить соответствующие драйверы; для обеспечения работы базовой станции никакое специальное программное обеспечение не требуется.

Для включения средств поддержки беспроводных устройств в ядре Linux предусмотрены опции, которые содержатся в меню Wireless LAN (Non-Hamradio). Опции в данном меню расположены по типам устройств, а не по технологиям, используемым в них (например, 802.11b или Bluetooth). Кроме того, существуют пакеты Wireless Extensions и Wireless Tools, которые упрощают управление беспроводными сетями, созданными с помощью средств Linux. Информацию об этих пакетах можно найти по адресу `http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html`. Здесь же расположены дополнительные ссылки на документы, имеющие отношение к беспроводным сетям.

Устройства PC Card

Большинство портативных компьютеров имеют как минимум одно гнездо PC Card. (Часто в документации по системе Linux для обозначения устройств PC Card используется старый термин PCMCIA. Устройства PC Card можно подключать и удалять в процессе работы компьютера. Поскольку при разработке системы Linux предполагалось, что сетевые интерфейсы не должны исчезать без предупреждения, для работы с устройствами PC Card создан специальный пакет Card Services. При подключении или удалении устройств PC Card инструменты Card Services соответственно запускают или останавливают компоненты ядра, имеющие отношение к этим устройствам. Дополнительную информацию о Card Services можно получить, обратившись по адресу `http://pcmcia-cs.sourceforge.net`.

В ядре 2.4.17 опции поддержки многих устройств PC Card находятся в меню PCMCIA Network Device Support. Некоторые из опций, соответствующих беспроводным устройствам, находятся в меню Wireless LAN (Non-Hamradio). После того как вы включите соответствующую опцию и сконфигурируете карту, она будет работать как обычное устройство ISA или PCI. Так, например, Ethernet PC Card распознается системой как устройство `eth0`, а для его настройки используются стандартные инструменты, которые будут рассмотрены в главе 2.

Версии ядра, предшествующие 2.4.x, требуют для поддержки устройств PC Card специальный пакет драйверов. Следует также заметить, что многие устройства PC Card до сих пор не поддерживаются стандартным ядром Linux. Упомянутый выше пакет драйверов входит в набор Card Services. Если вы используете ядро 2.4.x, для работы с PC Card вам вряд ли придется устанавливать специальные драйверы; такие драйверы могут потребоваться для модемов, SCSI-адаптеров и других устройств.

Устройства для связи по коммутируемым линиям

Для установления связи по коммутируемой линии чаще всего используются обычные модемы. Чтобы обмен данными был возможен, необходимо также включить средства поддержки протокола PPP (Point-to-Point Protocol — протокол межузлового взаимодействия). Соединение по коммутируемой линии устанавливается из командной строки либо с помощью инструмента, предоставляющего графический пользовательский интерфейс. Подробно вопросы установления соединения будут рассматриваться в главе 2.

Для того чтобы активизировать средства поддержки PPP, надо установить опцию PPP (Point-to-Point Protocol) Support в меню Network Device Support. Если вы включите эту опцию, станут доступны некоторые дополнительные опции, например PPP Support for Async Serial Ports и PPP Deflate Compression. Эти опции не всегда необходимы, но в некоторых случаях они могут повысить эффективность обмена информацией за счет сжатия данных, передаваемых по линии связи. Если вы собираетесь использовать средства ядра PPPoE для работы через DSL-соединение, вам придется установить экспериментальную опцию PPP over Ethernet. Для некоторых дополнительных PPPoE-пакетов эта опция не нужна.

Протокол PPP часто используется при передаче информации через соединения, устанавливаемые без участия модема, например, при обмене данными между компьютерами, подключенными через последовательные порты. Следует заметить, что такое подключение применяется чрезвычайно редко, так как Ethernet-карты недорогие и обеспечивают гораздо более эффективное взаимодействие по сети. Соединение через последовательные порты обычно устанавливают на короткое время, когда нет смысла использовать сетевые карты.

PPP — не единственный протокол, посредством которого может осуществляться связь по коммутируемой линии. В ядре Linux поддерживается также протокол SLIP (Serial Line Internet Protocol — протокол Internet для обмена по последовательной линии), который выполняет практически те же функции, что и PPP. Особенности протокола SLIP таковы, что он плохо подходит для взаимодействия с Internet-провайдером, поэтому вам вряд ли придется использовать его. SLIP используется некоторыми инструментами Linux для выполнения действий на локальном компьютере. Например, утилита, поддерживающая *dial-on-demand*, т. е. устанавливающая PPP-соединение при обнаружении сетевой активности, использует SLIP для выявления попыток обращения к компьютерам за пределами локальной сети.

Помимо PPP и SLIP, для организации обмена данными между компьютерами может использоваться протокол PLIP (Parallel Line Internet Protocol — протокол Internet для обмена по параллельной линии). Как нетрудно догадаться, этот протокол применяется тогда, когда компьютеры соединены друг с другом через параллельный порт (порт принтера). Параллельный порт позволяет гораздо быстрее передавать данные, чем последовательный порт RS-232; несмотря на это, соединение через параллельный порт также применяется редко, поскольку Ethernet обеспечивает более высокое быстродействие. Для того чтобы использовать протокол PLIP, надо установить опцию PLIP (Parallel Port) Support в меню Network Device Support; при этом предварительно следует активизировать опцию Parallel Port Support в одноименном меню, а при работе на x86 надо также выбрать опцию PC-Style Hardware. Информацию об организации сети PLIP можно получить в документе *PLIP Mini-HOWTO* (`http://www.linuxdoc.org/HOWTO/mini/PLIP.html`). Если вы не можете воспользоваться кабелем Turbo Laplink, то найдете в этом документе рекомендации по изготовлению кабеля для соединения компьютеров.

Компиляция и установка ядра

До сих пор мы рассматривали опции ядра, имеющие отношение к сетевым протоколам и аппаратным средствам, используемым для соединения вашего компьютера с сетью. Компиляция ядра непосредственно не связана с обеспечением сетевого взаимодействия, однако значение этой задачи нельзя недооценивать. Чтобы добавить или удалить некоторые сетевые средства, необходимо перекомпилировать ядро. В данном разделе рассматриваются основные вопросы компиляции и установки ядра системы.

ВНИМАНИЕ
!

Если вы установили необходимые вам опции сетевого взаимодействия, это совсем не означает, что вы полностью выполнили настройку ядра. Существуют также опции, предназначенные для управления контроллерами EIDE, адаптерами SCSI, файловой системой на диске, а также многие другие опции. Несмотря на то что эти опции не рассматриваются в данной книге, они чрезвычайно важны для обеспечения нормальной работы операционной системы. Если вы неправильно установите значения соответствующих опций, система не будет загружаться либо будет работать некорректно (например, скорость обмена данными с диском может стать недопустимо низкой). Опции ядра подробно обсуждаются в документе *Linux Kernel HOWTO*, который находится по адресу `http://www.linuxdoc.org/HOWTO/Kernel-HOWTO.html` (и на многих других серверах). Вопросы настройки ядра также рассматриваются в ряде книг по Linux.

Драйверы, встроенные в ядро, и драйверы, реализуемые в виде модулей

Как вы уже знаете, при настройке ядра можно включить или отключить некоторые свойства ядра, например, вы можете разрешить или запретить использование конкретного Ethernet-адаптера. На рис. 1.1 видно, что существуют опции, значения которых можно выбирать более чем из двух возможных вариантов. В качестве примера рассмотрим опцию Packet Socket. Для этой опции может быть заданы значения Y, M и N. Значение Y (Yes) указывает на то, что средства, соответствующие данной опции, должны быть включены в основной файл ядра, а значение N (No) запрещает использование этих средств. Значение M задает некоторое "промежуточное" решение. Если вы выберете значение M, то соответствующие средства будут скомпилированы, но не войдут в основной файл ядра. Вместо этого фрагмент кода будет реализован как отдельный модуль ядра, который по мере надобности загружается или удаляется из памяти. Для опций, являющихся подопциями других опций (например, Packet Socket: Mmapped IO, показанной на рис. 1.1), значение M обычно не предусмотрено. Решение о включении их в основной файл ядра или реализации в виде отдельного модуля принимается в зависимости от значения родительской опции.

Средства, включенные в основной файл ядра, доступны с момента загрузки системы и до окончания ее работы. Ситуация, при которой фрагмент кода будет удален из памяти, возникнуть не может. Существуют опции, для которых реализующий их программный код должен быть включен в основной файл ядра. Так, например, файловая система, в которой содержится корневой каталог системы, должна быть доступна с момента загрузки, поэтому соответствующий драйвер должен быть включен непосредственно в ядро. Если вы установили рассмотренную ранее опцию Root File System on NFS, вам придется скомпилировать средства поддержки сетевых устройств и включить их в ядро.

На первый взгляд может показаться, что все средства, которые будут использоваться при работе системы, следует включить в основной файл ядра, однако такой подход имеет серьезный недостаток: при этом увеличивается объем оперативной памяти, занимаемой ядром. Кроме того, размер файла ядра на диске также увеличивается, что может создавать трудности при загрузке системы. Поэтому в системе Linux предусмотрена возможность компилировать средства, соответствующие большей части опций, в виде модулей. Это позволяет работать с ядром небольшого размера и в то же

время обеспечивает поддержку большого количества разнообразных устройств. В частности, в виде модулей могут быть скомпилированы средства для работы с большинством сетевых устройств, поэтому драйверы, включаемые в состав дистрибутивных пакетов, обычно подготавливаются именно в таком виде.

Если администрирование компьютера под управлением Linux является вашей обязанностью, то именно вам предстоит решить, следует ли использовать драйверы сетевых карт, реализованные в виде модулей, или их следует включить в состав ядра. Если вы включите драйвер сетевой карты непосредственно в ядро системы, вам не придется обеспечивать при настройке системы, чтобы перед началом сетевого обмена загружался требуемый модуль. (На самом деле система, поставляемая как дистрибутивный пакет, изначально сконфигурирована так, что данная задача решается корректно.) С другой стороны, если вы администрируете большое количество компьютеров, на которых установлена система Linux, то возможно, предпочтете создать ядро и набор модулей, которые будете устанавливать на различные компьютеры. В этом случае целесообразно реализовать драйверы в виде модулей.

Программные средства поддержки стека протоколов также могут быть непосредственно включены в ядро или скомпилированы как модули. (Исключением являются протоколы TCP/IP; их можно либо включить в основной файл ядра, либо не использовать вовсе; в виде модулей можно реализовать лишь средства, соответствующие некоторым подопциям опции, управляющей использованием данного стека протоколов.) Так, например, если вы знаете, что каталоги в файловых системах других компьютеров предоставляемые средствами NFS, будут использоваться лишь непродолжительное время, целесообразно реализовать средства поддержки клиента NFS как отдельный модуль. Поступив таким образом, вы сэкономите часть оперативной памяти в течение времени, когда средства NFS не используются, но монтирование внешних каталогов будет осуществляться несколько дольше, чем это было бы в том случае, если бы фрагмент кода, реализующий клиент NFS, был включен непосредственно в ядро.

Как вы, наверное, поняли, однозначного ответа на вопрос, надо ли включать коды поддержки опций непосредственно в ядро или их следует компилировать как отдельные модули, не существует. Я рекомендую вам сначала выяснить, насколько часто соответствующие средства будут использоваться в процессе работы системы. Если они должны использоваться постоянно, включайте их в основной файл ядра; если же они будут задействованы лишь эпизодически, компилируйте их в виде отдельных модулей. Если размер ядра становится слишком велик и при его загрузке средствами `LOADIN` (DOS-утилита для загрузки Linux) возникают проблемы, следует отдать предпочтение модульной организации ядра. Возможность загрузки с помощью `LOADIN` желательно сохранить, так как это поможет справиться с некоторыми проблемами.

Компиляция ядра

После того как вы сконфигурировали ядро системы, выполнив `make xconfig` или другую команду, приведенную в начале данной главы, вы должны скомпилировать ядро и установить его модули. Для этого необходимо выполнить следующие команды:

```
# make dep
# make bzImage
# make modules
# make modules_install
```

Первая из этих команд выполняет подготовительную работу. Слово `dep` сокращенно обозначает dependency, соответственно при выполнении команды `make dep` анализируются выбранные вами опции и определяется, какие исходные файлы зависят от других. Если вы пропустите этот шаг, компиляция будет выполнена некорректно.

В результате выполнения второй команды строится основной файл ядра, который имеет имя `bzImage` и обычно размещается в каталоге `/usr/src/linux/arch/i386/boot`. Существуют различные варианты данной команды. Например, при создании ядра небольшого размера можно

использовать команду `make zImage` (ядро в формате `bzImage` дает возможность загрузчику, например LILO, обрабатывать ядро большего размера, чем это позволяет `zImage`). Как `zImage`, так и `bzImage` представляют собой сжатые варианты ядра. Они являются стандартом для компьютеров x86, но на других платформах вам придется вместо `make bzImage` вызвать команду `make vmlinux`. В результате выполнения данной команды строится несжатое ядро. Каталог, в который помещается основной файл ядра, может отличаться от приведенного выше. Если вы работаете на компьютере, отличном от x86, вместо каталога i386 будет использован каталог, имя которого соответствует текущей платформе. Так, например, на PowerPC этот каталог имеет имя `ppc`.

Команда `make modules`, как нетрудно догадаться, компилирует модули ядра. По команде `make modules_install` файлы, содержащие эти модули, копируются в соответствующие позиции в каталоге `/lib/modules`. В частности, в каталоге `/lib/modules` создается каталог, имя которого отражает версию ядра, а в нем, в свою очередь, — подкаталоги для конкретных классов драйверов.

Команды `make dep`, `make bzImage` (или эквивалентную ей команду) и `make modules` может выполнить любой пользователь, при условии, что он обладает правами чтения и записи данных в каталогах, содержащих исходные коды ядра. Выполнить команду `make modules_install` может только пользователь `root`.

В зависимости от установленных опций и от быстродействия компьютера, процесс компиляции ядра может занять от нескольких минут до нескольких часов. Как правило, основной файл ядра создается дольше, чем модули, но если число модулей велико, для их создания может потребоваться больше времени, чем для создания ядра. При компиляции на экран монитора выводится большое число сообщений, описывающих ход обработки исходных файлов. Иногда отображаются предупреждающие сообщения, на которые можно не обращать внимание. При появлении сообщения об ошибке компиляция прерывается.

Проблемы, возникающие при компиляции ядра

Если вы корректно установили опции, компиляция ядра, как правило, проходит без проблем, но в некоторых случаях возникают ошибки. Проблемы, встречающиеся при компиляции ядра, описаны ниже.

- **Ошибки в исходном коде или несовместимость кода**. Иногда встречаются драйверы, которые содержат ошибки в исходном коде либо не совместимы с остальными компонентами ядра. Такая ситуация может возникнуть при работе с ядром, находящимся в процессе разработки, либо при попытке включить в состав ядра нестандартный драйвер. В этом случае при компиляции отображается одно или несколько сообщений об ошибке. Чтобы избавиться от ошибок, надо обновить версию ядра или, по крайней мере, заменить драйвер, который стал источником проблем. Если без этого драйвера можно обойтись, лучше вовсе отказаться от его использования.
- **Отсутствие информации о зависимости файлов**. Если работа драйвера зависит от другого драйвера, то первый драйвер должен выбираться лишь после того, как будет выбран второй. В некоторых случаях сценарий, посредством которого выполняется конфигурирование системы, работает некорректно. При компиляции ядра это проявляется следующим образом: каждый файл по отдельности компилируется нормально, но собрать файл ядра не удается. Если драйвер компилируется как отдельный модуль, то при попытке загрузить его отображается сообщение об ошибке. Иногда в сообщении об ошибке содержится информация о том, какие действия надо предпринять, чтобы избавиться от нее. В некоторых случаях решить проблему удается, вызвав команду `make dep`, а затем повторно скомпилировав ядро. Иногда работоспособное ядро можно получить, отказавшись от включения драйвера непосредственно в основной файл и скомпилировав его в виде отдельного модуля (в некоторых

случаях приходится принимать обратное решение, т. е. включать в ядро драйвер, который, будучи подготовленным в виде отдельного модуля, стал источником проблем).

- **Устаревшие объектные файлы**. Если вы компилируете ядро, а затем изменяете конфигурацию и компилируете его повторно, утилита `make` автоматически определяет, какие файлы затрагивают внесенные изменения, и повторно компилирует их. Если при работе данной утилиты возникает сбой, это может привести к ошибке при создании ядра. В одних случаях не удается собрать ядро из готовых компонентов, в других случаях ошибки возникают при компиляции отдельных файлов. Для того чтобы решить эту проблему, надо выполнить команду `make clean`, которая удалит существующие объектные файлы, а затем повторить компиляцию системы.
- **Ошибка компилятора**. GNU C Compiler (GCC) считается надежным компилятором, но бывают случаи, когда он становится источником проблем. Версия GCC, которая входит в состав Red Hat 7.0, не может скомпилировать ядро 2.2.x, но эта проблема устранена в версии ядра 2.4.x. (С Red Hat 7.0 поставляются две версии GCC; для того чтобы компиляция ядра была выполнена успешно, надо вместо `gcc` использовать `kgcc`.)
- **Проблемы с использованием аппаратных средств**. GCC более интенсивно использует ресурсы компьютера по сравнению с другими программами, поэтому сбои аппаратных средств чаще проявляются в процессе компиляции ядра системы. Такие ошибки называются *ошибками signal 11*, так как GCC возвращает именно такое сообщение. Причиной подобных ошибок, как правило, являются неисправности в процессоре и оперативной памяти. Дополнительную информацию об этих проблемах и способах их устранения можно найти по адресу `http://www.bitwizard.nl/sig11`.

Если вы не можете самостоятельно разрешить проблемы, возникающие при компиляции ядра, отправьте сообщение в одну из групп новостей, посвященных системе Linux, например в `comp.os.linux.misc`. Подробно опишите ваш дистрибутивный пакет, укажите версию ядра, которую вы пытаетесь скомпилировать, и сообщения об ошибках. (Сообщения компилятора, не связанные с возникновением ошибок, лучше не указывать).

Инсталляция нового ядра и его использование

Чтобы готовое ядро можно было использовать, его необходимо инсталлировать. Как было сказано ранее, скомпилированное ядро помещается в каталог `/usr/src/linux/arch/i386/boot` (вместо i386 может присутствовать другой каталог, имя которого отражает название процессора). Файл ядра надо скопировать или переместить в каталог `/boot`. Желательно переименовать файл так, чтобы его имя отражало версию ядра и изменения, которые вы внесли в него. Например, вы можете назвать файл ядра `bzImage-2.4.17` или `bzImage-2.4.17-xfs`. Если команда `make modules_install` до сих пор не выполнялась, надо вызвать ее, инсталлировав тем самым модули ядра в каталог `/lib/modules/x.y.z`, где `x.y.z` — это номер версии ядра.

Копирования файла ядра в каталог `/boot` недостаточно. Чтобы ядро можно было использовать, необходимо также модифицировать загрузчик. Большинство дистрибутивных пакетов содержит Linux Loader (LILO); настройка этого загрузчика на новое ядро осуществляется путем редактирования файла конфигурации `/etc/lilo.conf`. В листинге 1.1 показано содержимое файла `lilo.conf`, настроенного на загрузку одного ядра.

Листинг 1.1. Простой файл `lilo.conf`

```
boot=/dev/sda
map=/boot/map
install=/boot/boot.b
prompt
default=linux
timeout=50
image=/boot/vmlinuz
    label=linux
    root=/dev/sda6
    read-only
```

LILO используется на компьютерах x86. Работая на других компьютерах, вам придется ориентироваться на другие типы загрузчиков. Эти загрузчики во многом напоминают LILO и отличаются от него лишь некоторыми деталями. Описания этих загрузчиков можно найти в документации на конкретные дистрибутивные пакеты.

Для того чтобы при загрузке системы вы могли выбирать между старой версией ядра и вновь созданным ядром, выполните следующие действия.

1. Откройте файл `/etc/lilo.conf` в текстовом редакторе.
2. Продублируйте строки файла, описывающие текущее ядро, используемое по умолчанию. Первая из этих строк начинается символами `image=`. Описание ядра продолжается до конца файла либо до тех пор, пока не встретится выражение `image=` или `other=`. В листинге 1.1 ядро описывается в последних четырех строках.
3. Модифицируйте строку `image=` так, чтобы она указывала на новый файл ядра. Например, вместо выражения `image=/boot/vmlinuz` включите выражение `image=/boot/bzImage-2.4.17`. (Во многих дистрибутивных пакетах Linux по умолчанию загружается файл ядра `vmlinuz`.)
4. Измените строку `label=` для нового ядра, указав после символа = новое значение, например `mykernel` или `2417`. Это значение позволит отличать новое ядро от прежнего. В процессе загрузки вам будет предложено выбрать имя требуемого ядра в меню или ввести его в командной строке.
5. Сохраните внесенные изменения.
6. Введите команду `lilo` для того, чтобы установить модифицированный загрузчик на жесткий диск.

Приведенные выше инструкции предполагают, что файл `/etc/lilo.conf` не содержит ошибок. При наличии ошибок выполнение п. 6 может привести к повреждению данных на жестком диске. Включая информацию о новом ядре, не изменяйте и не удаляйте другие данные, содержащиеся в файле.

При следующей загрузке компьютера LILO предложит вам указать, какое ядро должно использоваться при работе системы. В зависимости от настройки, вы сможете либо выбрать ядро в меню, либо указать имя ядра в строке после символов `lilo:`.

Если качество нового ядра устраивает вас, модифицируйте файл `/etc/lilo.conf` так, что это ядро будет загружаться по умолчанию. Для этого надо изменить строку `default=`. Измените текст после знака =, указав имя нового ядра, которое вы присвоили ему в п. 4, а затем в командной строке снова вызовите `lilo`.

Ядро Linux можно загрузить без использования LILO. В некоторых дистрибутивных пакетах вместо LILO содержится загрузчик Grand Unified Boot Loader (GRUB). Особенности конфигурирования GRUB описаны в документации на этот загрузчик. Кроме того, для загрузки Linux также можно использовать DOS-программу `LOADIN`. Чтобы такая загрузка стала возможной, надо скопировать ядро на диск, доступный из DOS, например, в раздел DOS или на гибкий диск. Для загрузки Linux надо ввести следующую команду:

```
C:> LOADIN BZIMAGE root=/dev/sda6 ro
```

В данном примере `BZIMAGE` — это имя файла ядра, который доступен из DOS, а `/dev/sda6` — идентификатор корневого раздела, используемый в Linux. Опция `ro` указывает на то, что данный раздел должен монтироваться в режиме, допускающем только чтение (впоследствии Linux повторно смонтирует этот раздел в режиме чтения и записи). `LOADIN` — удобный инструмент, позволяющий тестировать новые версии ядра, не изменяя настройку LILO. Кроме того, `LOADIN` дает возможность загрузить Linux, если LILO поврежден. Если на вашем компьютере отсутствует система DOS, вы можете воспользоваться системой FreeDOS (`http://www.freedos.org`), которая также позволяет выполнить данную задачу. Утилита `LOADIN` входит в состав большинства дистрибутивных версий Linux. На компакт-диске она обычно располагается в каталоге `dosutils`.

Резюме

Ядро Linux непосредственно участвует в выполнении всех операций ввода-вывода, в частности в передаче данных по сети. Если компьютер под управлением Linux планируется подключать к сети, необходимо установить соответствующие опции ядра. Вы можете оптимизировать ядро для выполнения конкретной задачи, включив необходимые опции и отключив средства, которые не используются, а лишь напрасно занимают оперативную память компьютера. Большинство опций, имеющих отношение к взаимодействию по сети, располагаются в двух меню: Networking Options и Network Device Support. Эти меню включают ряд подменю. Установив требуемые опции, надо скомпилировать ядро системы, для чего необходимо выполнить несколько команд. Для того, чтобы обеспечить загрузку ядра, следует изменить конфигурацию LILO.