

Файловая система FreeBSD

- ▶ Структура каталогов FreeBSD
- ▶ Мониторинг использования файловой системы
- ▶ Монтирование и демонтажение файловых систем FreeBSD
- ▶ Монтирование и демонтажение файловых систем *других операционных систем*
- ▶ Монтирование и демонтажение файловых систем CD-ROM и флоппи-дисков
- ▶ Понимание файла `/etc/fstab`
- ▶ Проверка и восстановление файловых систем с помощью утилиты `fsck`
- ▶ Установка и использование пользовательских квот файловой системы

Чтобы понимать, как работает операционная система FreeBSD, необходимо четко представлять, как она управляет файлами. Важно также и то, чем ее методы отличаются от методов других операционных систем.

FreeBSD использует **FFS** (BSD Fast File System, Система быстрого доступа к файлам BSD). Широко распространено заблуждение, что в современных версиях UNIX семейства BSD применяется **UFS** (**Universal File System**, Универсальная файловая система) или **UNIX File System** (Файловая система UNIX). Пользователи даже по-разному интерпретируют эту аббревиатуру! Причина этого заблуждения кроется в том, что многие утилиты, о которых будет рассказано в этой главе (например, mount), ссылаются на UFS как на файловую систему по умолчанию. Никто не возражает и вам упоминать о файловой системе BSD как о UFS, но при этом следует помнить, что "настоящая" UFS использовалась лишь в ранних версиях семейства BSD и не применяется во FreeBSD.

FFS — распространенная файловая система, используемая во FreeBSD, OpenBSD, NetBSD и других системах, включая Mac OS X (чья исходная версия, Darwin, была основана на ядре FreeBSD). В Linux, как правило, применяется Ext2FS, в Windows NT — NTFS, а Windows 95/98/Me — VFAT. Эта информация будет полезна при обсуждении монтирования файловых систем, принадлежащих другим операционным системам.

ПРИМЕЧАНИЕ

Более полный обзор различных типов файловых систем, существующих в современном компьютерном мире, можно найти по адресу <http://iimw.penguin.cz/~mhi/fs/>.

Структура каталогов FreeBSD

Для тех, кто работал с UNIX или UNIX-подобной операционной системой, структура каталогов FreeBSD, без сомнения, покажется знакомой (см. рис. 9.1).

Тем не менее между файловой системой FreeBSD и файловыми системами операционных систем Linux, Solaris и других вариантов UNIX существуют ключевые различия. Для пользователей, работавших в Windows или Macintosh, структура каталогов файловой системы FreeBSD покажется таинственной. Как и во многих других вопросах, связанных с UNIX, корни традиционной схемы именования каталогов теряются в далеком прошлом. Мы постараемся пролить свет на эту тайну, по крайней мере, давайте познакомимся с этой системой (см. табл. 9.1).

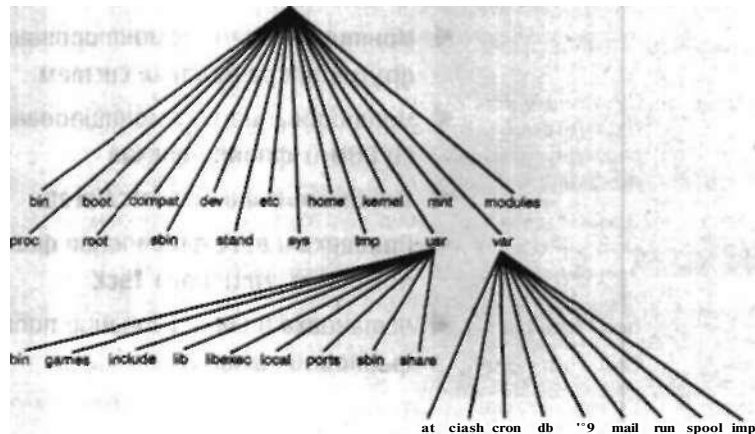


Рисунок 9.1

Иерархическая структура файловой системы FreeBSD, которая начинается с "/", то есть с корневого каталога

С помощью команды `ls /` можно увидеть содержимое корневого каталога файловой системы FreeBSD. Обратите внимание на следующие элементы: каталоги обозначены `/`, символические ссылки — `@`, а выполняемые файлы — `*`.

Таблица 9.1 Ключевые

и FreeBSD

Каталог	Назначение
<code>bin/</code>	Здесь находятся выполняемые файлы, скомпилированные со статическими библиотеками. Поэтому этим каталогом можно воспользоваться в случае аварийной загрузки, когда нет доступа к программам, использующим динамические библиотеки, или другим файловым системам, кроме <code>/</code> . Здесь находятся конфигурационные и исполняемые файлы, необходимые для загрузки системы. Кроме того, начиная с версии FreeBSD 5.0 в этом каталоге размещается ядро. Оно управляет всеми устройствами и сетью, а также выполняет ряд других задач. Подробную информацию о ядре вы найдете в главе 17.
<code>compat/</code>	Символическая ссылка на структуру каталогов, обеспечивающую совместимость с другими операционными системами, например с Linux.
<code>dev/</code>	Специальный каталог. Файлы внутри него являются устройствами, точнее, файлами специальных типов, которые обеспечивают программам интерфейс к устройствам, поддерживаемым ядром.
<code>etc/</code>	В ранних версиях системы здесь размещались разнообразные файлы, которые не могли быть помещены в другие каталоги. Теперь здесь хранятся системные конфигурационные файлы, в том числе базы данных пользовательских паролей и сценарии начальной загрузки.
<code>home@</code>	В зависимости от настройки системы этот элемент может быть каталогом, а может быть символической ссылкой на <code>/usr/home</code> . В нем собраны все начальные каталоги обычных пользователей.
<code>mnt/</code>	Пустой каталог, предназначенный для монтирования других дисков.
<code>modules/</code>	Здесь находятся загружаемые модули ядра.
<code>proc/</code>	Файловая система процессов (<code>procfs</code>). Она представляет собой интерфейс к таблице процессов. Используется некоторыми программами, однако не является важным компонентом работы операционной системы (т.е. ее можно благополучно демонтировать).
<code>root/</code>	Начальный каталог пользователя <code>root</code> . Он расположен за пределами каталога <code>/home</code> по соображениям безопасности и, кроме того, доступен при аварийной загрузке.
<code>sbin/</code>	Системные исполняемые файлы, скомпилированные со статическими библиотеками. Они отличаются от файлов, которые хранятся в <code>/bin</code> , тем, что способны изменять поведение системы, тогда как программы из <code>/bin</code> представляют собой лишь пользовательские утилиты.
<code>stand/</code>	Здесь содержится набор жестких ссылок на программы (<code>hard-linked programs</code>), которые обеспечивают среду "мини-FreeBSD" для инсталляции системы и аварийного запуска в автономном режиме (<code>standalone mode</code>). Скорее всего, вам понадобится лишь одна программа из этого каталога, — sysinstall . О ней рассказано в главе 2.
<code>sys@</code>	Ссылка на файлы с исходным кодом ядра, если они были установлены.
<code>tmp/</code>	Временные файлы. Данный каталог доступен для записи всем пользователям.

Каталог Назначение

<code>usr/</code>	"Шлюз" к остальной части системы: программы, использующие динамические библиотеки, пользовательские файлы и программы, устанавливаемые администратором. В последующих главах будет подробно обсуждаться содержимое этого каталога.
<code>var/</code>	Изменяющиеся файлы. Здесь содержатся runtime-файлы, используемые программами, log-файлы, спул (spool) и другие элементы, чье содержимое изменяется в процессе нормальной работы операционной системы.

Перечисленные выше файлы и каталоги составляют ядро файловой системы FreeBSD. Следует отметить, что между структурой каталогов FreeBSD и UNIX-подобных операционных систем, например, Linux, существуют важные различия.

Структура FreeBSD является строго контролируемой, так как следует правилу: "Все, что устанавливает администратор, размещается в каталоге `/usr/local`". Некоторые операционные системы предоставляют пользователям определенную свободу при установке программ, но только не FreeBSD — в ней поддерживается четкая структура портированных программ и пакетов (см. главу 15). Хотя программа может по умолчанию размещать свои библиотеки в каталоге `/var/lib`, а конфигурационные файлы — в `/etc`, FreeBSD изменяет сценарии инсталляции таким образом, чтобы соответствующие файлы попали в каталоги `/usr/local/lib` и `/usr/local/etc`. Фактически, все конфигурационные файлы любого программного обеспечения размещаются в каталоге `/usr/local/etc`, а если программа устанавливает сценарий, запускать который нужно на этапе начальной загрузки системы, то он находит свое место в каталоге `/usr/local/etc/rc.d`. Содержимое этого каталога выполняется при загрузке системы после запуска основных сценариев из аналогичного каталога `/etc/rc.d`.

Следствием столь строго управляемой структуры является достаточно простая поддержка операционной системы FreeBSD, особенно ее воссоздание на новой машине (это нужно делать, например, при обновлении оборудования). Теоретически, все содержимое каталога `/usr/local` можно скопировать с одной машины на другую и все программное обеспечение будет работать также успешно, как и ранее. Однако на самом деле это довольно рискованная операция, которая зачастую приводит к непредвиденным последствиям. Тем не менее это идеал, к которому стремится FreeBSD.

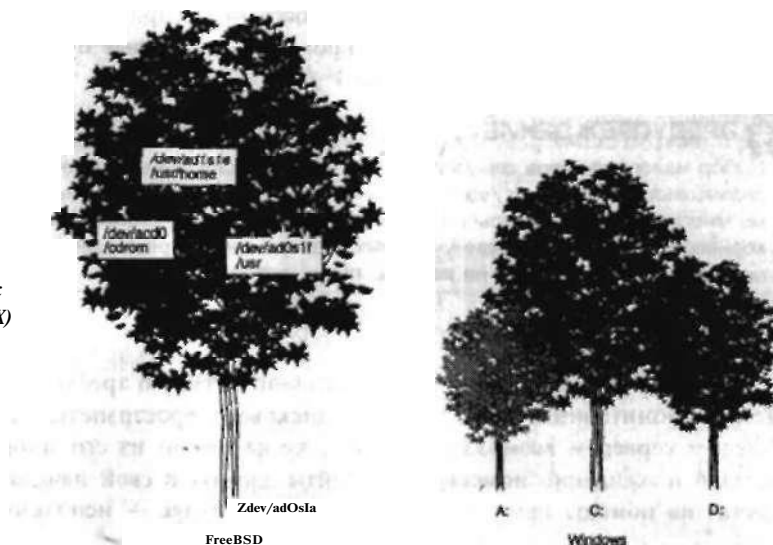
Для тех, кто привык к Linux или Solaris, FreeBSD имеет очевидный недостаток — те усилия, которые, вполне возможно, придется приложить для переноса программ с других систем во FreeBSD. Так, в частности, придется изменить пути к файлам. Например, программы, написанные на языке Python для Linux, где интерпретатор Python вызывается как `/usr/bin/python`, не будут выполняться во FreeBSD, поскольку Python, не являясь частью основной инсталляции, следовательно, должен находиться в каталоге `/usr/local/bin/python`. Все это несложно исправить для одного-двух файлов, но при переносе инсталляции, содержащей сотни подобных программ, это уже проблематично. В интересах более легкой совместимости платформ проще нарушить структурные требования FreeBSD, создав символическую ссылку `/usr/bin/python`, указывающую на `/usr/local/bin/python`.

Более подробное описание организации файловой системы FreeBSD предоставляет команда `man hier`.

Мониторинг использования файловой системы

Одна из концепций UNIX-подобных операционных систем, незнакомая пользователям Windows, заключается в идее *точек монтирования* (mount points). В Windows/DOS каждому диску системы присвоена буква (например, C:), и каждый из них имеет независимую файловую систему. Например, машина с двумя дисками и приводом CD-ROM имеет диски C:, D: и E:, с которыми может работать пользователь. В UNIX дело обстоит иначе. Существует только одна общесистемная структура каталогов, и *все* диски *монтируются как различные точки* в ней. Здесь уместна *следующая аналогия*: система Windows подобна винограднику, в котором кусты расположены в ряд, UNIX же представляет собой одно большое дерево, где меньшие деревья примыкают к стволу как ветки и вместе формируют единую иерархическую крону (см. рис. 9.2).

Рисунок 9.2
Структура файловых систем FreeBSD (UNIX) и Windows.



Преимуществом структуры последнего типа является легкость, с которой в систему можно добавить дисковое пространство: достаточно просто подмонтировать новый диск или раздел в какой-либо точке иерархии (мы продемонстрируем такой подход далее в этой главе). Недостаток же заключается в том, что в такой файловой системе гораздо сложнее переместить все содержимое с одного диска на другой. В *Windows* или *Mac OS* это осуществляется при помощи простого перетаскивания, поскольку каждый диск имеет отдельную файловую систему. Во FreeBSD такая задача относится к разряду более громоздких. Таким образом, иерархическая структура больше подходит для сервера, оборудование которого не меняется в течение долгого времени, чем для рабочей станции, на которой данные зачастую "живут" дольше, чем аппаратное обеспечение.

Команда `df`

Команда `df` (disk free — свободное пространство) — самый простой способ получить информацию об используемом дисковом пространстве; ознакомление с выво-

дом команды `df` является частью ежедневной проверки состояния системы (подробнее об этом см. главу 14). Вывод команды содержит основную информацию о файловых системах и именах устройств, используемых ими.

Вывод команды `df` выглядит следующим образом:

Filesystem	1k-blocks	Used	Avail	Capacity	Mounted
/dev/ad0s1a	49583	28427	17190	62%	/
/dev/ad0s1f	4254901	1959405	1955104	50%	/usr
/dev/ad0s1e	19815	12058	6172	66%	/var
proofs	4	4	0	100%	/proc

Каждый слайс (`slice`) или раздел (`partition`) считается файловой системой и может быть смонтирован в любую точку структуры каталогов. Вывод команды показывает, что система имеет три слайса на основном диске IDE (`/dev/ad0`), они смонтированы как `/`, `/usr` и `/var`. Такие установки производятся при инсталляции FreeBSD по умолчанию. Обратите внимание, что корневой файловой системе (`/`) отведено всего 50 Мб, а `/var` — 20 Мб. Остальное дисковое пространство принадлежит `/usr`. Это значит, что по умолчанию предполагается, что программы и данные будут размещаться главным образом в каталоге `/usr` и частично в `/var`.

Выбор малого размера для `/var` может оказаться опасным, поскольку `log`-файлы традиционно размещаются в каталоге `/var/log`. Эти файлы могут достигать очень больших размеров, поэтому многие администраторы предпочитают перемещать каталог `/var` в `/usr/var`, создавая в корневом каталоге необходимую символическую связь. Преимущества и недостатки различных методик разбиения диска на разделы обсуждаются в главе 19.

Команда `du`

Команды `df` иногда бывает недостаточно: зачастую требуются более специфичные методы мониторинга использования дискового пространства. Так, при управлении сетевым сервером возможна ситуация, когда любой из его многочисленных пользователей неожиданно помещает гигабайты данных в свой начальный каталог. В этом случае на помощь приходит команда `du` (`disk usage` — использование диска).

```
# du -d 1 /home/
22572 /home/bob
9 /home/fred
31 /home/alice
1520 /home/torn
66211 /home/pat
```

Команда `du` рекурсивно выводит размер всех каталогов, расположенных в текущем каталоге или в указанном в качестве параметра. Параметр `-d`, сопровождаемый числом, задает глубину рекурсии (без него вывод команды будет чрезвычайно велик). Опция `-s` задает режим сводки (`summary`), в этом случае вывод содержит одну строку с суммарным размером указанного каталога. Перечень опций команды `du`, включая обработку символических ссылок, можно найти на странице справочного руководства: `man du`.

Должность системного администратора накладывает на него бремя слежения за файловыми системами. Было бы неплохо, если бы система могла выполнять эту задачу самостоятельно. И такое решение существует: это *квоты* (`quotas`). К их обсуждению мы вернемся в конце этой главы.

Монтирование и демонтаж файловых систем FreeBSD

Здесь мы продемонстрируем многосторонность UNIX-подобных файловых систем. Предположим, что система заполнила один диск (**/dev/ad0**) и администратор добавляет другой диск как вторичное устройство первого контроллера (primary slave). Система опознает его как **/dev/ad1**. (Диски SCSI обозначаются как **/dev/da0** и т.д.) После разбиения диска на разделы и указания его метки (эта процедура подробно рассмотрена в главе 19) появится одна или несколько файловых систем, которые можно добавить в любую точку структуры каталогов системы. (Это напоминает прививку новой ветки к дереву.)

Команда mount

Предположим, например, что в систему постоянно добавляются пользователи, которые норовят заполнить свои начальные каталоги файлами большого размера. Становится очевидным (благодаря командам **df** и **du**), что раздел **/usr** почти полон — в основном за счет **/usr/home**. Не остается ничего другого как добавить новый диск, предназначенный для хранения начальных каталогов пользователей.

Покупается новый жесткий диск размером 50 Гб и разбивается на три раздела в пределах одного слайса FreeBSD — **/dev/ad1s1e** (100 Мб), **/dev/ad1s1f** (8 Гб) и **/dev/ad1s1g** (40 Гб). Третий (наибольший) раздел предназначен для начальных каталогов пользователей, остальные два — для добавления дискового пространства в других частях системы. В данный момент нас интересует только превращение старого раздела **/home** в новый 40-гигабайтный раздел, не содержащий только начальные каталоги пользователей.

У вас **/home** — это, скорее всего, символическая ссылка на **/usr/home**, которую следует удалить командой **rm /home**. (Реальный каталог **/usr/home** при этом не пострадает.) Если же **/home** — каталог, его необходимо переименовать, например, **mv /home /home.old**.

Теперь для новой файловой системы нужно создать *точку монтирования*. Точки монтирования должны быть каталогами, необязательно пустыми; однако следует помнить, что после монтирования устройства в непустой каталог, его содержимое становится недоступным. (Для одновременного доступа требуются объединенные файловые системы (union filesystems), которые на момент написания этой книги еще не поддерживались.) Для создания новой точки монтирования выполняется команда **mkdir /home**.

Теперь все готово к монтированию новой файловой системы. Для стандартных файловых систем FreeBSD оно выполняется командой **mount** без параметров. 40-гигабайтный раздел обозначается как **/dev/ad1s1g** и монтируется командой:

```
# mount /dev/ad1s1g /home
```

Если выполнение команды происходит без ошибок, новой файловой системой сразу же можно пользоваться. Проверить это позволяет команда **df**:

Filesystem	IK-blocks	Used	Avail	Capacity	Mounted on
/dev/ad0s1a	49563	28427	17190	62%	/
/dev/ad1s1g	39245453	3362491	32119340	12%	/home
/dev/ad0s1f	4254901	892410	3140012	22%	/usr
/dev/ad0s1e	19815	12058	6172	66%	/var
proofs	4	4	0	100%	/proc

Все прошло успешно! Теперь все файлы из `/usr/home` (или `/home.old`, в зависимости от настроек вашей системы) можно переместить в новую файловую систему `/home`.

Однако подобные операции не всегда проходят так легко. Монтирование файловых систем — это одна из тех областей системного администрирования, где потенциально существуют ловушки для неосторожных сисадминов. Так, например, команда может выдать сообщение об ошибке — `Incorrect super block` (Некорректный супер-блок) или еще менее информативное — `Invalid argument` (Неверный аргумент). Обычно причиной таких сообщений является неправильное указание имени устройства в командной строке `mount` (соглашение об именах во FreeBSD действительно может запутать администратора). Одна и та же файловая система может быть обозначена как `/dev/ad1s1e`, как `/dev/acd0` или как `/dev/ad3s1` — в зависимости от того, адресуется она в режиме слайсов (`slice mode`) или в специализированном режиме (`dedicated mode`). Кроме того, определенные имена устройств с суффиксами могут быть взаимозаменяемы с короткими именами. Все эти тонкости более подробно обсуждаются в главе 19.

Команда `mount` имеет опцию `-f`, которая требует произвести операцию монтирования в любом случае. Но если файловая система не монтируется, то для этого, видимо, существует определенная причина. Имеет смысл выяснить и устранить ее, а не использовать опцию `-f`. Такой причиной может оказаться неформатированный диск, файловая система неизвестного типа или "грязная" файловая система, т.е. система, останов которой был выполнен некорректно. В последнем случае файловая система может иметь ряд несоответствий, которые необходимо вначале устранить утилитой `fsck` (как это сделать, будет показано далее в этой главе), а уж потом выполнять монтирование.

Весьма полезно поле `options`, в котором указываются кодовые слова (флажки), разделенные запятыми. С их помощью устанавливаются такие свойства, как доступ к файловой системе только для чтения или для чтения-записи. По умолчанию возможен доступ для чтения-записи, но опция `-r` (или `rdonly`) позволяет монтировать файловую систему только для чтения. Полный список опции `mount` можно получить, выполнив команду `man mount`.

Команда `umount`

Приходит время, когда файловую систему необходимо демонтировать — для этого служит команда `umount` (а не "unmount"). Чтобы демонтировать файловую систему `/home`, используется следующая команда:

```
# umount /home
```

Тот же результат дает и команда `umount /dev/ad1s1g`, а `umount -a` демонтирует все файловые системы, кроме корневой.

Демонтирование файловых систем представляет собой более простую процедуру, чем их монтирование. Однако демонтируемая файловая система не должна использоваться в момент этой операции. Это значит, что такие системы, как `/usr` и `/var`, как правило, можно демонтировать только в однопользовательском режиме. В примере с разделом `/home` все подключенные к системе пользователи будут находиться, скорее всего, в своих начальных каталогах, поэтому, прежде чем демонтировать раздел `Доме`, их необходимо отключить. Чаше всего, начинающие пользователи, экспериментируя с командами `mount` и `umount`, этого не понимают.

Запомните: *нельзя находиться внутри демонтируемой файловой системы*/ Иначе команда демонтирования выдаст сообщение об ошибке: `Device busy` (Устройство

занято). Поэтому возьмите себе за правило: перед демонтированием любой файловой системы выполнять команду `cd /`.

Как и `mount`, команда `umount` имеет опцию `-f`, приводящую к насильственному демонтированию. И опять-таки прибегать к ней следует лишь в случае крайней необходимости; беспорядочные операции с файловой системой могут потенциально дестабилизировать работу всей операционной системы.

Монтирование и демонтирование файловых систем других операционных систем

Все изложенное выше применимо к стандартным файловым системам FreeBSD. Что же делать, если нужно смонтировать диск, принадлежащий другой операционной системе, например, Linux, Windows 98 или NT? Такую операцию тоже можно выполнить. В основном ядре FreeBSD по умолчанию поддерживает файловые системы, перечисленные в табл. 9.2.

Таблица 9.2 Файловые системы, поддерживаемые стандартным ядром

<i>Файловая система</i>	<i>Название</i>
FFS	Быстрая файловая система (Berkeley Fast Filesystem)
MFS	Файловая система в памяти (Memory Filesystem)
NFS	Сетевая файловая система (Network Filesystem)
MSDOSFS	Файловая система MS-DOS (MS-DOS Filesystem)
CD9660	Файловая система ISO 9660 (CD-ROM)
PROOFS	Файловая система процессов (Process Filesystem)

Поддерживаются и другие виды файловых систем, однако для их использования необходимо заново скомпилировать ядро с соответствующими опциями (см. главу 17).

Таблица 9.3 Дополнительные файловые системы

<i>Файловая система</i>	<i>Название</i>
FDDESC	Файловая система дескрипторов (File Descriptor Filesystem)
KERNFS	Файловая система ядра (Kernel Filesystem)
NTFS	Файловая система Windows NT (NT Filesystem)
NULLFS	Файловая система NULL (NULL Filesystem)
NWFS	Файловая система Novell NetWare (NetWare Filesystem)
PORTAL	Файловая система Portal (Portal Filesystem)
UMAPFS	Файловая система UID map (UID map Filesystem)
UNION	Объединенная файловая система (Union Filesystem)
CODA	Файловая система CODA (CODA Filesystem)
EXT2KS	Файловая система Ext2 (Ext2 Filesystem) (Linux)

Некоторые из этих файловых систем стабильнее, чем другие; как было сказано ранее, объединенные файловые системы, чья разработка еще не завершена, могут

повредить системе, поэтому монтировать их не рекомендуется. По поводу завершенности и надежности Ext2FS и NTFS также высказывалось немало опасений. Если файловая система не была включена в стандартное ядро, для этого, видимо, имелись определенные причины.

В идеальном случае, файловая система, не включенная в ядро, поддерживается как загружаемый модуль в каталоге /modules; такой модуль загружается автоматически при попытке монтирования файловой системы. Именно так обстоит дело со многими системами, включая CODA, PORTAL, NWFS, NULL, NTFS, UNION и другими типами, включенными в стандартное ядро, как модули.

Для монтирования файловых систем, поддерживаемых ядром по умолчанию (например, MSDOSFS), и даже тех, которые изначально не скомпилированы с ядром, FreeBSD имеет несколько удобных утилит. Они находятся в каталоге /sbin:

```
mount_cd9660*      mount_mfs*        mount_portal*
mount_devfs*      raount_rasdos*   mount_procfa*
mount_ext2fs*     mount_nfs*        mount_atd*
mount_fdsc*       mount_ntfs*       mount_ump*
mount_kemfs*      mount_null*       mount_union*
mount_linprocfs*  mount_nwfs*
```

Каждая из них похожа на mount и работает аналогичным образом. Фактически, утилиты mount_* представляют собой расширения опции -t команды mount. Эта опция поддерживается только для нескольких внутренних типов файловых систем. Если задан неизвестный аргумент, то происходит вызов соответствующей утилиты из предыдущего списка. Например, команда mount -t nfs /mat, на самом деле, заменяется на mountnfs /mnt. Зная это, проще сразу запустить необходимую утилиту.

Что касается стабильности и степени завершенности частично поддерживаемых файловых систем, рекомендую обратиться к страницам справочных руководств **man mount.***. Они рассказывают, чего именно не хватает в поддержке конкретной файловой системы. Например, из **man mount_nfs** можно узнать о проблемах, которые могут возникнуть при записи файлов, и об отсутствии поддержки компрессии, а **man mount_union** предупреждает об ограниченной функциональности этой файловой системы и потенциальной возможности потери данных. Обязательно ознакомьтесь с этими руководствами!

Монтирование файловой системы Windows/MS-DOS

Вначале рассмотрим монтирование диска Windows 98 как файловой системы MSDOSFS во FreeBSD. Для этого применяется команда mount_msdos. Она имеет несколько специальных опций, о которых рассказывать здесь нет необходимости, например, -W и -L, управляющих локальной кодовой таблицей, применяемой в длинных именах файловой системы FAT32/VFAT. В примере мы ограничимся установками по умолчанию, т.е. кодировкой ISO 8859-1, а также предположим, что чтение и запись в файловой системе производятся с правами пользователя root.

```
# mount_msdos /dev/ad1si /mnt
```

Помните: то, что в системах DOS и Linux называется "разделами" (partitions) во FreeBSD называется слайсами (slices), дабы избежать путаницы с традиционными разделами (partitions) BSD, являющимися подразделами слайсов. Поскольку файловая система

Windows 98 на устройстве /dev/ad1s1 использует слей, а не раздел, для обращения к ней достаточно указать номер слайса без дополнительных суффиксов, которые использовались для монтирования стандартных файловых систем FreeBSD, как было рассказано в начале этого раздела (там именем устройство было /dev/ad1slg).

Если используется дополнительный раздел DOS (extended DOS partition), команда может выдать ошибку — Invalid argument (Неверный аргумент). В этом случае обратите внимание, что разделы внутри дополнительного раздела DOS нумеруются начиная с 5, поэтому имя первого такого устройства: /dev/ad1s5.

При монтировании файловой системы MS-DOS возможны и дополнительные шаги: монтирование с UID/GID пользователя, не являющегося root; монтирование с маской прав доступа, управляющей доступом пользователей к содержимому файловой системы; вывод длинных имен файлов, а не коротких и т.д. Полностью все опции изложены на странице справочного руководства man mount msdos.

Флоппи-диск Windows содержит имена файлов, которые могут отображаться в стандартном коротком формате MS-DOS 8.3 или же в длинном формате Windows 95/98 (при этом дополнительная информация извлекается из метаданных файловой системы). Если дискета монтируется с помощью команды **mount msdos**. FreeBSD попытается найти на диске длинные имена файлов. В случае успеха все имена будут отображаться в длинном формате. Если же система обнаружит только короткие имена, будет использоваться короткий формат. Заставить команду **mount msdos** принудительно отображать имена файлов в длинном или коротком формате можно при помощи опции **-l** или **-e**. соответственно.

Монтирование файловой системы Linux

В большинстве Linux-систем используется файловая система Ext2FS. Поскольку поддержка Ext2FS не включена в стандартное ядро, монтирование этой файловой системы требует дополнительного шага. (Поддержка Ext2FS вскоре будет доступна в виде загружаемого модуля, хотя не момент написания книги ее еще не было.) Действия по компиляции и установке ядра обсуждаются в главе 17. При этом необходимо добавить следующую строку:

```
options      EXT2FS
```

После того как в системе запущено новое ядро, для монтирования файловой системы Linux достаточно воспользоваться командой:

```
# mount_ext2fs /dev/ad1s1 /mnt
```

За исключением поддержки в ядре, команда mount_ext2fs ведет себя почти идентично стандартной mount, поэтому сюрпризов не предвидится.

ИСПОЛЬЗОВАНИЕ fdisk ДЛЯ ПОЛУЧЕНИЯ ИНФОРМАЦИИ О РАЗДЕЛАХ

Определить, с какой файловой *системой* предстоит работать, *позволяет команда* ftisk:

```
* fdisk /dev/ad1
***** working on device /dev/ad1 *****
parameters extracted from in-core disklabel are:
cylinders=1247 heads=255 sectors/track=63 (16065 blks/cyl)
Figures below won't work with BIOS for partitions not in cyl 1
parameters to be used for BIOS calculations are:
cylinders=1247 heads=255 sectors/track=63 (16065 blks/cyl)
```

```

Media sector size is 512
Warning: BIOS sector numbering starts with sector 1
Information from DOS bootblock is:
The data for partition 1 is:
sysid 131,(Linux filesystem)
    start 63, size 2104452 (1027 Meg), flag 0
    beg: cyl 0/ sector 1/ head 1;
    end: cyl 130/ sector 63/ head 254
The data for partition 2 is:
sysid 130,(Linux swap or Solaris x86)
    start 2104515, size 787185 (384 Meg), flag 0
    beg: cyl 131/ sector 1/ head 0;
    end: cyl 179/ sector 63/ head 254
The data for partition 3 is:
sysid 131,(Linux filesystem)
    start 2891700, size 17141355 (8369 Meg), flag 0
    beg: cyl 180/ sector 1/ head 0;
    end: cyl 1023/ sector 63/ head 254
The data for partition 4 is:
<ONOSKD>

```

Каждый раздел (или слайс в терминологии FreeBSD) имеет системный идентификатор (**sysid**). Файловой системе Linux Ext2FS отвечает 131. FreeBSD — 165. Коды других файловых систем, которые распознает утилита **fdisk**, находятся в файле `/usr/src/sbin/i386/fdisk/fdisk.c`

Монтирование и демонтаж файловых систем CD-ROM и флоппи-дисков

После изучения монтирования различных файловых систем перейдем к сменным носителям — CD-ROM и флоппи-дискам. Диски CD-ROM обычно монтируются как CD9660, дискеты же, как правило, могут принадлежать либо FFS (стандарт FreeBSD), либо MS-DOS.

Монтирование компакт-дисков и дискет

В случае CD-ROM основная задача состоит в установлении правильного имени устройства. Приводы IDE обычно имеют имена вида `/dev/acd0c`, приводы SCSI — `/dev/cd0c`, а различные нестандартные приводы могут иметь и другие префиксы. Суффикс `c` указывает, что система адресуется ко всему диску в "специализированном" режиме (*dedicated mode*).

О текущем соглашении об именах можно узнать по адресу <http://www.FreeBSO.org/handbook/disks-naming.html>.

```
< mount_cd9660 /dev/acd0c /cdrom
```

Монтирование флоппи-диска также не вызывает затруднений. Скорее всего, имя нужного устройства это `/dev/fd0`; его не следует путать с записью, которую содержит ядро, — `fdc0`. Последняя отвечает реальному контроллеру флоппи-дисков шины ISA, тогда как имена `fd0` и `fd1` указывают на приводы, присоединенные к этому контроллеру.

```
# mount /dev/fd0 /floppy
# mount_msdos /dev/fd0 /floppy
```

Следует проявить осторожность: флоппи-диски могут быть защищены от записи, а диски CD-ROM физически доступны только для чтения, однако FreeBSD на этапе монтирования не проверяет возможность записи на носитель! Если монтировать защищенную от записи дискету без опции `-g` или `rdonly`, то при попытке записи возникнет ошибка ввода-вывода. Что произойдет дальше, зависит от стабильности приложения, осуществляющего запись, — начиная от простого вывода сообщения об ошибке и заканчивая полным зависанием системы. Поэтому при монтировании защищенной от записи дискеты или диска CD-ROM средствами командной строки обязательно указывайте опцию `-g` или `rdonly`, предотвращающую попытку записи на такой носитель на уровне ядра!

Демонтирование компакт-дисков и дискет

Поскольку компакт-диски и дискеты являются сменными носителями, существует потенциальный риск удалить носитель до того, как он будет реально демонтирован. Для сравнения: Windows динамически монтирует и обновляет состояние устройств при каждом обращении для чтения или записи. В Mac OS монтирование диска и физическая вставка-удаление носителя полностью контролируются программным обеспечением, что делает их взаимозависимыми. FreeBSD (и другие версии UNIX для x86) не имеют подобных излишеств, поэтому системный администратор должен сам заботиться о том, чтобы все смонтированные носители находились на месте.

Большинство приводов CD-ROM блокируется при монтировании носителя операционной системой (этому способствует программный механизм открытия устройства). Они не открываются в ответ на нажатие кнопки до тех пор, пока устройство не будет демонтировано. Дискеты, к сожалению, можно вытащить в любой момент, а компакт-диски — аварийно, открыв привод при помощи скрепки. Если смонтированное устройство удалено из привода до демонтажа, а программа пытается произвести чтение или запись, произойдет дестабилизация работы системы, о которой говорилось ранее.

Напрашивается следующий вывод: следует всегда помнить о демонтаже компакт-диска (`umount/cdrom`) или дискеты (`umount/floppy`) и выполнять эту операцию перед удалением носителя из привода. Это гарантирует стабильную работу системы.

Другие сменные носители

Количество сменных носителей, появляющихся на рынке, постоянно растет. Сегодня они включают в себя внешние устройства USB и FireWire, перезаписываемые компакт-диски и DVD. Приводы Zip и подобные им устройства стали уже достаточно привычными. FreeBSD поддерживает приводы Zip для параллельного порта как устройство `vr0`, а USB-приводы Zip — как `umass`. Вскоре к ним присоединятся десятки других устройств, каждое из которых имеет свои особенности по записи, смене носителя и монтируемости. Здесь мы рассмотрели лишь основы работы со сменными носителями на примере их патриархов — компакт-дисков и дискет.

Понимание файла `/etc/fstab`

Возникает вопрос: существует ли простой способ работать с монтируемыми устройствами автоматически, поскольку вся гибкость, которой обладают средства монтирования, становится только помехой при интенсивном использовании системы. После того как вы разберетесь, какие команды монтируют второй жесткий диск IDE,

сетевой диск NFS, дискету MS-DOS и SCSI-привод CD-ROM, нужно ли постоянно помнить эти команды? Нет! Существует более удобное средство — файл `/etc/fstab`.

Его содержимое можно посмотреть с помощью команды `cat /etc/fstab`:

```
# Device      Mountpoint FStype  Options  D    Pass#
/dev/ad0slb   none       swap    sw        0    0
/dev/ad0sla   /          ufs     rw        1    1
/dev/ad0slg   /home     ufs     rw        2    2
/dev/ad0slf   /var      ufs     rw        2    2
/dev/ad0sle   /uer      ufs     rw        2    2
/dev/acd0c    /cdrom    cd9660  ro,noauto 0    0
/dev/fd0      /floppy   msdos   rw,noauto 0    0
proc         /proc     procfs  rw        0    0
```

В этом файле содержится все, что системе нужно знать об определенной точке монтирования: какое устройство прикреплено к ней, какой тип файловой системы используется на нем, опции монтирования, а также в каком порядке следует производить проверку файловых систем при загрузке системы. *Файл `fstab`* тесно связан с командой `mount`. В комплексе два этих средства значительно облегчают управление файловой системой.

Основная функция `fstab` — задать системе профиль монтируемых устройств, которые следует активизировать на этапе загрузки. Когда в файле заданы все точки монтирования, их можно одновременно подключить одной командой: `mount -a`. Именно так происходит во время загрузки после того, как система выполняет проверку файловых систем командой `fsck -p`, т.е. в режиме ргееп (чистка), проверяя, что все файловые системы имеют статус `clean` (чистый). Затем выполняется команда `mount -a -t nonfs` — монтирование всех файловых систем, перечисленных в файле `/etc/fstab`, кроме NFS.

Мало того, такие установки упрощают работу с файловыми системами. Если точка монтирования указана в файле `/etc/fstab`, не нужно в точности запоминать формат ко-

```
# mount /ha
```

Команда читает всю необходимую информацию из файла `fstab`, где указано, что на устройстве `/dev/ad0slg` установлена файловая система UFS (точнее, FFS), которую необходимо смонтировать для чтения-записи. Аналогично, для монтирования флоппи-диска достаточно ввести команду:

```
# mount /floppy
```

Согласитесь, вполне дружественный подход к пользователю!

Опция `noauto`, указанная для записей `/cdrom` и `/floppy` сообщает команде `mount`, что эти файловые системы не монтируются на этапе загрузки. Как и с ресурсами NFS, нельзя поручиться, что диск CD-ROM или флоппи-диск будет доступен во время загрузки системы, поэтому опция `noauto` предотвращает потерю времени на бесполезные попытки монтирования. Однако ничего не мешает смонтировать нужную файловую систему после загрузки, — пример такой команды приведен выше.

В четвертом столбце файла `fstab` задаются любые опции команды `mount`, применимые к файловой системе. Можно использовать, например, любую из опций, указанных на страницах справочного руководства `man mount` или `man mount_*`, если это нестандартная файловая система.

Самый правый столбец файла `/etc/fstab` — поле "Pass#"; оно содержит флажок, необходимый `fsck` (об этой утилите рассказано далее). Числа, большие нуля, задают порядок проверки файловых систем. Корневой файловой системе присвоен номер

прохода, равный 1, т.е. она проверяется первой; затем проверяются смонтированные файловые системы, которым присвоен номер 2 (если позволяет оборудование, проверка выполняется параллельно). Число ноль отключает проверку файловой системы. Именно это необходимо для компакт-дисков, дискет, разделов подкачки и других ресурсов, которые или не могут быть повреждены в принципе, или их состояние не имеет значения для системы.

Левее колонки, указывающей номер прохода (pass #), находится колонка с номером уровня дампа (dump level number). Он используется командой **dump**. Это старая утилита UNIX, предназначенная для резервного копирования на основе уровней. Номер уровня дампа сообщает утилите **dump**, на каком уровне необходимо запустить резервное копирование файловой системы. Например, файловые системы, для которых он равен 1, копируются лишь тогда, когда уровень дампа равен 1 или меньше (нулевой уровень дампа означает: "полное резервное копирование всех файловых систем"). Обратите внимание, что нулевой номер уровня дампа в /etc/fstab полностью предотвращает резервное копирование файловой системы.

• Полное обсуждение процедур резервного копирования и восстановления содержится в главе 20. Там же подробно рассказано об утилите **dump**, а также других средствах резервного копирования и зеркалирования, например, CVSup.

Проверка и восстановление файловых систем с помощью утилиты fsck

Программа fsck (сокращение от File System Consistency check — проверка согласованности файловой системы) похожа на Microsoft ScanDisk и другие дисковые утилиты, по крайней мере, по своей интерактивной природе и роли в процессе загрузки. Ее основная задача заключается в проверке файловых систем из /etc/fstab и их пригодности для монтирования. Такой режим называется green (чистка) и задается опцией -p. Кроме того, он устраняет все несогласованности в файловых системах, не отмеченных как clean (чистый) корректным методом останова системы.

Вероятно, вам придется сталкиваться с работой программы fsck только при загрузке системы. Обычно она запускается после идентификации всех устройств, выдавая на экран что-то вроде:

```
/dev/ad0s1e:
103469 files, 858450 used, 9066025 free
(25777 frags, 1130031 blocks, 0.3% fragmentation)
```

^ П Р И М Е Ч А Н И Е

Не беспокойтесь о фрагментации, о которой сообщает fsck. Помните, что даже фрагментация около 2-3% (вряд ли вам доведется увидеть большее значение) намного меньше той, что присуща традиционным настольным операционным системам. Для диска с системой DOS/VFAT нередко приходится наблюдать значение 50% и более. Поэтому утилиты дефрагментации в большом почете на рынке настольных систем. В файловых системах UNIX разработаны механизмы, которые автоматически размещают данные в соседних секторах, поэтому фрагментация всегда находится на низком уровне. Жесткий диск с файловой системой UNIX не придется дефрагментировать никогда (я бы не был столь категоричен — Прим. ред). Обратитесь к разделу "Блоки, файлы и индексные дескрипторы" далее в этой главе, чтобы разобраться с механизмом хранения данных и фрагментации.

Проблемы обычно возникают тогда, когда работа системы не была завершена нормально — при сбое питания или при некорректном выключении системы. Файловые системы UNIX следят за своей структурой посредством синхронной записи на диск метаданных, которая требует нескольких циклов записи. Если несвоевременный останов происходит в процессе такой записи, метаданные могут быть повреждены. В этом случае использование файловой системы будет невозможно до ее корректировки. Для этих целей и предназначена fsck.

Если файловая система находится в поврежденном (unclean) *состоянии, утилита fsck* работает в режиме исследования (investigative mode). Программа последовательно, блок за блоком "проходит" по файловой системе, проверяя целостность метаданных. Эта процедура может занимать длительное время и зависит от размера и скорости диска. Если утилита находит несоответствие, которое нельзя устранить автоматически, гарантируя при этом целостность данных (подробнее см. man fsck), fsck запрашивает подтверждение пользователя. В большинстве случаев *следует такое подтверждение дать*. Но, если программа запросила подтверждение, то несоответствие, скорее всего, столь значительно, что часть данных будет потеряна. Обычно, это файл или файлы, запись которых происходила в момент аварии системы. Как правило, потери данных при этом невелики.

По окончании работы fsck система может выдать приглашение #. В этом случае необходимо продолжить загрузку командой boot или перезагрузить машину командой reboot. *Последний вариант предпочтительнее, поскольку позволяет убедиться, что сервер* приходит в рабочее состояние без каких-либо проблем при загрузке.

Утилита fsck выполняется не только при загрузке, ее можно запустить из командной строки для любой смонтированной файловой системы. Однако делать этого все же не стоит, особенно если система работает и загружена различными задачами! Важно, чтобы проверяемая на согласованность файловая система не изменялась. Если все же необходимо запустить fsck, следует *перейти в* однопользовательский режим (single-user mode):

```
* shutdown +5
```

Через пять минут после запуска команды многопользовательский режим (multiuser mode) будет отключен. Естественно, что, начиная с этого момента, все действия можно будет выполнить только с консоли. В однопользовательском режиме удаленно администрировать систему нельзя.

После того как система приведена в "неподвижное" состояние, можно запустить fsck. Подобный подход может оказаться необходимым, если вы обнаружите, что вывод команды dmesg (он ежедневно высылается пользователю root) содержит сообщение о некорректном индексном дескрипторе, и захотите разобраться в причинах проблемы, не перегружая систему. После запуска fsck для одного или всех устройств (используется синтаксис вида fsck -p /dev/ad1slg) нужно просто выйти из однопользовательского командного интерпретатора (командой exit) и вернуться к обычному (многопользовательскому) состоянию системы.

Однако иногда команду fsck можно вполне безопасно запускать и в многопользовательском режиме, например, при монтировании второго диска с несущественными данными или новой файловой системой. Запуск fsck на этапе загрузки проверяет только файловые системы, перечисленные в файле /etc/fstab. Вместо добавления нового устройства в файл fstab и перезагрузки можно просто попытаться смонтировать устройство. Если попытка завершится неудачей, следует запустить команду fsck (с синтаксисом, приведенным выше). А затем вновь попытаться смонтировать диск.

Все это можно смело проделать и в многопользовательском режиме, поскольку никто не будет записывать данные на устройство, которое еще не смонтировано!

Журнальные файловые системы и мягкое обновление

Для решения проблемы синхронной записи созданы различные решения. Одно из них — это журнальные (logging) файловые системы, в которых все метаданные до непосредственной записи на диск заносятся в журнал. Это заметно ускоряет работу утилиты fsck, поскольку ей не требуется анализировать всю файловую систему — местоположение несогласованностей известно заранее, поэтому их остается лишь исправить.

Однако FreeBSD не поддерживает журнальных файловых систем. В ней используется другое решение проблемы — мягкое обновление (Soft Updates). В то время как журнальные файловые системы заносят в log-файл все операции записи, мягкое обновление (начиная с версии FreeBSD 5.0 оно встроено в стандартное ядро) — это потенциально более развитая методика, при которой выполняются упорядоченные операции записи, не требующие применения внешнего log-файла. Мягкое обновление обеспечивает целостность метаданных и согласованность файловой системы точно так же или даже лучше, чем журнальные файловые системы. Кроме того, оно имеет преимущество в производительности: файловая система становится доступной сразу на этапе загрузки, а проверка согласованности происходит автоматически в фоновом режиме.

Поддержка мягкого обновления во FreeBSD 5.0 осуществляется демоном diskcheckd. Этот демон (включен по умолчанию) запускается в фоновом режиме и производит периодическую проверку целостности файловой системы, устраняя необходимость в запуске fsck при загрузке и уменьшая риски, связанные с аварийным остановом системы. Конфигурационным файлом этого демона является /etc/diskcheckd.conf; инструкции и примеры его использования приведены в справочном руководстве, вызываемом командой man diskcheckd. Все ошибки, найденные diskcheckd, записываются службой syslogd (о ней рассказано в главе 11). Во время работы демона diskcheckd его состояние можно просмотреть командой ps:

```
# ps -ax | grep diskcheckd
251  ??  Ss      0:00.28 diskcheckd: ad0 13.26% (diskcheckd)
```

Утилита **fsck** в FreeBSD очень похожа на **feck** других операционных систем за тем исключением, что ей не хватает нескольких удобных элементов интерфейса, например, строки состояния, которая есть у **fsck** в Linux. Ключевые свойства всех утилит, конечно же, одинаковы.

Более подробную информацию о мягком обновлении и сравнении его с журнальными файловыми системами можно найти по адресу <http://vwww.mckusick.com/soft^p/> и <http://vnvw.ece.cmu.edu/~ganger/papers/CSE-TR-254-95/>.

Использование fsck для восстановления суперблоков

Скорее всего, изложенное ниже вам не понадобится, однако, как говорит закон Мэрфи, лучший способ убедиться, что меры предосторожности не понадобятся, это своевременно принять их.

Распространенным типом сбоя файловой системы является повреждение суперблока: когда система по каким-либо причинам не может прочесть информацию из блока, содержащего важнейшие данные о файловой системе. Этот блок называется **суперблоком** и размещается в секторах с 16 по 31 от начала устройства. Он является настолько жизненно важным для файловой системы, что FreeBSD содержит альтернативные суперблоки в начале каждой группы цилиндров. Если основной суперблок будет поврежден, на устройстве можно будет отыскать множество его резервных копий. Первая из них всегда начинается с сектора 32, остальные расположены на диске с определенным интервалом.

Предположим, что необходимо смонтировать файловую систему (например, сменный жесткий диск), который при последнем использовании был работоспособен. Однако при запуске команда mount выдает сообщение об ошибке: "/dev/ad1slh on /mnt: Incorrect super block". Положение легко исправить с помощью fsck.

```
# fsck /dev/ad1elh
** /dev/ad1slh
BAD SUPER BLOCK: MAGIC NUMBER WRONG
LOOK FOR ALTERNATE SUPERBLOCKS? [yn] y
USING ALTERNATE SUPERBLOCK AT 32
** Last Mounted on /home2
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Cyl groups
148 files, 15660 used, 7038840 free (208 frags, 879829 blocks, 0.0%
fragmentation)
UPDATE STANDARD SUPERBLOCK? [yn] y
***** FILE SYSTEM WAS MODIFIED *****
```

Во *FreeBSD* **все выглядит просто**: на других платформах необходимо **использовать** опции командной строки, указывающие расположение резервного суперблока, причем, фиксированное местоположение имеет только суперблок на секторе 32. Утилита fsck копирует первый резервный суперблок на место основного и после этого файловая система становится пригодной к монтированию.

Определить, где именно на устройстве расположены все суперблоки, позволяет утилита newfs с параметром -N (при этом выводятся данные о файловой системе и никаких изменений не вносится):

```
# newfs -N /dev/ad1slh
Warning: 2672 sector(s) in last cylinder unallocated
/dev/ad1slh: 14550608 sectors in 3555 cylinders of 1 tracks, 4096 sectors
7108.7MB in 223 cyl groups (16 c/g, 32.00MB/g, 7936 i/g)
super-block backups (for fsck -b «) at:
32, 65568, 131104, 196640, 262176, 327712, 393248, 458784, 524320, 589856,
655392, 720928, 786464, 852000, 917536, 983072, 1048608, 1114144, 1179680,
1245216, 1310752, 1376288, 1441824, 1507360, 1572896, 1638432, 1703968,
1769504, 1835040, 1900576, 1966112, 2031648, 2097184, 2162720, 2228256,
2293792, 2359328, 2424864, 2490400, 2555936, 2621472, 2687008, 2752544,
2818080, 2883616, 2949152, 3014688, 3080224, 3145760, 3211296, 3276832,
3342368, 3407904, 3473440, 3538976, 3604512, 3670048, 3735584, 3801120,
3866656, 3932192, 3997728, 4063264, 4128800, 4194336, 4259872, 4325408,
4390944, 4456480, 4522016, 4587552, 4653088, 4718624, 4784160, 4849696,
4915232, 4980768, 5046304, 5111840, 5177376, 5242912, 5308448, 5373984,
5439520, 5505056, 5570592, 5636128, 5701664, 5767200, 5832736, 5898272,
```

```

5963808, 6029344, 6094880, 6160416, 6225952, 6291488, 6357024, 6422560,
6488096, 6553632, 6619168, 6684704, 6750240, 6815776, 6881312, 6946848,
7012384, 7077920, 7143456, 7208992, 7274528, 7340064, 7405600, 7471136,
7536672, 7602208, 7667744, 7733280, 7798816, 7864352, 7929888, 7995424,
8060960, 8126496, 8192032, 8257568, 8323104, 8388640, 8454176, 8519712,
8585248, 8650784, 8716320, 8781856, 8847392, 8912928, 8978464, 9044000,
9109536, 9175072, 9240608, 9306144, 9371680, 9437216, 9502752, 9568288,
9633824, 9699360, 9764896, 9830432, 9895968, 9961504, 10027040, 10092576,
10158112, 10223648, 10289184, 10354720, 10420256, 10485792, 10551328,
10616864, 10682400, 10747936, 10813472, 10879008, 10944544, 11010080,
11075616, 11141152, 11206688, 11272224, 11337760, 11403296, 11468832,
11534368, 11599904, 11665440, 11730976, 11796512, 11862048, 11927584,
11993120, 12058656, 12124192, 12189728, 12255264, 12320800, 12386336,
12451872, 12517408, 12582944, 12648480, 12714016, 12779552, 12845088,
12910624, 12976160, 13041696, 13107232, 13172768, 13238304, 13303840,
13369376, 13434912, 13500448, 13565984, 13631520, 13697056, 13762592,
13828128, 13893664, 13959200, 14024736, 14090272, 14155808, 14221344,
14286880, 14352416, 14417952, 14483488, 14549024

```

Как можно видеть, существует множество резервных копий. Вы также можете указать определенный суперблок (если ие хотите, чтобы он выбирался автоматически), например, команда **feck -Б 2490400 /dev/ad1s1h** использует суперблок из сектора 2490400.

Настройка пользовательских квот

Помните пользователей, которые беспрерывно зафужали файлы, что вынудило вас купить новый жесткий диск и перенести туда их каталоги? Предположим, что вы хотите запретить любому пользователю превышать лимит в 20 Мб без вашего специального разрешения. Сделать это позволяют *квоты* (quotas).

Поддержка квот не встроена в стандартное ядро. Чтобы разрешить их использование, в конфигурацию ядра необходимо добавить следующую строку:

```
options          QUOTA
```

О компиляции ядра подробно рассказано в главе 17. Кроме того, несколько флажков поддержки квот содержатся в файле `/etc/rc.conf`. В него необходимо добавить такие строки:

```
enable_quotas="YES"
check_quotas="NO"
```

Первая строка включает глобальную поддержку квот, вторая заставляет систему во время загрузки пропустить довольно длительную проверку согласованности (quotacheck -a), которая удостоверяет, что базы данных квот синхронизованы нужным образом. Чтобы разрешить эту проверку, установите значение YES (или удалите строку YES, которая используется по умолчанию).

Последний шаг заключается во включении (или выключении) квот в файловых системах. Для этого в четвертом поле файла `/etc/fstab` добавляется опция `userquota` и/или `groupquota`. Например:

```

/dev/adOslg    /home      ufs  rw,userquota,groupquota 2    2
/dev/adOslf    /var       ufs  rw,userquota              2    2
/dev/adOsle    /usr       ufs  rw,gropquota              2    2

```

После выполнения всех настроек следует перезагрузить компьютер. Теперь все готово к назначению квот пользователям. Это можно сделать для каждого пользователя отдельно (на сильно загруженном сервере это затруднительно) или для диапазона

UID (установив квоту для одного пользователя и применив ее как прототип для определенного диапазона UID). Чтобы установить квоту для первого пользователя, необходимо собрать утилиту `edquota`, которая позволяет редактировать атрибуты в текстовом файле (наподобие программы `chfn`, о которой рассказано в следующей главе). Текстовый редактор, который использует `edquota`, указан в переменной среды `EDITOR`, по умолчанию это `vi`, но его можно заменить на более дружелюбный к пользователю, например, `pico` или `ee`, командой `setenv EDITOR pico`.

```
# edquota -u bob
Quotas for user test:
/usr: blocks in use: 65, limits (soft = 50, hard = 75)
      inodes in use: 7, limits (soft = 50, hard = 60)
/var: blocks in use: 0, limits (soft = 50, hard = 75)
      inodes in use: 0, limits (soft = 50, hard = 60)
```

После установления квоты для одного пользователя ее можно распространить и на целый ряд других:

```
# edquota -p bob 1001-9999
```

Эта команда применяет указанные выше настройки квот к заданному диапазону UID (который может включать и идентификаторы еще не созданных пользователей).

Обратите внимание на разницу между блоками и индексными дескрипторами. Используются пределы для обоих значений. *Блоки* относятся к общему объему, а индексные дескрипторы понимаются в значении "файлы".

Блоки, файлы и индексные дескрипторы

Существуют различные способы размещения данных на диске. Их необходимо знать, чтобы понимать вывод таких команд, как `df` и `fsck`. Здесь рассказано о том, как хранятся данные.

Размер блока данных по умолчанию равен 8192 байта. Это единица хранения данных. Блок разделен на восемь фрагментов по 1024 байта. Файл, который занимает не полный блок, сохраняется фрагментарно вместе с другими файлами. Именно это подсчитывает утилита `fsck`, когда выводит информацию о фрагментации.

Когда файл, находящийся в одном блоке с другим файлом, увеличивается так, что ему уже не хватает места в этом блоке, FreeBSD перемещает весь файл в другой блок данных вместо того, чтобы просто разместить фрагмент файла в следующем блоке (как это делает система Windows). Таким образом, все файлы, меньшие 8192 байт, всегда сохраняются в отдельных блоках. Кроме того, файл произвольного размера занимает наименьшее возможное число блоков и имеет фрагменты только в одном блоке данных. Такой подход уменьшает время доступа и предотвращает высокую степень фрагментации, присутствующую другим системам.

Фактически, файл — это индексный дескриптор. Последний является фундаментальной группой связанных данных, которые мы и рассматриваем как файл. Такая модель требуется настоящей многопользовательской операционной системе для эффективного разделенного доступа к файлам. В индексном дескрипторе содержится следующая информация:

- Тип файла и режимы доступа к нему
- UID и GID владельца файла

Размер файла

Время последнего доступа и изменения файла, а также номер модифицированного индексного дескриптора

Число блоков данных, выделенных файлу

- Прямые и не прямые указатели на эти блоки данных

Доступ к блокам данных во FreeBSD осуществляется по указателям, хранимым в индексных дескрипторах, а именно: существует 12 прямых указателей, каждый из которых указывает на отдельный блок. Таким образом, к файлу размером до 96 килобайт возможен прямой доступ. Кроме того, есть три уровня не прямых указателей: одинарный, двойной и тройной. *Одинарный* не прямой указатель содержит ссылку на блок файловой системы, содержащий указатели на блоки данных. Такой блок файловой системы содержит 2048 дополнительных адресов 8-килобайтных блоков, что дает доступ к файлу размером до 16 мегабайт. *Двойной* не прямой указатель ссылается на блок файловой системы, содержащий 2048 блоков, содержащих одинарные не прямые указатели, каждый из которых ссылается на блок файловой системы с 2048 адресами 8-килобайтных блоков. Это дает доступ к файлу размером до 32 гигабайт. И наконец, на вершине иерархии находится *тройной* не прямой указатель, который делает возможным доступ к файлу размером до 70 терабайт! Следует отметить, что в UFS максимальный размер файла все-таки ограничен и равен одному терабайту.

Теперь несколько слов о жестких и мягких ограничениях и периоде отсрочки.

- *Жесткое ограничение (hard limit)* выполняется неукоснительно — если пользователь достиг установленного предела дискового пространства, системе запрещено выделять ему больше.
- *Мягкое ограничение (soft limit)* не запрещает пользователю создавать больше файлов или использовать дополнительное дисковое пространство. Но после превышения установленного лимита начинает действовать *период отсрочки (graceperiod)*, равный по умолчанию семи дням (эту установку можно изменить опцией `edquota -t`). По окончании периода отсрочки мягкие ограничения превращаются в жесткие. Такой подход позволяет пользователям на короткий период времени использовать больше дискового пространства, чем им выделено. Период отсрочки обнуляется, когда объем дискового пространства, занимаемого пользователем, становится ниже предела.

По установлению ограничений их можно просмотреть следующим образом:

```
• quota bob
Disk quotas for user bob (uid 1015):
Filesystem blocks quota limit grace files quota limit grace
/home      1812   20000 40000      37     0     0
```

Команда `quota` показывает информацию о квотах для заданного или текущего (если аргумент опущен) пользователя. Если пользователь выходит за установленные пределы, это отмечается звездочкой (*), а колонка с периодом отсрочки (`grace`) содержит срок до его окончания:

```
• quota bob
Disk quotas for user bob (uid 1015):
Filesystem blocks quota limit grace files quota limit grace
/home      28121* 20000 40000 6days 189     0     0
```

О ПРИМЕЧАНИЕ

Чтобы убедиться, что квоты установлены правильно, используется команда `mount` без аргументов. Вот вывод этой команды в системе, где квоты установлены для раздела `/home`:

```
/dev/ad0s1a on / (ufs, local)
/dev/ad0s1f on /home (ufs, local, with quotas)
/dev/ad0s1e on /usr (ufs, local)
procfs on /proc (procfs, local)
```

Если флажок `with quotas` отсутствует, это значит, что файловая система была смонтирована без квот. Проверьте файлы `/etc/fstab`, `/etc/rc.conf` и конфигурацию ядра. Если все выглядит верно, попробуйте перезагрузить систему.

Отключить квоты можно одним из трех способов: глобально, установив опцию `enable_quotas="NO"` в файле `/etc/rc.conf`; для каждой файловой системы в файле `/etc/fstab` или для определенного пользователя посредством команды `edquota`, установив мягкие и жесткие ограничения нулями. При желании, эти настройки можно применить к диапазону UFD с помощью команды `edquota -p`.

Управление файловой системой — это непростая задача, но, если вы разобрались с концепциями, изложенными здесь, вам сразу же станет понятны преимущества файловых систем UNIX. В многопользовательских операционных системах, каковой является и FreeBSD, имеются возможности, отсутствующие у настольных систем, например, работа с несколькими типами операционных систем, мониторинг использования дискового пространства, применение квот. Другие особенности работы с файловыми системами рассматривается в главе 19, где мы обсудим форматирование и разметку новых жестких дисков.