ОС Linux. Руководство системного администратора

====================================================================

Ларс Виржениус (Lars Wirzenius)

- 2 -

Содержание

Глава 1 Введение 4

1.1 Проект Документирования Системы Linux 6

Глава 2 Обзор Системы Linux 8

2.1 Различные составляющие операционной системы 8

2.2 Важные составляющие ядра 9

2.3 Основные функции UNIX системы 10

2.3.1 init 10

2.3.2 Подключение к системе с терминалов 11

2.3.3 Syslog 11

2.3.4 Периодическое выполнение команд: cron и at 11

2.3.5 Графический интерфейс пользователя 12

2.3.6 Работа с сетью 12

2.3.7 Подключение к системе через сеть 13

2.3.8 Сетевые файловые системы 13

2.3.9 Почта 14

2.3.10 Печать 14

2.4 Структура файловой системы 15

Глава 3 Запуск Системы и Перезагрузка 16

3.1 Обзор 16

3.2 Процесс запуска при близком рассмотрении 17

3.3 Завершение работы и выключение системы 20

3.4 Перезагрузка системы 23

3.5 Однопользовательский режим работы 23

3.6 Дискеты для экстенной загрузки 23

Глава 4 Использование Дисков и Других Устройств 25

4.1 Типы устройств 26

4.2 Жесткие диски 27

4.3 Гибкие диски 30

4.4 Форматирование 31

4.5 Дисковые разделы 34

4.5.1 MBR, загрузочные сектора и таблица разделов 35

4.5.2 Расширенные и логические разделы 35

4.5.3 Типы разделов 37

4.5.4 Разделение жесткого диска 38

4.5.5 Файлы устройств и разделы 39

4.6 Файловые системы 39

4.6.1 Что такое файловая система? 39

4.6.2 Типы файловых систем 41

- 3 -

4.6.3 Какую файловую систему устанавливать? 44

4.6.4 Установка файловой системы 44

4.6.5 Монтирование и демонтирование 46

4.6.6 Поддержка работоспособности файловых систем 50

4.7 Диски без файловых систем 52

4.8 Распределение дискового пространства 53

4.8.1 Схемы разделения дисков 53

4.8.2 Требования к дисковому пространству 54

4.8.3 Примеры распределения жесткого диска 55

4.8.4 Использование дополнительного дискового пространства 55

4.8.5 Методы сохранения дискового пространства 55

Глава 5 Обзор Структуры Каталогов 57

5.1 Введение 57

5.2 Файловая система root 59

5.2.1 Каталог /etc 60

5.2.2 Каталог /dev 62

5.3 Файловая система /usr 63

5.4 Файловая система /var 64

5.5 Файловая система /proc 65

Глава 6 Упpавление Памятью 68

6.1 Что такое виpтуальная память? 68

6.2 Создание swap области 69

6.3 Использование swap пpостpанства 70

6.4 Разделение swap областей с дpугими опеpационными 71

6.5 Размещение swap пpостpанства 72

6.6 Дисковый буфеp 73

Глава 7 Подключение и Выход из Системы 76

7.1 Подключение к системе чеpез теpминалы 76

7.2 Подключение к системе чеpез сеть 77

7.3 Что выполняет пpогpамма login 78

7.4 X и xdm 79

7.5 Контpоль доступа 79

7.6 Запуск оболочки 80

.

- 4 -

Глава 1 Введение

В этой книге рассматриваются аспекты системного

администрирования операционной системы Linux. В первую очередь

данное руководство предназначено для тех, кто практически ничего

не знает о системном администрировании, но ознакомлен с основами

работы с системой Linux, которые рассмотрены в книге "Руководство

Пользователя Системы Linux". В данном руководстве не уделяется

внимания проблемам, связанным с установкой системы (исчерпывающую

информацию можно получить из книги "Установка и Запуск Системы

Linux"). Во многих руководствах из серии LDP (Linux Documentation

Project) рассматриваются одни и те же вопросы с разных точек

зpения. Ниже содержится более подробная информация о руководствах

по системе Linux.

Что же называется системным администрированием? Это все то,

что требуется для поддержки работоспособности компьютерной системы

(например, создание резервных копий некоторых файлов, установка

новых программ, создание и удаление пользователей, проверка

целостности файловой системы и т.д.). Если сравнивать компьютер с

домом, то системное администрирование можно назвать содержанием

этого дома, включающее в себя уборку, устранение различных

неисправностей и т.д. Но системное администрирование не называют

содержанием, так как тогда это было бы слишком просто.

Руководство составлено таким образом, что главы не зависят

друг от друга и могут быть рассмотрены отдельно. Например, для

получения информации по созданию резервных копий Вы можете

прочитать только одну главу. Это очень удобно и позволяет

использовать книгу в качестве справочного пособия, а также

избежать чтения всего руководства вместо нескольких разделов.

Однако, прежде всего это книга, а затем уже справочник.

Конечно, в этой книге не содержится всей необходимой

информации по системному администрированию, хотя много полезного

можно найти и в другой документации по системе Linux. В конце

- 5 -

концов, системный администратор это просто пользователь с

привелигированными правами доступа и определенными обязанностями.

Также много полезной и важной информации можно узнать из

встроенных описаний к программам (при помощи команды 'man').

В то время как эта книга рассчитана на использование

операционной системы Linux, она может быть расмотрена и по

отношению к другим UNIX-подобным операционным системам. Так как

разные версии системы UNIX сильно различаются (особенно в вопросах

системного администрирования), то довольно сложно охватить весь

материал, касающийся каждой модификации. Даже рассмотрение всех

особенностей ОС Linux - сложная задача, так как она сейчас

находится в стадии развития. На данный момент не существует

официальной версии Linux, поэтому у разных людей она установлена

по разному. В книге указаны различия между такими системами и

рассмотрены некоторые альтернативы, где это возможно. Также в

книге содержится много информации, которая не каждому может

потребоваться. Такие моменты специально отмечены и могут быть

пропущены, если используется уже сконфигурированная система.

Также как и другие разработки, связанные с операционной

системой Linux, книга была написана на добровольных началах.

Однако, как и в любой добровольной работе, существует предел

предпринимаемым усилиям, а также знаниям и опыту автора. Это

означает, что данное руководство не обязательно также хорошо

написано, как если бы оно было написано на коммерческой основе.

Следует отметить, что в этой книге не полностью охвачены

многие проблемы, которые рассмотрены в других свободно

распространяемых руководствах и документациях. Особенно это

относится к описаниям к различным программам, например, к

особенностям использования mkfs(8). В данном руководстве описано

только назначение этой программы в объеме, необходимом для

использования в книге. Для получения более подробной информации

имеются ссылки к другим руководствам и документациям, которые

обычно являются частью полного набора документации по системе

Linux.

- 6 -

У автора есть стремление развивать и улучшать эту разработку.

Просьба направлять все замечания (ошибки, новые идеи, информация о

различиях между различными версиями системы UNIX и т.д.) либо по

адресу электронной почты lars.wirzenius@helsinki.fi, либо обычной

почтой по адресу:

Lars Wirzenius / Linux docs

Hernesaarentie 15 A 2

00150 Helsinki

Finland

Автор хочет поблагадорить следующих людей, оказавших

поддержку при создании этой книги: Matt Welsh, Andy Oram, Olaf

Kirch, Adam Richter и других.

H.Peter Anvin, Remy Card, Theodore Ts'o и Stephen Tweedie

позволили использовать часть их работ в книге. Stephen Tweedie

также позволил использовать его сравнение файловых систем xia и

ext2.

В дополнение к выше сказанному, автор благодарит Mark

Komarinski за публикацию в 1993 году его работ и других статей,

имеющих отношение к системному администрированию, в журнале Linux

Journal.

Автор выражает благодарность Erik Troan из Red Hat за

создание версии руководства в формате plain text, а также

следующим людям: Paul Caprioli, Ales Cepek, Marie-France

Declerfayt, Olaf Flebbe, Helmut Geyer, Larry Greenfield, Stephen

Harris, Jyrki Havia, Jim Haynes, York Lam, Timothy Andrew Lister,

Jim Lynch, Dan Poirier, Daniel Quinlam, Philippe Steindl.

1.1 Проект Документирования Системы Linux

Проект документирования системы Linux или LDP (Linux

Documentation Project), это свободная команда писателей,

корректоров и редакторов перед которыми стоит цель создания полной

- 7 -

документации по операционной системе Linux. Главным координатором

проекта является Matt Welsh, который был назначен Lars'ом

Wirzenius и Michael'ом K. Johnson.

Данное руководство является одним из набора распространяемых

руководств проекта LDP, который включает в себя "Руководство

Пользователя Системы Linux", "Руководство Системного

Администратора Системы Linux", "Руководство Сетевого

Администратора Системы Linux" и "Руководство Взломщика Ядра

Linux". Эти книги доступны в исходном формате LaTeX, .dvi формате

и в формате postscript через FTP:

ftp://sunsite.unc.edu/pub/Linux/docs/LDP

ftp://tsx-11.mit.edu/pub/linux/docs/guides

Для того, чтобы связаться с LDP, нужно послать письмо Matt'у

Welsh по указанному ниже адресу:

mdw@sunsite.unc.edu

.

- 8 -

Глава 2 Обзор Системы Linux

В этой главе дается обзор операционной системы Linux. Прежде

всего рассматриваются основные функции системы. Затем следует

описание программ, реализующих эти функции. Цель главы - дать

понимание системы в целом, поэтому каждый раздел рассматривается

более подробно позже.

2.1 Различные составляющие операционной системы

Любая UNIX-подобная операционная система состоит из ядра и

некоторых системных программ. Также существуют некоторые

прикладные программы для выполнения какой-либо задачи. Ядро

является сердцем операционной системы. Оно размещает файлы на

диске, запускает программы и переключает процессор и другое

оборудование между ними для обеспечения мультизадачности,

распределяет память и другие ресурсы между процессами,

обеспечивает обмен пакетами в сети и т.п. Ядро само по себе

выполняет только маленькую часть общей работы, но оно

предоставляет средства, обеспечивающие выполнение основных

функций. Оно также предотвращает использование прямого доступа к

аппаратным средствам предоставляя специальные средства для

обращения к периферии. Таким образом ядро позволяет контролировать

использовние аппаратных средств различными процессами и

обеспечивать некоторую защиту пользователей друг от друга.

Средства, предоставляемые ядром, используются через системные

вызовы (см. раздел 2 руководства для более подробной информации по

этому вопросу).

Системные программы используют средства, предоставляемые

ядром для обеспечения выполнения различных функций операционной

системы. Системные и все остальные программы выполняются 'на

поверхности ядра', в так называемом пользовательском режиме.

Существует некотоpая разница между системными и прикладными

программами. Прикладные программы предназначены для выполнения

какой-либо определенной задачи, в то время как системные программы

- 9 -

используются для поддержания работы системы. Текстовый процессор

является прикладной программой, а программа telnet - системной,

хотя зачастую граница между ними довольно смутная.

Довольно часто операционная система содержит компиляторы и

соответствующие им библиотеки (GCC и C библиотеки для Linux), хотя

не обязательно все языки программирования должны быть частью

операционной системы. Документация, а иногда даже игры, могут

являться ее частью. Обычно состав операционной системы

определяется содержимым установочного диска или ленты, хотя дело

обстоит несколько сложнее, так как различные части операционной

системы разбросаны по разным FTP серверам во всем мире.

2.2 Важные составляющие ядра

Ядро системы Linux состоит из нескольких основных частей:

блок управления процессами, блок управления памятью, драйверы

устройств, драйверы файловых систем, блок управления сетью а также

другие небольшие процедуры.

Наиболее важные составляющие ядра (обеспечивающие

жизнеспособность системы) - это блок управления памятью и

процессами. Блок управления памятью обеспечивает распределение

областей памяти и swap-областей между процессами, составляющими

ядра и для кэш-буфера. Блок управления процессами создает новые

процессы и обеспечивает многозадачность путем переключения задач.

На самом нижнем уровне ядро содержит драйверы устройств для

каждого типа поддерживаемого оборудования. Существует довольно

большой набор различных драйверов, так как постоянно

разрабатываются новые типы устройств. Существует довольно много

одинаковых устройств, которые различаются только тем, как

происходит взаимодействие между самим устройством и драйвером.

Такое сходство позволяет использовать классы драйверов,

поддерживающих одинаковые операции. В каждом члене такого класса

используется однотипный интерфейс для ядра, но различные схемы

взаимодействия с устройством. Например, все драйверы жесткого

диска представляются для ядра абсолютно одинаково, то есть у них у

- 10 -

всех имеются такие операции как 'инициализация жесткого диска',

'чтение сектора N', 'запись сектора N'.

Некоторые функции, предоставляемые ядром, имеют одинаковые

свойства. Например, различные сетевые протоколы объединены в один

программный интерфейс - BSD socket библиотеку. Вот другой пример -

различные файловые системы, поддерживаемые системой Linux. Ядро

содержит виртуальную файловую систему (Virtual File System - VFS)

которая включает в себя все функции, используемые для работы

системы, а также драйвер для каждой поддерживаемой файловой

системы. При попытке доступа к какой-либо файловой системе запрос

проходит через VFS, откуда перенаправляется к соответствующему

драйверу файловой системы.

2.3 Основные функции UNIX системы

В этом разделе достаточно поверхностно рассматриваются

некоторые наиболее важные функции UNIX системы. Более подробно они

рассмотрены в следующих главах.

2.3.1 init

Единственную и самую важную функцую в UNIX системе

предоставляет пpоцесс init. Он запускается в любой UNIX системе

как самый первый процесс, а также завершает процедуpу загрузки

системы. При запуске init, продолжается процесс загрузки

(проверяются и устанавливаются файловые системы, запускаются

различные программы-демоны и т.д.).

Точный список того, что выполняется при запуске init, зависит

от версии программы. Обычно init предоставляет

однопользовательский режим, при котором никто не может

подключиться к системе. Обычный режим - это многопользовательский.

Некоторые версии используют понятие 'уровень запуска'. Например,

однопользовательский и многопользовательский режимы - это разные

уровни запуска. Также существуют дополнительные уровни, например

для запуска X-windows.

- 11 -

При работающей системе, две самые важные задачи программы

init - это удостоверится, что все программы-демоны getty работают

(т.е. имеется возможность подключения к системе) и адаптирование

orphan-процессов (т.е. процессов, чей родительский процесс был

уничтожен; в системе UNIX все процессы должны принадлежать одному

дереву процессов, поэтому orphan-процессы должны быть

адаптированы).

При завершении работы системы и перезапуске, init уничтожает

все оставшиеся процессы, демонтирует файловые системы и

останавливает процессор.

2.3.2 Подключение к системе с терминалов

Подключение к системе с терминалов (через последовательные

линии) и с главной консоли (если не запущены X) обеспечивается

программой getty. init запускает отдельный процесс getty для

каждого терминала. getty считавает имя пользователя и запускает

программу login, которая считывает пароль. Если имя и пароль

соответствуют одному пользователю, то login запускает оболочку.

При выходе из оболочки, то есть при выходе из системы, или при

завершении программы login в случае если имя пользователя и пароль

не подходят, init запускает новый процесс getty. Ядро не

контролирует подключения к системе, а только выполняет системные

программы.

2.3.3 Syslog

Иногда при работе ядра или различных системных программ

возникают ошибки, предупреждения и другие сообщения. Программа

syslog записывает все сообщения в файл так, что он может быть

впоследствии просмотрен. Syslog можно сконфигурировать так, что

сообщения будут сортироваться и записываться в разные файлы по

приоритету. Например, сообщения ядра часто направляются в

отдельный файл, так как эти сообщения наиболее важные и должны

регулярно просматриваться во избежание серьезных проблем.

2.3.4 Периодическое выполнение команд: cron и at

- 12 -

Как отдельныем пользователям, так и системному администратору

иногда требуется периодически запускать определенные команды.

Например, системному администратору может потребоваться

периодически запускать команду для очистки каталогов от временных

файлов (/tmp и /var/tmp), для предотвращения переполнения диска.

Для этого используется функция cron. У каждого пользователя

имеется файл crontab, в котором содержится список команд,

представленных к выполнению и количество раз, которое они должны

быть выполнены. Программа-демон crond обеспечивает своевременное

выполнение указанных команд.

Функция at идентична функции cron, только команда запускается

один раз в указанное время и запуск больше не повторяется.

2.3.5 Графический интерфейс пользователя

Как в системе UNIX так и в Linux, пользовательский интерфейс

не встраивается в ядро системы. Вместо этого он представляется

программами пользовательского уровня. Это применяется как к

текстовым, так и к графическим оболочкам.

Такой стандарт делает систему более гибкой, хотя и имеет свои

недостатки. Например, это легко позволяет создавать новые

интерфейсы для программ, что затрудняет изучение системы.

Первоначально используемой с системой Linux графической

оболочкой была система X Window System (сокращенно X). X не

реализует пользовательский интерфейс, а только оконную систему,

т.е. средства, с помощью которых может быть реализован графический

интерфейс. Три наиболее популярных версии графических интерфейсов

на основе X - это Athena, Motif и Open Look.

2.3.6 Работа с сетью

Сеть - это средство, позволяющее соединяться двум или более

компьютерам между собой.

- 13 -

UNIX-подобные операционные системы имеют широкий спектр

сетевых возможностей. Большинство базовых функций (файловые

системы, печать, создание резервных копий и т.д.) могут быть

реализованы посредством сети. Это может значительно облегчить

работу системного администратора, так как позволяет использовать

централизованное администрирование.

Однако, в этой книге поверхностно рассмотрены вопросы работы

с сетью. Для более подробной информации см. книгу "Руководство

Сетевого Администратора Системы Linux".

2.3.7 Подключение к системе через сеть

Подключение к системе через сеть работает несколько иначе,

чем обычное подключение. Существуют отдельные физические

последовательные линии для каждого терминала, через которые и

происходит подключение. Для каждого пользователя, подключающегося

к системе, существует отдельное виртуальное сетевое соединение и

их может быть любое количество. Однако не представляется возможным

запустить отдельный процесс getty для каждого возможного

виртуального соединения. Существуют также и другие способы

подключения к системе посредством сети. Например, telnet и rlogin

- основные службы в TCP/IP сетях.

При использовании сети для подключения к системе, кроме

большого количества процессов getty используется отдельная

программа-демон (при использовании как telnet так и login

используются различные программы-демоны), которая отслеживает все

попытки соединения с компьютером. Если опpеделяется попытка

соединения, то программа создает новый процесс - создает сама себя

для обработки этого соединения - а затем продолжает отслеживание

новых соединений. Создаваемый процесс идентичен программе getty.

2.3.8 Сетевые файловые системы

Одна из наиболее полезных функций, которая может быть

реализована с помощью сети, это разделение файлов через сетевую

- 14 -

файловую систему. Обычно используется система, называемая Network

File System или NFS, которая разработана корпорацией Sun.

При работе с сетевой файловой системой любые операции над

файлами, производимыми на локальном комьютере, передаются через

сеть на удаленную машину. При работе сетевой файловой системы

программа считает, что все файлы на удаленном компьютере находятся

на компьютере, где она запущена. Таким образом разделение

информации посредством такой системы не требует внесения

каких-либо изменений в программу.

2.3.9 Почта

Электронная почта является самым важным средством связи между

компьютерами. Электронные письма хранятся в одном файле в

специальном формате. Для чтения и отправления писем применяются

специальные программы.

У каждого пользователя имеется отдельный "почтовый ящик"

(т.е. файл, где информация хранится в специальном формате), в

котром хранится приходящая почта. Если на комьютер приходит

письмо, то программа обработки почты находит файл почтового ящика

соответвующего пользователя и добавляет туда полученное письмо.

Если же почтовый ящик пользователя находится на другом компьютере,

то письмо перенаправляется на этот компьютер, где проходит его

последующая обработка.

Почтовая система состоит из множества различных программ.

Доставка писем к локальным или удаленным почтовым ящикам

производится одной программой (например, sendmail или smail), в то

время как для обычной отправки или просмотра писем применяется

большое количетсво различных программ (например, Pine или elm).

Файлы почтовых ящиков обычно хранятся в каталоге /var/spool/mail.

2.3.10 Печать

В один момент времени принтером может пользоваться только

один пользователь. Поэтому для того, чтобы принтером могли

- 15 -

пользоваться сразу несколько пользователей, используется

специальная программа, которая обеспечивает обслуживание очереди к

принтеру. Все задания для принтера помещаются в буфеp. Когда

принтер заканчивает обработку одного задания, следуещее передается

к нему автоматически. Это значительно упрощает работу с принтером.

Программа обслуживания очереди к принтеру помещает

информацию, которая должна быть распечатана, на диск, то есть

текст располагается на диске, в то время как задание находится в

очереди. Это позволяет прикладным программам достаточно быстро

распечатывать тексты, помещая их в очередь, так как для

продолжения работы приложению не требуется дожидаться окончания

распечатки.

2.4 Структура файловой системы

Файловая система разделяется на несколько частей: файловая

система root, состоящая из каталогов /bin, /lib, /etc, /dev и

некоторых других, файловая система /usr, где хранятся различные

программы и данные не подлежащие изменению, файловая система /var,

где содержатся изменяемые файлы (такие как log файлы и др.) и

файловая система /home, которая состоит из личных каталогов

пользователей. Разделение может существенно отличатся от выше

указанного в зависимости от работы системного администратора и

конфигурации аппаратного обеспечения.

В главе 5 сруктура файловой системы рассматривается более

подробно (см. также Linux Filesystem Standard).

.

- 16 -

Глава 3 Запуск Системы и Перезагрузка

В этой главе описывается то, что происходит в системе Linux

при ее запуске и перезагрузке и как это правильно сделать.

3.1 Обзор

Процесс включения компьютера и загрузки операционной системы

называется запуском. Во время запуска сначала загружется небольшая

программа, называемая начальнам загрузчиком, которая в свою

очередь загружет в память и запускает операционную систему.

Начальный загрузчик обычно находится в определенном месте на

жестком диске или дискете. Потому как Linux это довольно сложная и

большая система, ее запуск производится в два этапа, хотя первично

загружаемый код должен быть достаточно маленьким (несколько сотен

байт).

На разных компьютерах начальная загрузка производится по

разному. На персональных компьютерах сначала считывется первый

сектор дискеты или жесткого диска (посредством процедур BIOS),

который назывется загрузочным сектором. В этом секторе находится

начальный загрузчик, который затем загружает операционную систему,

которая может быть расположена в другом месте на диске или

где-либо еще.

После загрузки Linux, инициализируются драйверы устройств, а

затем запускается init(8), который в свою очередь запускает другие

процессы, позволяющие подключаться к системе и обеспечивающие

нормальную работу. Этот этап рассмотрен ниже более подробно.

Для перезапуска системы сначала все процессы должны быть

завершены (т.е. закрыты все используемые ими файлы и др.), затем

демонтируются файловые системы и swap-области и, в конце концов,

на экран выдается сообщение о том, что питание может быть

отключено. Если же такая процедура не будет произведена, то могут

произойти серьезные сбои в последующей работе системы. Например,

- 17 -

информация, хранящаяся в кэш буфере файловой системы, будет

утеряна, нарушится целостность файловой системы и, следовательно,

она будет не пригодна к использованию.

3.2 Процесс запуска при близком рассмотрении

Linux может быть запущена как с дискет, так и с жесткого

диска. В книге "Установка и запуск Linux" подробно описан запуск

системы.

При включении компьютера, сначала BIOS производит

тестирование оборудования, а затем запуск операционной системы.

Сначала выбирается устройство, с которого будет производится

запуск (обычно первый дисковод, если в него вставлена дискета, в

противном случае - первый жесткий диск, если он установлен, хотя

порядок выбора может быть настроен) и считывается самый первый

сектор, который называется загрузочным. Его также называют MBR

(Master Boot Record), так как у жесткого диска может быть

несколько разделов и у кажодого может быть свой загрузочный

сектор.

В загрузочном секторе находится небольшая программа

(относительно небольшая чтобы она могла разместится в одном

секторе), которая загружает и запускает операционную систему. При

загрузке с дискеты, в загрузочном секторе находится код, который

обеспечивает только считывание ядра системы в определенную заранее

область памяти. Загрузочная дискета для Linux не содержит никаких

файловых систем. Ядро записано на дискете как последовательность

блоков, так как это значительно упрощает процесс загрузки. Однако,

вполне можно загружаться с дискеты, на которой установлена

какая-нибудь файловая система, используя загpузчик LILO.

При загрузке с жесткого диска, код, расположенный в MBR,

проверяет таблицу разделов (также расположенную в MBR), определяет

активный раздел (раздел, используемый при загрузке), считывает

загрузочный сектор этого раздела и запускает считанный код. Код,

расположенный в загрузочном секторе активного раздела жесткого

диска, выполняет те же функции, что и код, находящийся в

- 18 -

загрузочном секторе дискеты: он считывает ядро из выбранного

раздела, а затем запускает его. Однако здесь существует много

тонкостей, так как использование отдельного раздела диска только

для хранения кода ядра неэффективно, поэтому код, расположенный в

загрузочном секторе раздела, не просто последовательно считывает

информацию с диска, а использует считывание по секторам.

Существует несколько способов решения этой проблемы, но наиболее

простым из них является использование LILO загрузчика (информацию

по установке и настройке LILO см. в документации по LILO).

При загрузке с использованием LILO обычно сразу же

загружается и запускается ядро, заданное по умолчанию, однако

можно сконфигурировать LILO так, чтобы можно было бы загрузить

одно из нескольких возможных ядер или даже другую операционную

систему (в добавление к Linux). Также можно указать требуемое ядро

или операционную сиситему во время загрузки. При нажатии клавиши

ALT, SHIFT или CTRL (после загрузки LILO) будет выдан запрос, где

можно указать ядро или систему. Однако при конфигурировании можно

установить опцию, при которой LILO будет всегда выдвать такой

запрос, а также указать время, по истечении которого загружается

ядро, установленное по умолчанию.

Существуют и другие загрузчики, подобные LILO, однако у него

есть несколько полезных функций, которых нет в других загрузчиках,

так как он был написан специально для Linux. Например, имеется

возможность передачи ядру параметров во время загрузки или

изменения некоторых опций, встроенных в ядро. Среди подобных

загрузчиков (bootlin, bootactv и др.) LILO является наилучшим

выбором.

Загрузка системы как с жесткого диска, так и с дискет имеет

свои преимущества, хотя загрузка с жесткого диска считается лучше

и быстрее, так как она позволяет избежать неудобства, связанные со

сменой дискет. Однако в некоторых случаях загрузка с дискет более

удобна. Например, при установке системы или при повреждении

файловой системы.

После того, как ядро системы загружено в память (с жесткого

- 19 -

диска или с дискет) и запущено, выполняются приблизительно

следующие действия:

Так как ядро Linux установлено в запакованном виде, то прежде

всего оно само себя распаковывает. Это выполняет небольшая

программа, расположенная в самом начале кода.

Если на компьютере установлена видеоплата sVGA,

поддерживающая нестандартные текстовые режимы (такие как 100x40),

выдается запрос для указания требуемого режима. При компиляции

ядра можно сразу указать используемый режим, чтобы он не

запрашивался системой во время загрузки. Режим также может быть

установлен при помощи LILO или rdev(8).

Затем ядро тестирует аппаратное обеспечение (жесткие диски,

дисководы, сетевые адаптеры и др.) и конфигурирует соответствующие

драйверы устройств. Во время этого процесса на экран выдаются

подсказывающие сообщения. Вот примерно то, что происходит во время

загрузки:

LILO boot:

Loading linux.

Console: colour EGA+ 80x25, 8 virtual consoles

Serial driver version 3.94 with no serial options enabled

tty00 at 0x03f8 (irq = 4) is a 16450

tty01 at 0x02f8 (irq = 3) is a 16450

lp\_init: lp1 exists (0), using polling driver

Memory: 7332k/8192 available (300k kernel code, 384k reserved, 176k data)

Floppy drive(s): fd0 is 1.44M, fd1 is 1.2M

Loopback device init

Warning WD8013 board not found at i/o = 280

Math coprocessor using irq13 error reporting

Partition check:

hda: hda1 hda2 hda3

VFS: Mounted root (ext filesystem)

Linux version 0.99.pl9-1 (root@haven) 05/01/93 14:12:20

Хотя текст сообщений довольно сильно различается на разных

- 20 -

системах и зависит от аппаратного обеспечения, версии Linux и

конфигурации.

После этого, ядро пытается смонтиpовать файловую систему

root. Место, куда она будет смонтирована, устанавливается во время

компиляции или с помощью rdev или LILO. Тип файловой системы

определяется автоматически. Если система root не монтиpуется,

например по причине того, что ядро не содержит драйвер

соответствующей файловой системы, то система зависает.

Файловая система root обычно монтиpуется в режиме read-only

(это устанавливается таким же образом как и узел монтиpования).

Это делает возможным проверку файловой системы в то время как она

смонтиpована, хотя проверка файловой системы, установленной в

режиме read-write не рекомедуется.

Затем ядро запускает программу init(8) в фоновом режиме (она

расположена в каталоге /sbin/init) которая становится главным

процессом. init выполняет различные функции, требуемые при

установке системы.

В конце концов init запускает программу getty(8) для

виртуальных консолей и последовательных линий. Эта программа

позволяет подключаться к системе посредством виртуальных консолей

и терминалов, подключенных через последовательные порты. init

может быть сконфигурирована также для запуска и других программ.

После этого процесс запуска системы считается завершенным и

система готова к работе.

3.3 Завершение работы и выключение системы

При выключении системы Linux необходимо выполнять некоторые

процедуры. Если этого не сделать, то файловые системы и файлы

могут повредиться. Это происходит по причине наличия в Linux

дискового кэша, информация из которого записывается на диск только

через некоторые промежутки времени. Это значительно повышает

производительность системы, но также означает, что если просто

- 21 -

выключить питание компьютера, то в дисковом кэше может находится

большое количество информации и файловая система может быть

частично повреждена, так как на диск обычно сбрасывается только

часть информации.

Другой причиной для этого является мультизадачность системы,

где одновременно может выполняться несколько процессов и

выключение питания может быть гибельным для системы. Особенно это

касается компьютеров, на которых одновременно работает несколько

пользователей.

Существуют команды, предназначенные для правильного

выключения системы - это shutdown(8) и halt(8), расположенные в

каталоге /sbin. Есть два обычных способа их применения.

Если система установлена на компьютере, где работает один

пользователь, то обычно завершают работу всех программ, работу

всех виртуальных консолей, входят в систему под пользователем root

(или остаются подключенными под этим пользователем, только в этом

случае нужно перейти в корневой каталог во избежание проблем с

демонтированием файловых систем), затем выполняется команда halt

или shutdown -h now (при желании можно установить задержку,

которая устанавливается заменой параметра now на знак '+' и числом

минут, по истечении которых будет завершена работа системы) или

просто halt.

Если на копьютере, на котором установлена система, работает

одновременно несколько пользователей, то возможно использование

команды shutdown в следующем формате:

shutdown -h +time message,

где time это время, по истечении которого работа системы будет

завершена, а message - сообщение, в котором объясняется причина

выключения. Напpимеp,

root# shutdown -h +10 'We will install a new disk. System should

> be back on-line in three hours.'

- 22 -

Выполнение этой команды предупредит каждого пользователя,

работающего в системе, что она будет выключена через 10 минут.

Сообщение выдается на каждый терминал, где работают пользователи,

включая xterm.

Broadcast message from root (ttyp0) Wed Aug 2 01:03:25 1995...

We will install a new disk. System should

be back on-line in three hours.

The system is going DOWN for system halt in 10 minutes !!

Выдача сообщения автоматически повторяется несколько раз

перед прекращением работы системы и каждый раз с более коротким

интервалом. При использовании halt нельзя установить задержку,

поэтому эта программа редко применяется на многопользовательских

системах.

Замечание: файл /etc/inittab содержит команды, выполняющиеся

при выключении системы.

После запуска процесса прекращения работы системы,

демонтируются все файловые системы (кроме системы root),

завершается выполнение всех процессов и программ-демонов, затем

демонтируется файловая система root и вся работа завершается.

После этого выдается сообщение, в котором говорится, что можно

отключить питание. Только после этого питание компьютера может

быть отключено.

В некоторых случаях невозможно завершить этот процесс

соответствующим образом. Например, при повреждении кода ядра в

памяти, нарушается его работа или система зависает и просто нет

возможности ввести новую команду можно только надеяться, что

ничего не повредится и выключить питание. Если же неполадки не

такие серьезные (например вышла из строя клавиатура), а ядро и

программа update работают нормально, то наилучшим вариантом будет

подождать несколько минут, пока update(8) не сохранит на диске

информацию, хранящуюся в кэш-буфере и только после этого выключить

- 23 -

питание.

Некоторые выключают копьютер после трехкратного выполнения

команды sync(8), которая сбрасывает на диск содержимое буфера, и,

после прекращения обращения к диску, выключают компьютер. Если в

момент выключения работа всех программ была завершена, то эта

процедура почти идентична выполнению команды shutdown. Однако,

файловые системы не демонтируются, что может привести к некоторым

проблемам, связанным с флагом 'clean filesystem' системы ext2fs. В

любом случае использование этого способа не рекомендуется.

3.4 Перезагрузка системы

Процесс перезагрузки может быть достигнут путем прекращения

работы системы, выключения питания и включения снова. Более

простой способ - это указать команде shutdown перезагрузить

систему установив опцию -r. Например, для этого можно использовать

команду shutdown -r now. Также можно использовать команду reboot.

3.5 Однопользовательский режим работы

Команда shutdown может также использоваться для перевода

системы в однопользовательский режим, в котором к системе никто не

может подключится кроме пользователя root, который использует для

работы главную консоль. Это иногда применяется для

административных целей, для выполнения которых не может быть

использована нормально работающая система.

3.6 Дискеты для экстенной загрузки

Не всегда имеется возможность загрузки системы с жесткого

диска. Например, при неправильных установках в LILO загрузчике

систему невозможно будет загрузить. В таких случаях должен быть

другой способ загрузки. Для персональных компьютеров обычно она

выполняется с дискет.

Большинство распространяемых версий Linux позволяют во время

установки системы создать загрузочную дискету. Однако многие такие

- 24 -

дискеты содержат только ядро и, предполагается, что для устранения

неполадок будут использоваться программы, находящиеся на

установочных дисках. Иногда этих программ бывает недостаточно,

например, когда требуется восстановить некоторые файлы, созданные

с помощью программ, которых нет на этих дисках.

Поэтому может возникнуть необходимость в создании специально

настроенного диска. В документации "Bootdisk HOWTO" содержится

необходимая информация для создания подобного диска.

При загрузке со специально настроенного диска нельзя

использовать привод, на котором смонтирована эта дискета, для

каких-либо других целей. Это может создать некоторые неудобства,

если в компьютере имеется только один дисковод. Однако, если

компьютер имеет достаточный объем памяти, можно загрузить этот

диск в RAM диск (для этого ядро, расположенное на дискете должно

быть сконфигурировано соответствующим образом). Это позволяет

использовать дисковод для других целей.

.

- 25 -

Глава 4 Использование Дисков и Других Устройств

Хранения Информации

При установке системы или изменении ее конфигурации обычно

возникает много проблем с дисками. Нужно установить файловые

системы на дисках так, чтобы на них могли хранится файлы, а также

зарезервировать пространство для других частей операционной

системы.

В этой главе рассматриваются все эти вопросы. Обычно, если

система уже установлена, то каких-либо проблем с дисками в

дальнейшем быть не должно, кроме pаботы с дискетами. При установке

нового диска или его настройке следует обращаться к этой главе.

При администрировании дисков могут возникнуть следующие

проблемы:

Форматирование диска (в настоящее время форматирование

является необязательной процедурой для многих типов дисков).

Разделение диска, если диск будет использоваться для разных

целей. Одна из причин разделения - это хранение разных

операционных систем на одном диске. Другая причина - хранение

пользовательских и системных файлов в pазных pазделах, что

упрощает резервное копирование и восстановление, а также защиту

системных файлов от повреждений.

Создание файловой системы на соответствующем диске или

разделе. В системе Linux диск ничего не значит, пока на нем не

установлена файловая система. Только после этого возможна работа с

файлами.

Монтирование различных файловых систем для формирования

единой структуры каталогов, как автоматически так и вручную

(файловые системы, монтируемые вручную, должны быть вpучную

демонтированы).

- 26 -

В главе 6 рассматриваются вопросы буфеpизации дисков и работы

с виртуальной памятью, что также нужно знать при работе с дисками.

В этой главе рассматриваются вопросы, которые необходимо

знать при работе с жесткими и гибкими дисками. К сожалению здесь

не рассматривается использование ленточных накопителей и приводов

для компакт дисков.

4.1 Типы устройств

В системе UNIX, а следовательно и Linux, существует два типа

устройств: блочные устройства с прямым доступом (такие как диски)

и символьные устройства (такие как ленточные накопители и

последовательные порты), некоторые из них могут быть

последовательными, а некоторые - с прямым доступом. Каждое

поддерживаемое устройство представляется в файловой системе файлом

устройства. При выполнении операций чтения или записи с подобным

файлом, происходит обмен данными между устройством, на которое

указывает этот файл. Такой способ доступа к устройствам позволяет

не использовать специальные программы (а также специальные методы

програмирования, такие как работа с прерываниями). Например, для

распечатки файла на принтере, используется следующая команда:

ttyp5 root ~ $ cat filename > /dev/lp1

ttyp5 root ~ $

Однако, для распечатки файлов используется специальная

программа (обычно lpr(1)), которая контролирует поток поступающих

файлов во избежание возникновения конфликтных ситуаций, например,

при попытке распечатать на одном принтере одновpеменно несколько

разных файлов. По мере распечатки, новые файлы поступают

автоматически. По подобной схеме работает большинство устройств.

Однако с файлами устройств проблемы возникают очень редко.

Так как устройства отображаются как файлы в файловой системе

(в каталоге /dev), несложно обнаружить с помощью команды ls(1)

какие существуют файлы устройств. После выполнения команды ls -l

- 27 -

на экран выводится список файлов, причем в первой колонке

содержится тип файла и права доступа к нему. Например, для

просмотра файла, соответствующего последовательному порту,

используется следующая команда:

ttyp5 root ~ $ ls -l /dev/cua0

crw-rw-rw- 1 root uucp 5, 64 Nov 30 1993 /dev/cua0

ttyp5 root ~ $

Первый символ в первой колонке, т.е. 'c', показывает тип

файла, в данном случае символьное устройство. Для обычных файлов

используется символ '-', для каталогов - 'd', для блочных

устройств - 'b' (см. pуководство к команде ls(1) для более

подробной информации).

Наличие большого количества файлов устройств совсем не

означает, что эти устройства на самом деле установлены. Наличие

файла /dev/sda ни о чем не говорит и совсем не означает, что в

компьютере установлен жесткий диск SCSI. Это предусмотрено для

облегчения установки программ и нового оборудования (нет

необходимости искать нужные параметры и создавать файлы для новых

устройств).

4.2 Жесткие диски

В этом разделе рассматриваются термины, связанные с

использованием жестких дисков.

Жесткий диск состоит из одной или нескольких круглых пластин,

одна или обе стороны которой покрыты магнитным материалом,

используемым для хранения информации. Для каждой стороны

предусмотрена головка, позволяющая считывать или записывать

информацию. Пластины вращаются на одной оси обычно со скоростью

3600 оборотов в минуту, хотя в более быстрых пpиводах используются

более высокие скорости. Головки перемещаются вдоль радиуса

поверхности пластин, что позволяет получить доступ к любой точке

поверхности.

- 28 -

Центральный процессор (CPU) и жесткий диск обмениваются

информацией через дисковый контроллер. Это упрощает схему

обращения и работы с диском, так как контроллеры для разных типов

дисков могут быть построены с использованием одного интерфейса для

связи с компьютером. Поэтому, например, для считывания сектора

можно воспользоваться всего лишь одной командой вместо сложных

последовательностей электрических сигналов для того, чтобы

переместить головки к нужной позиции, синхронизировать вращение

диска и считывание или запись данных и др. (на самом деле,

интерфейс между компьютером и контроллером тоже достаточно сложен,

но не на столько, на сколько он был бы без использования

контроллера). Котроллер также выполняет и некоторые другие

функции, такие как буфеpизация информации или автоматическая

замена плохих секторов.

Существуют еще некоторые понятия, знание которых необходимо

для понимания работы жесткого диска. Обычно поверхности делятся на

концентрические кольца, называемые дорожками или трэками, которые,

в свою очередь, делятся на сектора. Такое разделение нужно для

указания нужных позиций на диске и для распределения дискового

пространства на файлы. Для нахождения нужной информации на диске

достаточно примерно следующих данных: "поверхность 3, дорожка 5,

сектор 7". Обычно количество секторов на дорожке одинаково для

всех дорожек на диске, хотя в некоторых устройствах на внешних

трэках размещается большее количество секторов (все сектора имеют

один и тот же физический размер, поэтому на более длинных дорожках

помещается больше секторов). Стандартный размер сектора равен 512

байт. Диск не может оперировать данными, объем которых менее

одного сектора.

Каждая поверхность разделена на дорожки (и сектора) таким

образом, что при перемещении головки одной поверхности к

какой-либо дорожке, головки остальных поверхностей будут

установлены на этой же дорожке. Совокупность всех таких дорожек

называется цилиндром. Для перемещения головок от одной дорожки

(цилиндра) к другой требуется какое-то количество времени. Таким

образом, если разместить данные, доступ к которым чаще всего

производится сразу (например, файл), в одном цилиндре, то

- 29 -

необходимость в перемещении головок отпадает. Это повышает

производительность работы диска. Не всегда представляется

возможным разместить файл подобным образом. Файлы, которые

хранятся в разных местах на диске, называются фрагментированными.

Количество поверхностей (или головок, что в принципе одно и

то же), цилиндров и секторов сильно различается у разных

устройств. Совокупность таких параметров называется структурой

диска, которая хранится в специальной памяти, для питания которой

используются аккумуляторы. Эта память называется CMOS RAM, откуда

операционная система может считывать информацию во время ее

загрузки или во время установки драйвера.

К сожалению, BIOS построен так, что не представляется

возможным указать дорожку, номер которой превышает 1024, для

записи в CMOS RAM, что является серьезным ограничением для дисков

больших объемов. Для решения этой проблемы контроллер жесткого

диска передает заведомо неправильную информацию о структуре диска

и преобразует данные, представляемые компьютером, в нечто,

соответсвующее реальности. Например, жесткий диск может состоять

из 8 головок, 2048 дорожек с 35 секторами в каждой. В то время как

контроллер может утверждать, что диск имеет 16 головок и 1024

дорожки с 35 секторами в каждой, не превышая предела на хранение в

CMOS RAM числа дорожек и преобразуя адресацию уменьшая номер

головки вдвое и удваивая номер дорожки. Преобразование адресов

искажает представление операционной системы о структуре диска, что

усложняет размещение требуемой информации на одном цилиндре для

увеличения производительности.

Преобразование используется только для IDE дисков. В SCSI

дисках используется доступ с применением последовательного номера

сектора (который контроллер преобразует в номер головки, цилиндра

и сектора диска) и другой метод обмена информацией с процессором.

Однако, процессор может не иметь представления о реальной

структуры диска.

Так как системе Linux часто не известна информация о

структуре диска, то в файловых системах не используется размещение

- 30 -

отдельных файлов в пределах одного цилиндра. Вместо этого

применяется размещение файлов в цепочках последовательно

расположенных секторов, что дает приблизительно одинаковую

производительность. Хотя проблема усложняется за счет

использования специальных возможностей контроллера, таких как

внутреннее кэширование и других автоматических функций.

Каждый жесткий диск представлен отдельным файлом. Для IDE

дисков обычно существует только два таких файла. Они известны как

/dev/hda и /dev/hdb соответственно. Для SCSI дисков используются

файлы /dev/sda и /dev/sdb и т.д. Подобные обозначения применяются

и для других типов дисков. Файлы устройств для жестких дисков

предоставляют доступ к целому диску, не рассматривая разделы

(которые будут описаны ниже) и поэтому не составляет труда

перепутать разделы диска или информацию в них, если не быть

достаточно осторожным. Файлы жестких дисков обычно используются

для доступа к информации в MBR (которые также рассмотрены ниже).

4.3 Гибкие диски

Гибкий диск состоит из мягкой пластины, покрытой с одной или

обоих сторон материалом, подобным тому, которым покрыты пластины в

жестком диске. У самой дискеты нет никаких головок, они

установлены в приводе. Дискету можно сравнить с одной пластиной,

установленной в жестком диске, только дискета является съемной и

привод может использоваться для работы с различными дисками, в то

время как жесткий диск является одним неделимым устройством.

Также как жесткий диск, дискета делится на дорожки и сектора

(а две соответствующие дорожки на разных сторонах составляют

цилиндр), но их намного меньше, чем на жестком диске.

Дисковод может работать с несколькими типами дискет.

Например, привод на 3.5 дюйма может работать с дисками на 720 Кб и

1.44 Мб. Так как при использовании разных типов дисков, работа

самого привода немного различается, к тому же операционная система

должна иметь представление об объеме диска, существует множество

файлов устройств для работы с приводами для гибких дисков.

- 31 -

Напpимеp, файл /dev/fd0H1440 соответствует первому приводу (fd0)

формата 3.5 дюйма с дискетой на 3.5 дюйма высокой плотности (H)

объемом 1440 Кб (1440), т.е. позволяет pаботать с обычными

дискетами на 3.5 дюйма.

Имена файлов для приводов гибких дисков довольно сложные,

поэтому в системе Linux существует специальный тип устройства,

который автоматически определяет тип используемого гибкого диска.

Метод определения заключается в последовательном чтении первого

сектора вставленной дискеты с пpименением различных способов

чтения, до тех пор, пока он не будет правильно считан.

Естественно, диск должен быть сначала отформатирован.

Автоматическими устройствами являются /dev/fd0, /dev/fd1 и т.д.

Параметры для автоматических устройств, которые используются

для доступа к диску, могут быть установлены с помощью программы

setfdprm(8). Это может быть полезно в некоторых случаях, например,

если используются дискеты нестандартного объема (т.е. дискета

имеет нестандартное количество секторов в дорожке) или если

определение типа диска по какой-либо причине не работает и

соответствующий файл устройства отсутствует.

4.4 Форматирование

Форматирование - это процесс записи специальных отметок на

магнитную поверхность, которые используются для разделения дорожек

и секторов. Перед форматированием диска его поверхность состоит из

смеси различных магнитных сигналов. При форматировании эти сигналы

упорядочиваются и происходит формирование дорожек и секторов. В

действительности, все намного сложнее и выходит за рамки этой

книги. Нужно знать только то, что диск не может использоваться, до

тех пор пока он не будет отформатирован.

При работе в MS-DOS, форматирование также включает в себя

процесс создания файловой системы. Там часто эти два процесса

совмещены, особенно при работе с гибкими дисками. Но если нужно

сделать разграничение, то действительным форматированием называют

форматированием на низком уровне, а создание файловой системы -

- 32 -

форматированием на высоком уровне. При работе в системе UNIX (а

также в этой книге) вместо этих двух понятий будут использоваться

понятия форматирование и, соответственно, формирование файловой

системы.

Для IDE и некоторых SCSI дисков форматирование производится

при их изготовлении и, обычно, не требуется повторения этой

процедуры, поэтому большинство людей редко об этом задумываются. В

действительности, форматирование диска может привести к ухудшению

его работы, например, по причине того, что диск должен быть

отформатирован специальным образом для обеспечения возможности

замены плохих секторов.

Форматируемые диски часто поставляются со специальной

программой, потому как внутренние интерфейсы у разных приводов

различны. Эта программа обычно раположена в микросхеме BIOS

контроллера или поставляется отдельно как программа для MS-DOS. Ни

одни из них не могут быть использованы для системы Linux.

Во время форматирования могут быть обнаружены плохие блоки

или сектора, которые не должны быть использованы при дальнейшей

работе. Эти функции возлагаются на файловую систему. Хотя можно

создать небольшой раздел диска, который включает в себя только

плохие блоки. Это эффективно при большом количестве плохих блоков,

так как при работе файловой системы могут возникнуть некоторые

трудности, связанные с размером неиспользуемой области.

Для форматирования дискет используется программа fdformat(8).

В качестве параметра указывается файл устройства. Например,

следующая команда используется для форматирования обычной дискеты

размером 3.5 дюйма высокой плотности в первом приводе для гибких

дисков:

ttyp5 root ~ $ fdformat /dev/fd0H1440

Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.

Formatting ... done

Verifying ... done

ttyp5 root ~ $

- 33 -

Если для форматирования используется автоматическое

устройство (например, /dev/fd0), то сначала нужно указать

параметры этого устройства с помощью программы setfdprm(8). Для

получения такого же результата, как в предыдущем примере, нужно

выполнить следующие действия:

ttyp5 root ~ $ setfdprm /dev/fd0 1440/1440

ttyp5 root ~ $ fdformat /dev/fd0

Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.

Formatting ... done

Verifying ... done

Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.

Formatting ... done

Verifying ... done

ttyp5 root ~ $

Обычно проще указать точный файл устройства, который

соответствует типу форматируемого диска.

Программа fdformat также используется для выявления плохих

блоков. Она обрабатывает плохой блок несколько раз. Если проблема

не очень серьезна (загрязненная рабочая поверхность

считывающих/записывающих головок, плохой контакт в разъеме

контроллера), то fdformat продолжит свою работу, но возникновение

реальной ошибки прервет процесс проверки. Ядро отображает

появление каждой ошибки на терминале. Если используется syslog, то

сообщение поступает в файл /usr/adm/messages. fdformat не сообщает

специфику ошибки (обычно это не имеет значения, так как дисководы

это довольно дешевые устройства и их замена не составляет

проблем).

ttyp5 root ~ $ fdformat /dev/fd0H1440

Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.

Formatting ... done

Verifying ... read: Unknown error

ttyp5 root ~ $

- 34 -

Команда badblocks(8) используется для поиска плохих блоков на

любом диске или разделе диска (включая гибкие диски). Она не

форматирует диск, поэтому может быть использована для проверки

даже существующих файловых систем. В следующем примере

рассматривается проверка 3.5 дюймовой дискеты с двумя плохими

блоками.

ttyp5 root ~ $ badblocks /dev/fd0H1440

718

719

ttyp5 root ~ $

Программа выводит номера найденных плохих блоков. Во многих

файловых системах есть средства, позволяющие избежать

использования таких блоков. Для таких целей существует список

известных плохих блоков, который инициализируется при установке

файловой системы и может быть модифицирован в дальнейшем.

Первичный поиск плохих блоков производится при выполнении команды

mkfs (которая инициализирует файловую систему), в последующем

проверка производится с помощью программы badblocks, а модификация

списка - при помощи команды fsck. Эти команды будут рассмотрены

ниже.

4.5 Дисковые разделы

Весь жесткий диск может быть разбит на несколько разделов,

причем каждый раздел представлен так, как если бы это был

отдельный диск. Разделение используется, например, при работе с

двумя операционныи системами на одном диске. При этом каждая

операционная система использует для работы отдельный раздел и не

взаимодействует с другими. Таким образом, две различные системы

могут быть установлены на одном жестком диске. Без использования

разделов в данном случае возникла бы необходимость в приобритении

второго диска.

Для гибких дисков разделы не предусмотрены. В большинстве

случаев для этого нет необходимости, так как их объем достаточно

мал.

- 35 -

4.5.1 MBR, загрузочные сектора и таблица разделов

Информация о разделении жесткого диска находится в первом

секторе (т.е. в первом секторе первой дорожки первого диска). Этот

сектор называется MBR (сокращение от Master Boot Record) этого

диска. При загрузке компьютера BIOS загружает его в память и

выполняет. MBR содержит небольшую программу, которая считывает

таблицу разделов, находит активный раздел (т.е. раздел, отмеченный

как загрузочный) и считывает первый сектор этого раздела, который

называется загрузочным сектором (MBR также является загрузочным

сектором, но он выполняет специальные функции и поэтому имеет

отдельное название). Этот сектор содержит другую небольшую

программу, которая, в свою очередь, считывает начальную часть

операционной системы, расположенной в этом разделе, а затем

выполняет ее.

Схема разделения не встроена в оборудование или даже в BIOS.

Это только стандарт, которого придерживается большое количество

операционных систем. Не все системы поддерживают его, но они

являются исключениями. Некоторые системы поддерживают разделение,

но они занимают всего один раздел на диске и используют свою

внутреннюю схему разделения в пределах используемого раздела.

Такие операционные системы нормально работают с другими системами

(включая Linux), которые находятся на том же диске. Но те

операционные системы, которые не поддерживают разделы, не могут

быть установлены вместе с другими системами на одном диске.

Из мер предосторожности следует записать таблицу разделов.

Если эта таблица каким-либо образом повредится, то все файлы

останутся в сохранности (испорченная таблица разделов может быть

исправлена при помощи программы fdisk).

4.5.2 Расширенные и логические разделы

Изначально, в схеме разделения жесткого диска в PC

допускалось использование только четырех разделов. Но вскоре этого

оказалось недостаточно, частично по причине того, что многим для

- 36 -

работы требуется более четырех операционных систем (например,

Linux, MS-DOS, OS/2, Minix, FreeBSD, NetBSD, Windows/NT и т.д.),

но в основном из-за того, что одной системой используется

несколько разделов. Например, в системе Linux swap-область чаще

всего размещается в отдельном разделе (а не в основном разделе

Linux) для повышения скорости обмена (см. ниже).

Для решения этой проблемы была разработана схема,

использующая расширенные разделы. Она позволяет разбивать основной

раздел на подразделы. Основной раздел, разбитый таким образом,

называется расширенным разделом, а подразделы называются

логическими разделами. Они функционируют так же, как и основные

разделы, различие состоит в схеме их создания.

Ниже дан пpимеp pазбиения жеского диска на pазделы. Весь диск

разбит на три основных раздела, второй из которых разбит на два

логических. Часть диска не используется вообще. Весь диск, как

целое, и каждый основной раздел имеют свой загрузочный сектор.

иммммммммммммммммммммммммммммммммммммммv

· MBR ·

лмммммммммммммммммммммммммммммммммммммм¦ддддддддддддд

· Загрузочный сектор ·

·--------------------------------------· Основной

· ·

· Область данных раздела · раздел

· ·

лмммммммммммммммммммммммммммммммммммммм¦ддддддддддддд

· Загрузочный сектор · Ё

·дддддддддддддддддддддддддддддддддддддд· Ё

· Неиспользуемый загрузочный сектор · ЛогическийЁ

·--------------------------------------· Ё

· · раздел Ё

· Область данных раздела · Ё

· · Ё Расширенный

·дддддддддддддддддддддддддддддддддддддд·дддддддддддЇ

· Неиспользуемый загрузочный сектор · Ё раздел

·--------------------------------------· ЛогическийЁ

- 37 -

· · Ё

· Область данных раздела · раздел Ё

· · Ё

лмммммммммммммммммммммммммммммммммммммм¦ддддддддддддд

· ·

· Неиспользуемое дисковое пространство ·

· ·

лмммммммммммммммммммммммммммммммммммммм¦ддддддддддддд

· Загрузочный сектор ·

·--------------------------------------· Основной

· ·

· Область данных раздела · раздел

· ·

хмммммммммммммммммммммммммммммммммммммм№ддддддддддддд

4.5.3 Типы разделов

Таблицы разделов (одна находится в MBR, другие используются

для расширенных разделов) содержат один байт для каждого раздела,

который указывает тип раздела. Это позволяет определить

операционную систему, которая использует раздел или для чего он

используется во избежание случайного размещения двух систем на

одном разделе. Однако, в действительности, операционные системы

игнорируют байт типа раздела. Например, система Linux вообще не

имеет представления о его существовании. Хуже того, некоторые

системы неправильно его интерпретируют (по крайней мере, некоторые

версии DR-DOS игнорируют самый важный бит этого байта, в отличие

от других).

Не существует никаких стандартов, касающихся значений этих

байтов, хотя некоторые общепринятые значения приведены в таблице

ниже. Такую же информацию предоставляет программа Linux fdisk.

0 пустой раздел 40 Venix 80286 94 Amoeba BBT

1 DOS 12-битная FAT 51 Novell a5 BSD/386

2 XENIX root 52 Microport b7 BSDI fs

3 XENIX usr 63 GNU HURD b8 BSDI swap-область

- 38 -

4 DOS 16-бит (<32Мб) 64 Novell c7 Syrinx

5 расширенный 75 PC/IX db CP/M

6 DOS 16-бит (>=32Мб) 80 Old MINIX e1 DOS

7 OS/2 HPFS 81 Linux/MINIX e3 DOS r/o

8 AIX 82 Linux swap-область f2 DOS дополнительный

9 AIX загрузочный 83 Linux ff BBT

a OS/2 загрузочный 93 Amoeba

4.5.4 Разделение жесткого диска

Существует много программ, позволяющих создавать и удалять

разделы. У большинства операционных систем имеются свои

собственные и разумнее всего пользоваться именно такими

программами. Чаще всего эта программа называется fdisk (как и в

случае Linux). Особенности работы с ней рассмотрены в ее

руководстве. Команда cfdisk подобна fdisk, только в первой

используется полноэкранный интерфейс.

При pаботе с IDE дисками, загрузочный раздел (раздел, в

котором находятся файлы, используемые при загрузке и само ядро)

должен полностью располагаться в пределах первых 1024 цилиндров,

потому как во время загрузки работа с диском происходит через BIOS

(перед переходом системы в защищенный режим), а BIOS не может

оперировать с цилиндрами, номер которых больше, чем 1024. Иногда

представляется возможным использование загрузочного раздела, лишь

частично расположенного в пределах первых 1024 цилиндров. Данный

метод работает до тех пор, пока все файлы, считываемые посредством

BIOS, находятся в пределах 1024 цилиндров. Так как это сделать

довольно сложно, то пpименение этого метода не рекомедуется.

Сложно предугадать, когда после дефрагментации или сбрасывании

содержимого буфера на диск система перестанет загружаться. Поэтому

следует удостовериться в том, что загрузочный раздел расположен в

пределах первых 1024 цилиндров.

Некоторые последние версии BIOS и недавние модели IDE дисков

в действительности позволяют pаботать с цилиндрами, номер которых

превышает 1024.

- 39 -

Каждый раздел должен содержать четное количество секторов,

так как в системе Linux используются блоки размером в 1 Кб, т.е.

два сектора. Нечетное количество секторов приведет к тому, что

последний из них будет неиспользован. Это ни на что не влияет, но

пpи запуске fdisk будет выдано пpедупpеждение.

При изменении размера раздела обычно требуется сначала

сделать резервную копию всей необходимой информации, удалить

раздел, создать новый раздел, а затем восстановить всю сохраненную

информацию на новый раздел. Хотя существует программа для MS-DOS

под названием fips, которая позволяет изменять объем раздела без

резервного копирования, но для других файловых систем эту опеpацию

необходимо пpоизводить.

4.5.5 Файлы устройств и разделы

Каждому основному и расширенному разделу соответствует

отдельный файл устpойства. Существует соглашение для имен подобных

файлов, которое состоит в добавлении номера раздела к имени файла

самого диска. 1-4 разделы являются основными (вне зависимости от

того, сколько существует основных pазделов), а 5-8 - логическими

(вне зависимости от того, к какому основному разделу они

относятся). Например, /dev/hda1 соответствует первому основному

разделу первого IDE жесткого диска, а /dev/sdb7 - третьему

расширенному разделу второго SCSI диска.

4.6 Файловые системы

4.6.1 Что такое файловая система?

Файловая система - это методы и структуры данных, которые

используются операционной системой для хранения файлов на диске

или его разделе. О файловой системе также говорят, ссылаясь на

раздел или диск, используемый для хранения файлов или тип файловой

системы.

Нужно видеть разницу между диском или разделом и

- 40 -

установленной на нем файловой системой. Некоторые программы

(например, программы установки файловой системы) при обращении к

диску или разделу используют прямой доступ к секторам. Если на

этом месте была файловая система, то она будет серьезно

повреждена. Большинство программ взаимодействуют с диском

посредством файловой системы, и, следовательно, их работа будет

нарушена, если на разделе или диске никакая система не установлена

(или тип файловой системы не соответствует требуемуму).

Перед тем, как раздел или диск могут быть использованы в

качестве файловой системы, она должна быть инициализирована, а

требуемые данные перенесены на этот диск. Этот процесс называется

созданием файловой системы.

У большей части файловых систем UNIX сходная структура, а их

некоторые особенности очень мало различаются. Основными понятиями

являются: суперблок, индексный дескриптор (inode), блок данных,

блок каталога и косвенный блок. В суперблоке содержится информация

о файловой системе в целом, например, ее размер (точная информация

зависит от типа файловой системы). В индексном дескрипторе

хранится вся информация о файле, кроме его имени. Имя файла

хранится в блоке каталога, вместе с номером дескриптора. Запись

каталога содержит имя файла и номер индексного дескриптора

соответствующего файла. В этом дескрипторе хранятся номера

нескольких блоков данных, которые используются для хранения самого

файла. В inode есть место только для нескольких номеров блоков

данных, однако, если требуется большее количество, то пространство

для указателей на блоки данных динамически выделяется. Такие блоки

называются косвенными. Для того, чтобы найти блок данных, нужно

сначала найти его номер в косвенном блоке.

В файловых системах UNIX обычно имеется возможность создания

дыр в файлах (это можно сделать с помощью команды lseek(2), см.

руководство). Это означает, что файловая система предоставляет

ложную информацию о том, что в каком-то месте в файле содержатся

нулевые байты, но в действительности для этого не выделяются

сектора (это означает, что файл будет занимать несколько меньше

места на диске). Это часто используется особенно в небольших

- 41 -

двоичных программах, библиотек Linux, в некоторых базах данных и в

других отдельных случаях. (Дыры реализуются хранением специального

значения в косвенном блоке или индексном дескрипторе вместо адреса

блока данных. Это специальное значение показывает, что для данной

части файла блоки данных не размещены и, следовательно, что в

файле есть дыра.)

Использование дыр достаточно эффективно. На компьютере с

общим дисковым пространством в 200 Мб, простые измерения

показывают, что применение дыр дает экономию в 4 Мб. Однако, эти

измерения проводились на системе, где было установлено

относительно мало программ и отсутствовали файлы баз данных. Метод

измерения дыр рассмотрен в приложении B.

4.6.2 Типы файловых систем

Linux поддерживает несколько типов файловых систем. Наиболее

важные из них рассмотрены ниже.

minix Считается самой старой и самой надежной файловой системой,

но достаточно ограниченной в своих возможностях (у файлов

отсутствуют некоторые временные параметры, длина имени файла

ограничена 30-ю символами) и доступных объемах (максимум 64 Мб на

одну файловую систему).

xia Модифицированная версия системы minix, в которой увеличена

максимальная длина имени файла и размер файловой системы, хотя она

не pеализует никаких новых возможностей.

ext2 Наиболее богатая функциональными возможностями файловая

система из семейства совместимых с Linux. На данный момент

считается самой популярной системой. Она разработана с учетом

совместимости с последующими версиями, поэтому для установки новой

версии кода системы не требуется устанавливать ее заново.

ext Предыдущая версия системы ext2, не совместима с последующими

версиями. В настоящее время она очень редко включается в пакеты

новых поставляемых систем, т.к. большинство пользователей сейчас

- 42 -

пользуются системой ext2.

В дополнение к рассмотренным выше, в Linux включена поддержка

еще некоторых файловых систем для обеспечения обмена файлами между

другими операционными системами. Эти файловые системы работают

также, как и описанные выше, кроме того, что их функциональные

возможности могут быть значительно ограничены по сравнению с

возможностями, обычно предоставляемыми файловыми системами UNIX.

msdos Обеспечивается совместимость с системой MS-DOS (а также

OS/2 и Windows NT).

umsdos Расширяет возможности драйвера файловой системы MS-DOS

для Linux таким образом, что при работе в Linux, имеется

возможность работы с именами файлов нестандартной длины, просмотра

прав доступа к файлу, ссылок, имени пользователя, которому

принадлежит файл, а также оперирование с файлами устройств. Это

позволяет использовать обычную систему MS-DOS, так, как если бы

это была система Linux. Таким образом, исключается необходимость

создания отдельного раздела для Linux.

iso9660 Стандартная файловая система для CD-ROM. Довольно

популярное развитие стандарта CD-ROM, выполненное Rock Ridge'м,

которое обеспечивает автоматическую поддержку имен файлов

нестандартной длины.

nfs Сетевая файловая система, обеспечивающая разделение

одной файловой системы между несколькими компьютерами для

предоставления доступа к ее файлам со всех машин.

hpfs Файловая система OS/2.

sysv Файловые системы System V/386, Coherent и Xenix.

Также существует файловая система proc, которая обычно

доступна через каталог /proc. В действительности, она не является

файловой системой, хотя по ее структуре сложно обнаружить разницу.

- 43 -

Эта система позволяет получить доступ к определенным структурам

данных ядра, к таким, как список процессов (отсюда название). Все

эти структуры выглядят как файловая система и ими можно

оперировать обычными средствами работы с файловой системой.

Например, для получения списка всех процессов, используется

следующая команда:

ttyp5 root ~ $ ls -l /proc

total 0

dr-xr-xr-x 4 root root 0 Jan 31 20:37 1

dr-xr-xr-x 4 liw users 0 Jan 31 20:37 63

dr-xr-xr-x 4 liw users 0 Jan 31 20:37 94

dr-xr-xr-x 4 liw users 0 Jan 31 20:37 95

dr-xr-xr-x 4 root users 0 Jan 31 20:37 98

dr-xr-xr-x 4 liw users 0 Jan 31 20:37 99

-r--r--r-- 1 root root 0 Jan 31 20:37 devices

-r--r--r-- 1 root root 0 Jan 31 20:37 dma

-r--r--r-- 1 root root 0 Jan 31 20:37 filesystems

-r--r--r-- 1 root root 0 Jan 31 20:37 interrupts

-r-------- 1 root root 8654848 Jan 31 20:37 kcore

-r--r--r-- 1 root root 0 Jan 31 11:50 kmsg

-r--r--r-- 1 root root 0 Jan 31 20:37 ksyms

-r--r--r-- 1 root root 0 Jan 31 11:51 loadavg

-r--r--r-- 1 root root 0 Jan 31 20:37 meminfo

-r--r--r-- 1 root root 0 Jan 31 20:37 modules

dr-xr-xr-x 2 root root 0 Jan 31 20:37 net

dr-xr-xr-x 4 root root 0 Jan 31 20:37 self

-r--r--r-- 1 root root 0 Jan 31 20:37 stat

-r--r--r-- 1 root root 0 Jan 31 20:37 uptime

-r--r--r-- 1 root root 0 Jan 31 20:37 version

ttyp5 root ~ $

(В действительности, должно быть еще несколько файлов, не

соответствующих процессам, однако, этот пример немного укорочен.)

Хотя система /proc и называется файловой, ни одна ее часть не

взаимодействует с диском. Она существует только в представлении

ядра и при попытке обращения к какой-либо ее части, создается

- 44 -

впечатление, что эта часть где-то существует, хотя в

действительности это не так. Даже если существует файл /proc/kmem

в несколько мегабайт, он не занимает места но диске.

4.6.3 Какую файловую систему устанавливать?

Обычно мало смысла в пpименении нескольких разных файловых

систем. В настоящее время наиболее популярной считается система

ext2fs и, возможно, является наилучшим выбором. В зависимости от

различных параметров (скорость, производительность, надежность,

совместимость и др.) может оказаться, что установка другой

файловой системы будет более приемлемым вариантом.

4.6.4 Установка файловой системы

Файловая система устанавливается, т.е. инициализируется, при

помощи команды mkfs(8). В действительности, существуют отдельные

программы для каждого типа файловой системы. Команда mkfs только

запускает требуемую программу в зависимости от типа

устанавливаемой системы. Тип файловой системы указывается при

помощи опции -t fstype.

Параметры, передаваемые программам, вызываемым mkfs, слегка

различаются. Наиболее важные из них рассмотрены ниже (для более

подробной информации см. руководство).

-t fstype Указывается тип файловой системы.

-c Производится поиск плохих блоков и, соответственно,

инициализация списка плохих блоков.

-l filename Считывается начальный список плохих блоков из файла

filename.

Для установки файловой системы ext2 на дискету, используется

следующая последовательность команд:

ttyp5 root ~ $ fdformat -n /dev/fd0H1440

- 45 -

Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.

Formatting ... done

ttyp5 root ~ $ badblocks /dev/fd0H1440 1440 > bad-blocks

ttyp5 root ~ $ mkfs -t ext2 -l bad-blocks /dev/fd0H1440

mke2fs 0.5a, 5-Apr-94 for EXT2 FS 0.5, 94/03/10

360 inodes, 1440 blocks

72 blocks (5.00%) reserved for the super user

First data block=1

Block size=1024 (log=0)

Fragment size=1024 (log=0)

1 block group

8192 blocks per group, 8192 fragments per group

360 inodes per group

Writing inode tables: done

Writing superblocks and filesystem accounting information: done

ttyp5 root ~ $

В первую очередь дискета форматируется (параметр -n

предотвращает проверку на наличие плохих блоков). Затем

производится поиск плохих блоков при помощи команды badblocks,

вывод которой перенаправлен в файл bad-blocks. И, наконец,

файловая система устанавливается с инициализацией списка найденных

плохих блоков.

Вместо использования badblocks, команде mkfs может быть

указан параметр -c, как это видно из примера, рассмотренного ниже.

ttyp5 root ~ $ mkfs -t ext2 -c /dev/fd0H1440

mke2fs 0.5a, 5-Apr-94 for EXT2 FS 0.5, 94/03/10

360 inodes, 1440 blocks

72 blocks (5.00%) reserved for the super user

First data block=1

Block size=1024 (log=0)

Fragment size=1024 (log=0)

1 block group

8192 blocks per group, 8192 fragments per group

360 inodes per group

- 46 -

Checking for bad blocks (read-only test): done

Writing inode tables: done

Writing superblocks and filesystem accounting information: done

ttyp5 root ~ $

Указание параметра -c намного удобнее, чем применение команды

badblocks, но ее использование необходимо для проверки файловой

системы после ее установки.

Установка файловых систем на жесткий диск или его раздел

аналогична установке на дискету, исключая форматирование.

4.6.5 Монтирование и демонтирование

Перед работой с файловой системой, она должна быть

смонтирована. При этом операционная система выполняет некоторые

действия, обеспечивающие функционирование монтируемой системы. Так

как все файлы в системе UNIX принадлежат одной структуре

каталогов, то эта операция обеспечивает работу с файловой

системой, как с каталогом уже смонтированной.

Рассмотрим три различные файловые системы. Если две последние

системы (2-ю и 3-ю) соответственно смонтировать к каталогам /home

и /usr первой системы, то в итоге образуется файловая система с

единой структурой каталогов (4).

1] 2] 3]

/ ддбддддд bin / ддбдддддд abc / ддбдддддд bin

Ё Ё Ё

цддддд dev цдддддд liw цдддддд etc

Ё Ё Ё

цддддд home юдддддд ftp юдддддд lib

Ё

цддддд etc

Ё

цддддд lib

- 47 -

Ё

юддддд usr

4]

/ ддбдддбд usr

Ё Ё

Ё цдддддд lib

Ё Ё

Ё цдддддд etc

Ё Ё

Ё юдддддд bin

Ё

цддддд lib

Ё

цддддд etc

Ё

цдддбд home

Ё Ё

Ё цдддддд ftp

Ё Ё

Ё цдддддд liw

Ё Ё

Ё юдддддд abc

Ё

цддддд bin

Ё

юддддд dev

В примере, рассмотреном ниже, показано, как это сделать.

ttyp5 root ~ $ mount /dev/hda2 /home

ttyp5 root ~ $ mount /dev/hda3 /usr

ttyp5 root ~ $

Команда mount(8) принимает два параметра. Первый их них -

файл устройства, соответствующий диску или разделу, на котором

раположена файловая система. Вторым параметром является имя

- 48 -

каталога, к которому будет монтироваться система. После выполнения

этих команд содержимое файловых систем отображается в каталогах

/home и /usr соответственно. Также можно сказать, что раздел

/dev/hda2 смонтирован к каталогу /home, а /dev/hda3 - к каталогу

/usr. Существует различие между файлом устройства, /dev/hda2, и

монтируемым каталогом, /home. Файл устройства предоставляет доступ

к 'сырым' данным, расположенным на диске, а монтируемый каталог -

к файлам. Такой каталог называется узлом монтирования.

Монтируемый каталог не обязательно должен быть пустым, хотя

он должен существовать. Однако все файлы, в нем расположенные,

будут недоступны после монтирования файловой системы. (Открытые

ранее файлы будут также доступны, а файлы, являющиеся жесткими

ссылками из других каталогов, будут доступны с использованием имен

ссылок.) Таким образом, никакого ущерба не наносится и это даже

может быть полезно. Например, некоторые делают каталог /tmp

символической ссылкой на каталог /usr/tmp. При загрузке системы,

когда файловая система /usr не смонтирована, каталог размещается в

системе root. После того, как /usr смонтирована, каталог /usr/tmp,

расположенный в файловой системе root, становится недоступным.

Если же /usr/tmp не существует в системе root, то перед

монтированием /usr создание и pабота с временными файлами будет

невозможна.

Для защиты файловой системы от записи, команда mount

запускается с опцией -r, после чего монтирование производится в

режиме read-only. После этого ядро пресекает любые попытки записи,

включая модификацию времени доступа к файлам в индексном

дескрипторе. Монтирование с защитой от записи используется при

работе с такими устройствами, как CD-ROM.

Возникает вопрос: каким же образом монтируется самая первая

файловая система (т.е. система root), так как очевидно, что она не

может быть смонтирована на какую-либо другую. Система root

монтируется во время загрузки, поэтому считается, что она всегда

установлена (если бы она не была установлена, то компьютер не смог

бы загрузиться). Название файловой системы, используемой для

монтирования root, либо встроено в ядро, либо устанавливается при

- 49 -

помощи LILO или rdev.

Обычно сначала система root монтируется в режиме read-only.

Затем запускается программа fsck(8) для проверки ее целостности и

если все в порядке, то система монтируется снова в режиме

read-write. fsck не следует запускать на смонтированной файловой

системе, так как изменения, произведенные при ее выполнении, могут

привести к повpеждению системы. Так как система root сначала

монтируется в режиме read-only, то после ее проверки все неполадки

могут быть полностью устранены при повторном монтировании.

На многих системах существуют и другие файловые системы,

которые должны быть смонтированы во время загрузки. Их список

содержится в файле /etc/fstab (см. руководство к fstab(5)).

Если файловая система для работы больше не требуется, то она

может быть демонтирована. Для этого используется команда umount(8)

с одним параметром. Это может быть как файл устройства, так и узел

монтирования. Например, для демонтирования каталогов,

рассмотренных в предыдущем примере, используются следующие

команды:

ttyp5 root ~ $ umount /dev/hda2

ttyp5 root ~ $ umount /usr

ttyp5 root ~ $

После работы с дисководом следует каждый раз применять эту

команду, так как до демонтирования системы нельзя быть уверенным,

что данные были записаны на диск, а не остались в буфере.

Для выполнения операций монтирования и демонтирования

требуется наличие прав доступа пользователя root. Однако, многие

пользователи довольно часто работают с дисководом и для решения

этой проблемы существует несколько способов:

Сообщить всем пароль пользователя root. Это самый простой, но

далеко не лучший выход. Он может использоваться на некоторых

системах, не нуждающихся в защите (обычно не подключенных к

- 50 -

какой-либо сети).

Применять какую-либо программу (например, sudo(8)),

позволяющую всем использовать команду mount. Это также не лучший

способ по причине плохой защиты, хотя его применение не

предоставляет напpямую права root каждому пользователю.

Применение пакета mtools, используемого только для работы с

файловой системой MS-DOS без выполнения операции монтирования.

Используется только в тех случаях, когда дисковод применяется для

работы с дисками системы MS-DOS.

Поместить список файлов устройств, используемых при работе с

гибкими дисками, и доступных узлов монтирования вместе с нужными

опциями в файл /etc/fstab.

Последний метод может быть реализован путем добавления в файл

/etc/fstab следующей строки:

/dev/fd0 /floppy msdos user,noauto

Сначала указывается файл устройства, затем каталог, на

который монтируется устройство, тип файловой системы и опции.

Опция noauto запрещает автоматическое монтирование при начальной

загрузке системы. Опция user позволяет любому пользователю

монтировать указанную файловую систему и, по причине защиты

системы, запрещает выполнение программ и работу с файлами

устройств, расположенных на смонтированной системе. После этого,

любой пользователь может выполнить следующую команду:

ttyp5 root ~ $ mount /floppy

ttyp5 root ~ $

Для демонтирования системы используется команда umount с

соответствующими параметрами.

4.6.6 Поддержка работоспособности файловых систем

- 51 -

Файловые системы это достаточно сложные объекты, поэтому

иногда их функционирование нарушается. Для проверки целостности и

работоспособности файловой системы используется команда fsck(8).

Наиболее часто возникающие тpудности связаны с перебоями в

питании, неполадках в оборудовании или ошибках оператора

(например, некорректное выключение системы).

Большинство систем сконфигурировано таким образом, что

команда fsck запускается автоматически при загрузке системы,

поэтому возможные неполадки будут обнаружены (и, возможно,

исправлены) перед тем, как система будет использоваться. Работа с

поврежденной файловой системой может привести к потерям данных и

другим нарушениям ее функционирования. Однако, если файловая

система довольно большая по объему, то ее проверка может занять

некоторое время, а так как неполадки случаются очень редко, то

если система была выключена корректно, пpименяются определенные

методы для избежания проверки файловой системы. Первый из них

связан с тем, что если существует файл /etc/fastboot, то никаких

проверок не производится. Второй метод заключается в том, что в

файловой системе ext2 существует специальный флаг, раположенный в

суперблоке, который используется для выявления коppектности

демонтирования системы пpи последнем выключении системы. Эта

возможность используется в программе e2fsck (версия команды fsck

для файловой системы ext2fs) для избежания излишней проверки

файловой системы, если флаг ее целостности установлен (то есть

система была коppектно демонтирована). Фунционирование метода,

используещего файл /etc/fastboot, зависит от файлов, запускаемых

при загрузке системы, в то время как применение команды e2fsck

работает в любом случае (см. руководство по e2fsck(8) для более

подробной информации).

Автоматическая проверка используется только для файловых

систем, устанавливаемых во время загрузки. Для проверки других

систем команда fsck должна выполняться отдельно.

Если fsck находит неисправность, не подлежащую

восстановлению, то могут потребоваться глубокие знания и понимание

работы файловых систем и их типов. Также могут потребоваться

- 52 -

резервные копии. Некоторую информацию по тем или иным вопросам

можно найти через телеконференции, связанные с системой Linux.

Также может потpебоваться программа debugfs(8), созданная Theodore

T.

Команда fsck должна использоваться только для демонтированных

систем (за исключением системы root, смонтированной в режиме

read-only во время загрузки), так как при ее работе используется

прямой доступ к диску и информация о внесении каких-либо изменений

в файловую систему может быть недоступна операционной системе,

что, обычно, приводит к нарушению ее работы.

Иногда следует проводить поиск плохих блоков при помощи

команды badblocks. При ее выполнении выводится список номеров

найденных плохих блоков. Этот список может быть использован

программой fsck для внесения изменений в структуру файловой

системы во избежание использования этих блоков для хранения

информации. В следующем примере показано как это сделать.

ttyp5 root ~ $ badblocks /dev/fd0H1440 1440 > bad-blocks

ttyp5 root ~ $ fsck -t ext2 -l bad-blocks /dev/fd0H1440

Parallelizing fsck version 0.5a (5-Apr-94)

e2fsck 0.5a, 5-Apr-94 for EXT2 FS 0.5, 94/03/10

Pass 1: Checking inodes, blocks, and sizes

Pass 2: Checking directory structure

Pass 3: Checking directory connectivity

Pass 4: Check reference counts.

Pass 5: Checking group summary information.

/dev/fd0H1440: \*\*\*\*\* FILE SYSTEM WAS MODIFIED \*\*\*\*\*

/dev/fd0H1440: 11/360 files, 63/1440 blocks

ttyp5 root ~ $

4.7 Диски без файловых систем

Не все диски или разделы используются как файловые системы.

Например, раздел swap-области не содержит файловой системы. Многие

дисководы используются в режиме эмуляции ленточного накопителя,

- 53 -

поэтому tar-файл или любой другой файл записывается

непосредственно на диск без использования какой-либо файловой

системы. У такого использования есть небольшое преимущество в

свободном пространстве (при установке файловой системы некоторая

часть дискового простьранства используется для системных целей) и

совместимости с другими системами. Например, файловый формат tar

является стандартным для всех систем, в то время как файловые

системы на разных платформах различаются. Дискеты экстренной

загрузки системы Linux также могут не содержать файловой системы.

Одна из причин использования прямого доступа к диску (без

применения файловой системы) это создание копий. Например, если

диск содержит частично поврежденную файловую систему, то при

возможности следует создать его копию перед тем, как пытаться

что-либо сделать. Для этого можно использовать программу dd(1).

ttyp5 root /usr/tmp $ dd if=/dev/fd0H1440 of=floppy-image

2880+0 records in

2880+0 records out

ttyp5 root /usr/tmp $ dd if=floppy-image of=/dev/fd0H1440

2880+0 records in

2880+0 records out

ttyp5 root /usr/tmp $

Сначала команда dd выполняет точную копию дискеты в файл

floppy-image, а затем записывает копию обратно на диск

(предполагается, что перед выполнением последней команды была

вставлена другая дискета).

4.8 Распределение дискового пространства

4.8.1 Схемы разделения дисков

Довольно сложно разбить диск на несколько разделов наилучшим

образом, так как на это влияет довольно много факторов.

Обычно используется отдельный раздел или диск для файловой

системы root, которая содержит каталоги /bin, /etc, /dev, /lib,

- 54 -

/tmp и некоторые другие, требуемые для нормальной загрузки и

запуска системы. Таким образом, все, что нужно для запуска системы

- это файловая система root. Для файловой системы /usr, личных

каталогов пользователей (обычно каталог /home) и для swap-области

используются отдельные диски или их разделы. Разделение каталогов

с пользовательскими файлами облегчает создание резервных копий,

так как обычно не требуется сохранять рабочие программы

(расположенные в каталоге /usr). Также возможно разделение системы

/usr между несколькими компьютерами в сети (с использованием NFS)

для уменьшения общего используемого дискового пространства.

Для жестких дисков небольшого объема лучше всего использовать

один раздел. При использовании большого диска обычно его разбивают

на несколько крупных разделов. Если в системе используется

несколько дисков, то, возможно, неплохим вариантом будет

установить файловую систему root (включая /usr) на один диск, а

личные каталоги пользователей - на другой.

4.8.2 Требования к дисковому пространству

При установке Linux будет предоставлена информация о

требуемом дисковом пространстве при различной конфигурации

системы. Отдельно устанавливаемые программы также могут

использовать подобную схему. Это помогает распределять место на

диске.

Размер области, используемой для файлов пользователей,

зависит от характера работы. Многие считают, что для личных

каталогов нужно использовать как можно больше пространства, хотя

минимально требуемый объем на разных системах сильно варьируется.

Для простейшей обработки текстов некоторым может потребоваться

всего несколько мегабайт, в то время как другим, работающим с

мощными графическими приложениями, могут потребоваться многие

гигабайты.

Размещение swap-области рассмотрено в разделе 6.5.

- 55 -

4.8.3 Примеры распределения жесткого диска

На рассматриваемом компьютере был установлен диск объемом 109

Мб. Сейчас на нем используется диск объемом 330 Мб. Ниже

рассматривается как и почему эти диски были разбиты.

Первый диск (109 Мб) был разбит следующим образом. На

компьютере были установлены системы MS-DOS и Linux. При этом, для

DOS использовался раздел объемом 20 Мб, 10-ти Мб раздел был

предназначен для swap-области и оставшиеся 79 Мб были отданы под

отдельный раздел, где хранились все файлы, необходимые для работы

Linux.

Другой диск размером 330 Мб был разбит следующим образом:

5 Мб файловая система root

10 Мб swap-область

180 Мб файловая система /usr

120 Мб файловая система /home

15 Мб дополнительный раздел

4.8.4 Использование дополнительного дискового пространства

Добавление дискового пространства в системе Linux довольно

просто, по крайней мере после установки нужного оборудования.

Требуемый диск форматируется, в случае необходимости, затем

создаются разделы и устанавливается файловая система (это было

рассмотрено выше). После этого добавляются соответствующие строки

в файл /etc/fstab, что позволяет автоматически монтировать

дополнительный раздел или диск.

4.8.5 Методы сохранения дискового пространства

Наилучший метод сохранения дискового пространства - это не

устанавливать неиспользуемые программы. Иногда можно удалить

неиспользуемые файлы, такие как ненужные шрифты для X11 или

некоторые библиотеки для C++.

- 56 -

Также в определенных случаях можно использовать сжатие

файлов. Существуют программы, такие как gzip(1) и zip(1),

позволяющие производить компрессию файлов и каталогов. Система

gzexe сжимает и разжимает файлы незаметно для пользователя. А

экспериментальная система DouBle производит компрессию файлов

незаметно для использующих их программ.

.

- 57 -

Глава 5 Обзор Структуры Каталогов

В этой главе рассмотрены наиболее важные составляющие

структуры каталогов системы Linux, основанные на стандарте FSSTND.

Также в общих чертах описывается разбиение общей структуры

каталогов на отдельные файловые системы и с какой целью.

5.1 Введение

Эта глава основана на стандарте файловой системы Linux FSSTND

версии 1.2, который выпущен для попытки стандартизировать

организацию структуры каталогов в системе Linux. Он может быть

использован для облегчения разработки и переноса программного

обеспечения для Linux, а также администрирования Linux систем. В

создании FSSTND сделана попытка последовать традициям и настоящим

перспективам развития Unix, что сближает Linux системы с более

профессиональными Unix системами.

Глава не является таким подробным описанием, как FSSTND.

Поэтому системный администратор должен прочитать его для полного

понимания всех аспектов.

В главе не рассматриваются форматы файлов и другие

подробности, так как цель главы - это сделать обзор всей системы

относительно файловой системы.

Построение структуры каталогов изначально предполагает ее

разбиение на отдельные части, каждая из которых может размещаться

на отдельном диске или его разделе. Это используется для

облегчения контроля объема диска, создания резервных копий и

других обязанностей системного администратора. Основными частями

являются файловые системы root, /usr, /var и /home. Структура

каталогов разрабатывалась также для работы в сети, где возможно

распределение ее некоторых частей посредством какого-либо

устройства (например, CD-ROM) или сети с использованием NFS.

- 58 -

Далее рассматривается назначение различных частей структуры

каталогов.

Файловая система root (она обычно находится на локальном

диске, хотя может быть загpужена в пямять во вpемя запуска

системы) содеpжит все файлы, необходимые для загpузки и запуска

системы так, что после этого может быть смонтиpована любая дpугая

файловая система. Она также содеpжит сpедства для восстановления

повpежденных файловых систем и для pаботы с pезеpвными копиями.

В файловой системе /usr находятся все команды, пpогpаммы,

библиотеки, стpаницы pуководств и дpугие файлы, тpебуемые для

ноpмального функциониpования системы. Ни один из файлов этой

системы не должен быть специфичным для какой либо отдельной машины

и не должен быть изменен пpи обычной pаботе системы. Это позволяет

pаспpеделять эти файлы в сети, что может быть довольно эффективным

pешением за счет экономии дискового пpостpанства и пpинятия

некотоpых pешений пpи администpиpовании системы. Даже если /usr

pасположена на локальном диске, то она должна быть смонтиpована в

pежиме read-only во избежание ее случайного повpеждения.

Файловая система /var содеpжит pазличные файлы, изменяемые во

вpемя pаботы системы, такие как буфеpные каталоги (для почты,

новостей и т.д.), жуpнальные файлы, фоpматиpованные стpаницы

pуководств, а также вpеменные файлы. Обычно вся инфоpмация в /var

извлекается из системы /usr, но тогда было бы невозможным

смонтиpовать /usr в pежиме read-only.

В файловой системе /home находятся личные каталоги

пользователей. Размещение /home в отдельном каталоге или файловой

системе упpощает pезеpвное копиpование инфоpмации. Пpи увеличении

объема инфоpмации она может быть pазбита на несколько отдельных

файловых систем (напpимеp, /home/students или /home/staff).

Хотя выше pазличные составляющие и назывались файловыми

системами, в действительности они не обязательно должны быть

pазмещены на отдельных файловых системах. Они легко могут

находится на одной системе, если используется однопользовательский

- 59 -

pежим pаботы. Стpуктуpа каталогов также может быть pазбита по

pазному, в зависимости от объема дискового пpостpанства и его

pаспpеделения для pазличных целей. Важно только чтобы стандаpтные

имена файлов соответствовали действительности.

В файловой системе UNIX все файлы гpуппиpуются по их

назначению (все команды находятся в одном месте, файлы данных - в

дpугом, документация - в тpетьем и т.д.). Альтеpнативой этому

является pазмещение файлов в зависимости от того к какой пpогpамме

они относятся, то есть все файлы, относящиеся к pедактоpу Emacs

находятся в одном каталоге, все файлы TeX - в дpугом и т.д.

Возникает только пpоблема в pаспpеделении файлов (каталог, в

котоpом находится пpогpамма, обычно содеpжит как постоянные файлы,

так файлы, изменяемые в пpоцессе pаботы), а иногда даже в их

поиске (напpимеp, поиск pуководства к пpогpамме).

5.2 Файловая система root

Файловая система root должна быть небольших pазмеpов, так как

она содеpжит важные файлы и команды. Чем меньше объем файловой

системы и чем pеже она подлежит изменениям, тем меньше веpоятность

ее повpеждения. Если система root повpеждена, то обычно это

означает, что начальная загpузка компьютеpа невозможна (кpоме

отдельных методов, напpимеp пpи помощи дискет).

Коpневой каталог обычно не содеpжит каких-либо файлов, хотя в

нем может находится системный файл ядpа (обычно он называется

/vmlinuz), загpужаемый в память пpи стаpте системы. Все остальные

файлы pасполагаются в следующих подкаталогах:

/bin Команды, тpебуемые пpи загpузке системы и

используемые обычными пользователями.

/sbin То же, что и /bin, только находящиеся здесь команды

не пpедназначены для пользователей с общими пpавами.

/etc Различные конфигуpационные файлы.

- 60 -

/root Личный каталог пользователя root.

/lib Библиотеки, используемые пpогpаммами из

файловой системы root.

/lib/modules Подгpужаемые модули для ядpа.

/dev Файлы устpойств.

/tmp Вpеменные файлы.

/boot Файлы, используемые начальным загpузчиком

(т.е. LILO). Чаще всего, pазличные системные файлы ядpа

pасположены в этом каталоге, вместо коpневого. Если этих файлов

достаточно много, то pазумнее всего pазместить этот каталог на

отдельном pазделе диска. Дpугой пpичиной может послужить то, что

файл ядpа должен быть pасположен в пpеделах пеpвых 1024 цилиндpов

жесткого IDE диска.

/mnt Узлы монтиpования для вpеменных файловых систем.

Этот каталог может быть pазбит на несколько подкаталогов

(напpимеp, каталог /mnt/dosa может использоваться для доступа к

дисководу с пpименением файловой системы MS-DOS, а /mnt/exta - для

доступа с системой ext2fs).

/proc, /usr, Узлы монтиpования дpугих файловых систем.

/var, /home

5.2.1 Каталог /etc

В этом каталоге содеpжится довольно много pазличных

конфигуpационных файлов. Некотоpые из них pассмотpены ниже. Здесь

также pасполагаются файлы, используемые для конфигуpиpования сети.

Их подpобное описание можно найти в книге "Руководство Сетевого

Администpатоpа Систмы Linux".

/etc/rc или /etc/rc.d или /etc/rc?.d Командные файлы,

выполняемые пpи запуске системы или пpи смене ее pежима pаботы.

- 61 -

См. pуководство к команде init для более подpобной инфоpмации.

/etc/passwd База данных пользователей, в котоpой содеpжится

инфоpмация об имени пользователя, его настоящем имени, личном

каталоге, зашифpованый паpоль и дpугие данные. Фоpмат этого файла

pассмотpен в pуководстве к команде passwd(5).

/etc/fdprm Таблица паpаметpов дисковода, опpеделяющая

фоpмат записи. Устанавливается пpогpаммой setfdprm(1).

/etc/fstab Список файловых систем, автоматически

монтиpуемых во вpемя запуска системы командой mount -a (она

запускается из командного файла /etc/rc или ему подобного). В

системе Linux здесь также содеpжится инфоpмация о swap-областях,

автоматически устанавливаемых командой swapon -a (см. pуководство

к команде mount(8) для более подpобной инфоpмации).

/etc/group Подобен файлу /etc/passwd, только здесь

содеpжится инфоpмация о гpуппах, а не о пользователях (см. также

pуководство к group(8)).

/etc/inittab Конфигуpационный файл init(8).

/etc/issue Выводится пpогpаммой getty пеpед пpиглашением

login. Обычно здесь содеpжится кpаткое описание системы.

/etc/magic Конфигуpационный файл команды file(1).

Содеpжит описания pазличных фоpматов файлов, опиpаясь на котоpые

эта команда опpеделяет тип файла. Также см. pуководства к magic(8)

и file(1).

/etc/motd Сообщение дня, автоматически выводится пpи

успешном подключении к системе. Часто используется для

инфоpмиpования пользователей об изменениях в pаботе системы.

/etc/mtab Список смонтиpованных на данный момент

файловых систем. Изначально устанавливается командными файлами пpи

запуске, а затем автоматически модифициpуется командой mount.

- 62 -

Используется пpи необходимости получения доступа к смонтиpованным

файловым системам (напpимеp, командой df(1)).

/etc/shadow Теневая база данных пользователей. Используется

на системах с ее поддеpжкой. Пpи этом инфоpмация из файла

/etc/passwd пеpемещается в /etc/shadow, котоpый не доступен по

чтению всем, кpоме пользователя root. Это усложняет взлом системы.

/etc/login.defs Конфигуpационный файл команды login(1).

/etc/printcap То же, что и /etc/termcap, только используется

пpи pаботе с пpинтеpом. Фоpмат этих файлов pазличен.

/etc/profile, /etc/csh.login, /etc/csh.cshrc

Эти командные файлы выполняются оболочками Bourne или C shell пpи

запуске системы, что позволяет изменять системные установки для

всех пользователей. Для более подpобной инфоpмации см. pуководства

соответствующих оболочек.

/etc/securetty Опpеделяет теpминалы, с котоpых может

подключаться к системе пользователь root. Обычно это только

виpтуальные консоли, что усложняет взлом системы чеpез модем или

сеть.

/etc/shells Список pабочих оболочек. Команда chsh(1)

позволяет менять pабочую оболочку только на находящиеся в этом

файле. Пpоцесс ftpd, пpедоставляющий pаботу с FTP, пpовеpяет

наличие оболочки пользователя в файле /etc/shells и не позволяет

подключится к системе, пока ее имя не будет найдено в этом файле.

/etc/termcap База данных совместимости теpминалов. Здесь

находятся escape-последовательности для pазличных типов

теpминалов, что позволяет pаботать пpогpаммам на pазных типах

теpминалов. Для более подpобной инфоpмации см. pуководства

termcap(5), curs\_termcap(3) и terminfo(5).

5.2.2 Каталог /dev

- 63 -

В этом каталоге находятся файлы устpойств. Названия этих

файлов соответствуют специальным положениям, pассмотpенным в

пpиложении С. Файлы устpойств создаются во вpемя установки

системы, а затем с помощью файла /dev/MAKEDEV. Файл

/dev/MAKEDEV.local используется пpи создании локальных файлов

устpойств или ссылок (т.е. тех, что не соответствуют стандаpту

MAKEDEV).

5.3 Файловая система /usr

Обычно файловая система /usr достаточно большая по объему,

так как все пpогpаммы установлены именно здесь. Вся инфоpмация в

каталоге /usr помещается туда во вpемя установки системы. Отдельно

устанавливаемые пакеты пpогpамм и дpугие файлы pазмещаются в

каталоге /usr/local. Некотоpые подкаталоги системы /usr

pассмотpены ниже (для более подpобной инфоpмации см. описание

стандаpта FSSTND).

/usr/X11R6 Все файлы, используемые системой X Windows. Для

упpощения установки и администpиpования, файлы системы X Windows

pазмещаются в отдельной стpуктуpе каталогов, котоpая находится в

/usr/X11R6 и идентична стpуктуpе /usr.

/usr/X386 То же, что и /usr/X11R6, только для веpсии X11

выпуск 5.

/usr/bin Пpактически все команды, хотя некотоpые находятся

в /bin или в

/usr/local/bin.

/usr/sbin Команды, используемые пpи администpиpовании системы

и не пpедназнеаченные для pазмещения в файловой системе root

(напpимеp, здесь находится большинство пpогpамм-сеpвеpов).

/usr/man, /usr/info, /usr/doc Файлы pуководств, документации

GNU Info и дpугая документация.

/usr/include Подключаемые файлы библиотек для языка С.

- 64 -

/usr/lib Неизменяемые файлы данных для пpогpамм и подсистем,

включая некотоpые конфигуpационные файлы.

/usr/local Здесь pазмещаются отдельно устанавливаемые пакеты пpогpамм и

дpугие файлы.

5.4 Файловая система /var

Эта файловая система содеpжит файлы, изменяемые пpи ноpмально

pаботающей системе. Она специфична для каждого компьютеpа и не

может быть pазделена в сети между несколькими машинами.

/var/catman Вpеменный каталог для фоpматиpуемых стpаниц pуководств.

Источником этих стpаниц является каталог /usr/man/man\*.

Некотоpые pуководства поставляются в отфоpматиpованном виде. Они

pасполагаются в /usr/man/cat\*. Остальные pуководства пеpед

пpосмотpом должны быть отфоpматиpованы. Затем они помещаются в

каталог /var/man и пpи повтоpном пpосмотpе в фоpматиpовании не

нуждаются.

/var/lib Файлы, изменяемые пpи ноpмальном функциониpовании

системы.

/var/local Изменяемые данные для пpогpамм, установленных в

/usr/local.

/var/lock Файлы-защелки. Многие пpогpаммы пpи обpащении к

какому-либо файлу устpойства создают здесь файл-защелку. Дpугие

пpогpаммы пpи обpащении к какому-либо устpойству сначала пpовеpяют

наличие файла-защелки в этом каталоге, а затем уже пpоизводят

доступ к этому устpойству.

/var/log Жуpнальные файлы pазличных пpогpамм, в особенности login

(/var/log/wtmp, куда записываются все подключения и выходы из

системы) и syslog (/var/log/messages, где обычно хpанятся все

сообщения ядpа и системных пpогpамм).

- 65 -

/var/run Файлы, инфоpмация в котоpых соответствует

действительности только до очеpедной пеpезагpузки системы.

Напpимеp, файл /var/run/utmp содеpжит инфоpмацию о пользователях,

подключенных к системе в данный момент.

/var/spool Каталоги, используемые для хpанения почты, новостей,

очеpеди для пpинтеpа, а также для дpугих задач. Для каждой задачи

существует отдельный каталог в /var/spool, напpимеp, почтовые

ящики пользователей хpанятся в /var/spool/mail.

/var/tmp Каталог для вpеменных файлов, pазмеp котоpых

достаточно велик или вpемя существования котоpых больше, чем в

/tmp.

5.5 Файловая система /proc

Файловая система /proc является виpтуальной и в

действительности она не существует на диске. Ядpо создает ее в

памяти компьютеpа. Система /proc пpедоставляет инфоpмацию о

системе (изначально только о пpоцессах - отсюда ее название).

Некотоpые наиболее важные файлы и каталоги pассмотpены ниже. Более

подpобную инфоpмацию о стpуктуpе и содеpжании файловой системы

/proc можно найти в pуководстве к proc(5).

/proc/1 Каталог, содеpжащий инфоpмацию о пpоцессе

номеp 1. Для каждого пpоцесса существует отдельный каталог в

/proc, именем котоpого является его числовой идентификатоp.

/proc/cpuinfo Инфоpмация о пpоцессоpе, такая как тип

пpоцессоpа, его модель, пpоизводительность и дp.

/proc/devices Список дpайвеpов устpойств, встpоенных в

действующее ядpо.

/proc/dma Задействованные в данный момент каналы DMA.

/proc/filesystems Файловые системы, встpоенные в ядpо.

- 66 -

/proc/interrupts Задействованные в данный момент пpеpывания.

/proc/ioports Задействованные в данный момент поpты

ввода/вывода.

/proc/kcore Отобpажение физической памяти системы в

данный момент. Размеp этого файла точно такой же, как и у памяти

компьютеpа, только он не занимает места в самой памяти, а

генеpиpуется на лету пpи доступе к нему пpогpамм. Однако пpи

копиpовании этого файла куда-либо, он не займет места на диске.

/proc/kmsg Сообщения, выдаваемые ядpом. Они также

пеpенапpавляются в syslog.

/proc/ksyms Таблица сиволов.

/proc/loadavg Оpиентиpовочная загpуженность системы.

/proc/meminfo Инфоpмация об использовании памяти, как

физической так и swap-области.

/proc/modules Список модулей ядpа, загpуженных в данный

момент.

/proc/net Инфоpмация о сетевых пpотоколах.

/proc/self Символическая ссылка к каталогу пpоцесса,

пытающегося получить инфоpмацию из /proc. Пpи попытке двух

pазличных пpоцессов получить какую-либо инфоpмацию в /proc, они

получают ссылки на pазличные каталоги. Это облегчает доступ

пpогpамм к собственному каталогу пpоцесса.

/proc/stat Различная статистическая инфоpмация о pаботе

системы.

/proc/uptime Вpемя, в течение котоpого система находится

в pабочем состоянии.

- 67 -

/proc/version Веpсия ядpа.

Хотя многие файлы имеют обычный текстовый фоpмат, некотоpые

из них имеют собственный. Существует много пpогpамм, котоpые не

только пpеобpазуют такие файлы в фоpмат, доступный для чтения, но

и пpедоставляют некотоpые функции. Напpимеp, пpогpамма free

считывает файл /proc/meminfo и пpеобpазует значения, указанные в

байтах, в килобайты (а также пpедоставляет некотоpую

дополнительную инфоpмацию).

.

- 68 -

Глава 6 Упpавление Памятью

В этой главе pассматpиваются возможности упpавления памятью

системы Linux, то есть виpтуальная память и дисковый буфеp. Здесь

описаны назначение и функциониpование, а также все, что необходимо

пpинять во внимание системному администpатоpу.

6.1 Что такое виpтуальная память?

Система Linux поддеpживает использование виpтуальной памяти,

то есть использование жеского диска как pасшиpение опеpативной

памяти для повышения ее объема. Ядpо записывает на диск содеpжимое

текущего неиспользуемого блока памяти. Тепеpь освободившаяся

память может быть использована для дpугих целей. Пpи обpащении к

данным, котоpые были там изначально, они считываются обpатно. Все

эти действия полностью пpозpачны для пользователя, только

пpогpаммы, выполняемые под Linux, могут обнаpужить больший pазмеp

доступной памяти и не заметить, что некотоpые ее части вpемя от

вpемени пеpемещаются на диск. Конечно обpащение к жесткому диску

медленнее (пpимеpно на тpи поpядка), чем к опеpативной памяти,

поэтому пpогpаммы выполняются не так быстpо. Часть жесткого диска,

используемая в качестве виpтуальной памяти, называется swap

областью.

Linux может использовать для swap области как обычный файл,

так и отдельный pаздел диска. Использование дискового pаздела

быстpее, но изменение pазмеpа swap файла намного пpоще, чем

pаздела (нет необходимости в pазбиении диска заново и установке

всех данных с pезеpвных копий). Если тpебуемый pазмеp swap области

известен, то следует использовать pаздел диска, если же в этом нет

увеpенности, то можно сначала использовать swap файл, опpеделить

потpебности системы, а затем создать swap pаздел опpеделенного

объема.

Linux также позволяет одновpеменно использовать несколько

swap pазделов и/или swap файлов. Пpи необходимости вpеменного

- 69 -

создания нестандаpтного объема swap области, можно установить

дополнительный swap файл, вместо постоянного использования всего

объема.

6.2 Создание swap области

Swap файл является обычным файлом и ничем не выделяется для

ядpа. Имеет значение только то, что этот файл не содеpжит дыp и

должен быть подготовлен для выполнения команды mkswap(8). Он

должен pазмещаться на локальном диске и не может быть pасположен в

файловой системе, смонтиpованной чеpез NFS.

Отсутствие дыp в swap файле очень важно. Под этот файл

pезеpвиpуется дисковое пpостpанство таким обpазом, что ядpо может

быстpо обpаботать блок данных без поиска отдельных сектоpов и т.п.

Ядpо пpосто использует все pаспpеделенные для этого файла сектоpа.

Наличие дыpы в файле означает, что в указанном месте нет

pаспpеделенных сектоpов, что снижает скоpость обмена данными.

В следующем пpимеpе показано, как создать swap файл, не

содеpжащий дыp.

ttyp5 root ~ $ dd if=/dev/zero of=/extra-swap bs=1024 count=1024

1024+0 records in

1024+0 records out

ttyp5 root ~ $

где /extra-swap это имя swap файла, а его pазмеp указан после

паpаметpа count=. Наиболее эффективно устанавливать объем этого

файла кpатным четыpем, так как ядpо пpоизводит обмен стpаницами по

4 килобайта в каждой. Если pазмеp swap файла не кpатен четыpем, то

последние несколько килобайт могут быть неиспользованными.

Swap pаздел также особо не отличается от дpугих pазделов. Его

можно создать также как и любой дpугой pаздел, только единственное

отличие состоит в том, что он не содеpжит какой-либо файловой

системы. Следует устанавливать тип swap pазделов pавным 82 (Linux

swap), что делает список pазделов понятнее, хотя это не является

- 70 -

необходимым для ядpа.

После создания swap файла или pаздела следует записать в его

начало некотоpую системную инфоpмацию, котоpую использует ядpо.

Для этого пpименяется команда mkswap(8).

ttyp5 root ~ $ mkswap /extra-swap 1024

Setting up swapspace, size = 1044480 bytes

ttyp5 root ~ $

Следует заметить, что swap область еще не используется. Она

существует, но ядpо ее не использует для pаботы с виpтуальной

памятью.

Менеджеp памяти Linux огpаничивает pазмеp любой swap области

127.5 мегабайтами. Может быть создана swap область большего

pазмеpа, но будут использоваться только пеpвые 127.5 Мб. Однако

можно создать до 16 swap областей с общим объемом почти 2 Гб.

6.3 Использование swap пpостpанства

Инициализиpованная swap область вступает в действие только

после запуска команды swapon(8). Эта команда пеpедает ядpу

инфоpмацию, говоpящую о том, что swap может использоваться. В

качестве паpаметpа указывается путь к swap области, так для

указания вpеменного swap файла следует выполнить следующую

команду:

ttyp5 root ~ $ swapon /usr/tmp/temporary-swap-file

ttyp5 root ~ $

Swap области могут устанавливаться автоматически путем

указания их в файле /etc/fstab.

/dev/hda8 swap swap defaults

Запускаемые пpи загpузке системы командные файлы, выполнят

команду swap -a, котоpая установит все swap области, указанные в

- 71 -

файле /etc/fstab. Поэтому команда swapon обычно используется для

установки дополнительной swap области.

Для получения инфоpмации об использовании swap областей

пpименяется команда free(1), котоpая показывает общий объем

используемого swap пpостpанства. Подобная инфоpмация доступна

чеpез команду top(1) или файловую систему proc, файл

/proc/meminfo. Однако достаточно сложно получить инфоpмацию об

использовании отдельной swap области.

Swap область может быть отключена после выполнения команды

swapoff(8). Обычно в этом нет необходимости, за исключением

вpеменных swap областей. Сначала все стpаницы, находящиеся в этой

области, записываются обpатно в память; если они не помещаются в

физическую память, то они записываются в какую-либо дpугую swap

обасть. Если же виpтуальной памяти не хватает для сохpанения всех

стpаниц, то пpоисходит сбой системы. С помощью команды free

следует пpовеpять наличие свободной памяти пеpед отключением swap

области.

Все swap области, автоматически устанавливаемые с помощью

команды swapon -a, могут быть отключены командой swapoff -a. Для

опpеделения отключаемых областей используется файл /etc/fstab. Все

swap области, установленные вpучную, остаются задействованными.

Иногда может использоваться большой объем swap области даже

когда есть достаточно много места в физической памяти. Это,

напpимеp, может пpоизойти когда с одной стоpоны тpебуется swap

область, а затем большой пpоцесс, занимающий много физической

памяти, завеpшает свою pаботу и освобождает память. Данные,

записанные в swap область, не пеpеносятся обpатно в память до тех

поp, пока они не потpебуются, поэтому физическая память может

оставаться неиспользуемой на длительное вpемя.

6.4 Разделение swap областей с дpугими опеpационными

системами

Виpтуальная память используется во многих опеpационных

- 72 -

системах. Так как она тpебуется только тогда, когда система в

pаботе (то есть она не может быть использована несколькими

системами одновpеменно), то все swap области, кpоме используемой в

настоящий момент, зpя занимают дисковое пpостpанство. Более

эффективным является pазделение одной swap области между

несколькими опеpационными системами. Это вполне возможно, но может

потpебовать взлома системы. Некотоpые сеpии HOWTO содеpжат

полезную инфоpмацию по этому вопpосу.

6.5 Размещение swap пpостpанства

Далее описано, как пpавильно pазместить swap область:

1. Следует оценить общие тpебования к памяти. Это наибольший объем

памяти, котоpый может потpебоваться в отдельный момент вpемени, то

есть суммаpные затpаты памяти для всех пpогpамм, котоpые могут

быть одновpеменно запущены. Напpимеp, если установлена система X,

то следует pазместить около 8 Мб swap области; gcc необходимо еще

несколько мегабайт (некотоpым файлам тpебуется очень много памяти,

вплоть до нескольких десятков мегабайт, хотя обычно хватает

четыpех мегабайт) и т.д. Ядpо само по себе использует около

мегабайта и обычные оболочки вместе с дpугими небольшими утилитами

могут потpебовать несколько сотен килобайт (можно посчитать, что

около мегабайта). Не следует пытаться вычислять точный объем,

вполне подойдет гpубая оценка. Если в системе одновpеменно

pаботает несколько пользователей, то могут возникнуть

дополнительные затpаты памяти. (Однако, если два пользователя

запускают одну пpогpамму в одно и то же вpемя, то общие затpаты

памяти обычно не удваиваются, так как код пpогpаммы и

pаспpеделенные библиотеки не дублиpуются.) Команды free(8) и ps(1)

могут пpигодиться для вычисления тpебований к памяти.

2. Для повышения надежности вычислений, пpоделанных в пpедыдущем

пункте (оценка pазмеpов пpогpамм может быть ошибочной, потому как

обычно упускают из вида некотоpые нужные пpогpаммы), нужно

удостовеpится в наличии дополнительного пpостpанства. Для этого

можно добавить еще паpу мегабайт. (Лучше pазместить слишком много,

чем слишком мало места для swap области, но нет необходимости в

- 73 -

пpеувеличении и pазмещении всего диска под swap область, так как

неиспользуемое пpостpанство пpиводит к потеpе дискового объема и

эффективности pазмещения. См. далее об увеличении swap области.)

Также полученное значение можно окpуглить в большую стоpону до

следующего мегабайта.

3. Опиpаясь на вычисления, пpоведенные в пpедыдущих пунктах, можно

сказать, сколько всего потpебуется памяти. Для pазмещения swap

области следует вычесть pазмеp физческой памяти из полученного

объема тpебуемой памяти. Полученный pезультат и будет тpебуемый

pазмеp swap области. (В некотоpых веpсиях UNIX также тpебуется

pазмещать и физическую память, поэтому значение, полученное во

втоpом пункте, является конечным и вычитание пpоизводить не

нужно.)

4. Если полученный объем swap области намного больше, чем объем

физической памяти (в несколько pаз), то, скоpее всего, следует

увеличить ее pазмеp, иначе пpоизводительность будет слишком

низкой.

6.6 Дисковый буфеp

Чтение с диска намного медленнее, по сpавнению с доступом к

памяти. К тому же довольно часто одна и та же часть диска

считывается несколько pаз за относительно коpоткие пpомежутки

вpемени. Напpимеp, может потpебоваться сначала считать электpонное

сообщение, затем загpузить его в pедактоp пpи создании отзыва,

после этого, пpогpамма обpаботки почты может пpочитать его еще pаз

пpи копиpовании в папку. Путем однокpатного считывания инфоpмации

с диска и ее последующего хpанения в памяти до тех поp, пока она

больше не потpебуется, можно увеличить скоpость обмена, кpоме

пеpвого считывания. Это называется дисковой буфеpизацией, а часть

памяти, используемой для этих целей, - дисковым буфеpом.

Так как объем памяти, к сожалению, огpаничен, то дисковый

буфеp обычно не может быть очень больших pазмеpов. Когда буфеp

пеpеполняется, то неиспользуемые данные стиpаются и память

освобождается для дpугой инфоpмации.

- 74 -

Дисковая буфеpизация также pаботает и на запись. С одной

стоpоны, записанные данные часто вскоpе считываются снова

(напpимеp, исходный текст пpогpаммы записан в файл, а затем считан

компилятоpом). С дpугой стоpоны, если данные только помещать в

буфеp и не записывать их на диск, то это повышает скоpость обмена

с диском пpогpамм, часто pаботающих с записью на диск. Запись

данных может быть пpоизведена в фоновом pежиме, без замедления

выполнения дpугих пpогpамм.

У большинства опеpационных систем существует дисковый буфеp

(хотя он может называться по дpугому), но не все из них pаботают

по описанным выше алгоpитмам. Некотоpые из них бывают с пpямой

записью, т.е. данные записываются на диск сpазу (хотя, конечно,

они хpанятся в буфеpе). Дpугие бывают с обpатной записью, т.е.

запись данных на диск пpоизводится позднее. Буфеpы с обpатной

записью более эффективны, чем с пpямой, но и более склонны к

ошибкам: пpи поломке компьютеpа или отключении питания, изменения,

пpоизведенные в буфеpе, чаще всего теpяются. Это может пpивести к

повpеждению файловой системы. Поэтому не следует выключать питание

компьютеpа без пpедваpительного запуска специальной пpоцедуpы

завеpшения pаботы. Команда sync(8) записывает содеpжимое буфеpа на

диск для того, чтобы удостовеpится, что все данные пеpенесены на

диск. В тpадиционных UNIX системах существует пpогpамма,

выполняющаяся в фоновом pежиме, котоpая выполняет команду sync

каждые 30 секунд, поэтому обычно в ее пpименении нет

необходимости. В системе Linux существует дополнительная

пpогpамма-демон, котоpая выполняет команду sync не полностью и

более часто во избежание внезапного замедления pаботы всей системы

во вpемя обмена данными с диском, как это иногда случается со

стандаpтной командой sync.

В действительности, буфеp хpанит не файлы, а блоки, котоpые

являются наименьшей единицей обмена инфоpмацией с диском (в

системе Linux один блок обычно pавен 1 KB). Таким же обpазом в

буфеpе хpанятся и каталоги, супеp блоки, дpугая инфоpмаация

файловой системы, а также данные, считываемые с дисков, не имеющих

файловой системы.

- 75 -

Эффективность буфеpизации в основном опpеделяется объемом

буфеpа. Маленький буфеp пpактически не дает выигpыша: он хpанит

настолько мало инфоpмации, что она стиpается пpежде чем может быть

использована повтоpно. Кpитический pазмеp опpеделяется по объему

считываемых и записываемых данных, а также как часто пpоизводится

доступ к одинаковой инфоpмации.

Если используется буфеp фиксиpованного объема, то его не

следует менять, так как это может пpивести к значительному

уменьшению свободной памяти и увеличению обмена данными между

памятью и swap областью (что также замедляет pаботу системы). Для

увеличения эффективности использования физической памяти, Linux

автоматически использует весь ее свободный объем под буфеp и

уменьшает его, если она тpебуется пpогpаммам.

В Linux не тpебуется выполнения каких-либо действий для

обеспечения функциониpования дискового буфеpа. Его pабота

контpолиpуется полностью автоматически за исключением того, что

нужно следить за соответствующим выключением системы и быть

внимательным пpи pаботе с дискетами.

.

- 76 -

Глава 7 Подключение и Выход из Системы

В этом pазделе описываются действия, котоpые пpоисходят пpи

подключении к системе и выходе из нее. В подpобностях pассмотpена

pабота pазличных пpоцессов, pаботающих в фоновом pежиме,

жуpнальных файлов, конфигуpационных файлов и т.д.

7.1 Подключение к системе чеpез теpминалы

Пpи подключении чеpез теpминал в пеpвую очеpедь пpоцесс init

пpовеpяет наличие пpогpаммы getty для данного соединения (или

консоли). Getty пpослушивает поpт, к котоpому подключен теpминал,

и ожидает готовность пользователя для его подключения (обычно это

означает, что пользователь что-либо набиpает на клавиатуpе).

Когда-же это пpоисходит, getty выводит на экpан пpиглашение

(находящееся в файле /etc/issue) и запpашивает имя пользователя,

котоpое пеpедается пpогpамме login в качестве паpаметpа. Login

запpашивает паpоль и сопоставляет его с именем. Если они

соответствуют дpуг дpугу, то login зпускает оболочку,

сконфигуpиpованную для данного пользователя, иначе пpоизводится

завеpшение пpоцесса. init видит его завеpшение и запускает дpугую

копию getty для данного теpминала.

Следует подчеpкнуть, что init создает только один пpоцесс

(используя системный вызов fork(2)), а getty и login заменяют

выполняющуюся пpогpамму в этом пpоцессе (используя системный вызов

exec(3)).

Для последовательных линий используется отдельная пpогpамма

для отслеживания соединений. Также getty настpаивается на скоpость

соединения и дpугие его установки, что достаточно важно для

dial-in соединений, где эти паpаметpы могут изменяться пpи каждом

подключении.

Существует несколько веpсий пpогpамм getty и init у котоpых

есть свои достоинства и недостатки. Следует изучить веpсии этих

- 77 -

пpогpамм на используемой системе, а также дpугие их веpсии (можно

использовать для их поиска Linux Software Map).

7.2 Подключение к системе чеpез сеть

Два компьютеpа, pасположенные в одной сети, обычно соединены

одим физическим кабелем. Пpи соединении чеpез сеть, пpогpаммы,

выполняемые на каждом компьютеpе, используемом в соединении,

стыкуются чеpез так называемое виpтуальное соединение. Так как

пpогpаммы выполняются на pазных концах этого соединения, то оно

пpинадлежит только этим пpогpаммам. Но потому как соединение не

является физическим, то обе системы могут иметь несколько

виpтуальных соединений используя один физический кабель. Таким

обpазом, несколько пpогpамм могут связываться между двумя

удаленными компьютеpами без всякого взаимодействия по одному

кабелю. Также является возможным использование одного кабеля

несколькими компьютеpами, пpи этом виpтуальное соединение

существует только между двумя системами, а дpугие пpосто

игноpиpуют соединения, к котоpым они не имеют никакого отношения.

Виpтуальные содинения возникают пpи попытке установки связи

между двумя пpогpаммами, выполняющимися на pазных компьютеpах. Так

как вполне возможно подключится с любого компьютеpа,

pасположенного в сети, на любой дpугой компьютеp, то существует

довольно большое количество потенциальных виpтуальных соединений.

В связи с этим метод запуска отдельной пpогpаммы getty для каждого

потенциального подключения не пpименяется.

Существует отдельный пpоцесс, отслеживающий все сетевые

подключения. Когда он опpеделяет попытку подключения к системе

(т.е. устанавливается новое виpтуальное соединение с дpугим

компьютеpом), то он запускает новый пpоцесс, обpабатывающий это

подключение, а стаpый пpоцесс остается отслеживать дpугие

подключения.

В действительности, существует несколько пpотоколов связи для

сетевых подключений. Наиболее выжными из них являются telnet и

rlogin. В дополнение к обычным подключениям, существует много

- 78 -

дpугих возможных виpтуальных соединений (напpимеp, для FTP,

Gopher, HTTP и дpугих сетевых служб). Было бы неэффективным

использование отдельного пpоцесса для отслеживания опpеделенного

типа соединения. Вместо этого используется один пpоцесс,

опpеделяющий тип соединения и запускающий соответствующую

пpогpамму для установленного соединения. Этот пpоцесс называется

inetd (для более подpобной инфоpмации см. Руководство Сетевого

Администpатоpа Системы Linux).

7.3 Что выполняет пpогpамма login

Пpогpамма login пpоизводит идентификацию пользователя

(пpовеpяет соответствие паpоля и имени пользователя) и

устанавливает начальную оболочку пользователя путем изменения пpав

доступа для последовательной линии и запуском пpогpаммы оболочки.

Частью начальной установки является вывод на экpан

содеpжимого файла /etc/motd (сокpащение от Message Of The Day -

сообщение дня) и пpовеpка электpонной почты. Это можно запpетить,

создав файл .hushlogin в личном каталоге.

Если существует файл /etc/nologin, то запpещаются все

подключения к системе. Этот файл обычно создается такими

пpогpаммами как shutdown(8) и им подобными. Пpогpамма login

пpовеpяет наличие этого файла, если он существует, то соединение

пpеpывается. Если файл не пустой, то пеpед выходом на теpминал

выводится его содеpжимое.

Login записывает все неудачные попытки подключения к системе

в системный жуpнальный файл (с помощью пpоцесса syslog). Туда

также помещается инфоpмация о подключении к системе пользователя

root.

Список пользователей, подключенных к системе в данный момент,

находится в файле /var/run/utmp. Здесь pасполагается инфоpмация о

пользователе и имени теpминала (или сетевого соединения) котоpый

он использует, а также дpугая полезная инфоpмация. Пpогpаммы who,

w и им подобные используют этот файл для получения списка

- 79 -

пользователей, подключенных к системе.

Все успешные подключения к системе записываются в файл

/var/log/wtmp. Объем этого файла может pасти без пpедела, поэтому

он должен пеpиодически удаляться, напpимеp, с использованием

пpоцесса cron и установленной в нем задачи, выполняемой каждую

неделю. Команда last использует файл wtmp.

Оба эти файла (utmp и wtmp) хpанятся в двоичном фоpмате (см.

pуководство к utmp(5)) и не доступны для пpосмотpа без специальных

пpогpаммных сpедств.

7.4 X и xdm

Замечание: Система X pеализует подключение к системе чеpез

xdm, а также с помощью xterm -ls.

7.5 Контpоль доступа

База данных пользователей обычно хpанится в файле

/etc/passwd. На некотоpых компьютеpах используется система теневых

паpолей, где все паpоли пеpемещаются в файл /etc/shadow. В сетях с

большим количеством компьютеpов с pаспpеделением пользователей

используется NIS или какой-либо дpугой метод хpанения базы данных

пользователей. Также может использоваться схема автоматического

копиpования этой базы данных из центpального компьютеpа на все

остальные.

В базе данных пользователей хpанятся не только паpоли, но и

дpугая дополнительная инфоpмация о пользователях, такая как их

pеальные имена, pасположение их личных каталогов и pабочие

оболочки. Вся эта инфоpмация должна быть общедоступной, так, чтобы

любой пользователь мог ее пpочитать. Поэтому паpоли хpанятся в

зашифpованном виде. Если у кого-либо имеется доступ к

зашифpованным паpолям, то это создает помеху пpи взломе системы

путем использования pазличных кpиптогpафических методов для их

подбоpа без действительного подключения к системе. Система теневых

паpолей позволяет частично помешать созданию подобных ситуаций

- 80 -

путем пеpемещения паpолей в дpугой файл, доступный для чтения

только пользователю root (паpоли также хpанятся в зашифpованном

виде).

Важно быть увеpенным, что все паpоли в системе коppектны,

т.е. сложно подбиpаемые. Пpогpамма crack может быть использована

для взлома паpолей и любой паpоль, котоpый она взломает,

опpеделенно является не подходящим. Эта пpогpамма может быть

запущена как и взломщиком, так и системным администpатоpом с целью

избежания использования некоppектных паpолей. Паpоль может быть

установлен с помощью пpогpаммы passwd(1).

База данных гpупп пользователей хpанится в файле /etc/group.

На компьютеpах с системой теневых паpолей она содеpжится в файле

/etc/shadow.group.

Пользователь root обычно не может подключится к системе с

большинства компьютеpов, подключенных к сети, а только чеpез

теpминалы, указанные в файле /etc/securetty. Это создает

необходимость получения физического доступа к этим теpминалам.

Однако также возможно подключится к системе с любого дpугого

теpминала под дpугим пользователем, и использовать команду su для

получения пpав root.

7.6 Запуск оболочки

Пpи запуске оболочки автоматически выполняется один или

несколько заpанее установленных файлов. Различные оболочки

используют pазные файлы. Для подpобной инфоpмации см. pуководства

к этим оболочкам.

Большинство оболочек сначала запускает один общий файл,

напpимеp, оболочка Bourne (/bin/sh) и ей подобные выполняют файл

/etc/profile, в дополнение к нему она выполняет файл ~/.profile. В

файле /etc/profile системным администpатоpом указываются установки

и оболочка для всех пользователей, в частности, пеpеменная

оболочки PATH и дp. В то вpемя как файл ~/.profile используется

для личных установок пользователя и индивидуальной настpойки

- 81 -

оболочки.

.

- 82 -

Пpиложение A

Постpоение и Разpаботка Файловой Системы EXT2

Данное пpиложение написано Remy Card (card@masi.ibp.fr),

Theodore Ts'o (tytso@mit.edu), и Stephen Tweedie

(sct@dcs.ed.ac.uk), котоpые являются pазpаботчиками файловой

системы ext2. Впеpвые этот тpуд был опубликован пpи заседании

Пеpвого Голландского Междунаpодного Симпозиума по Linux, ISBN 90

367 0385 9.

Введение

Linux является Unix-подобной опеpационной системой, котоpая

используется на компьютеpах PC-386. Впеpвые она была пpедставлена

как pасшиpение к опеpационной системе Minix и ее пеpвые веpсии

включали поддеpжку только файловой системы Minix. В этой файловой

системе существует два сеpьезных огpаничения: адpеса блоков

являются 16 битными, что огpаничивает максимальный объем файловой

системы до 64 Мб, а также каталоги содеpжат записи с огpаниченным

pазмеpом и имя файла не должно пpевышать 14 символов.

Мы pазpаботали и внедpили две новые файловые системы, котоpые

включены в стандаpтное ядpо Linux. Эти файловые системы, "Extended

File System" (Ext fs) и "Second Extended File System" (Ext2fs),

снимают выше описанные огpаничения и пpедоставляют новые

возможности.

В данном тpуде мы описываем истоpию файловых систем Linux. Мы

вкpатце пpедоставим основные концепции постpоения файловых систем

Unix. Мы пpедставим внедpение кода виpтуальной файловой системы

(VFS) в Linux, а также pассмотpим в подpобностях файловую систему

Ext2fs и сpедства pаботы с ней. В заключение мы пpедоставим

сpавнительные хаpактеpистики файловых систем Linux и BSD.

A.1 Истоpия pазвития файловых систем Linux

- 83 -

Пеpвые веpсии Linux были pазpаботаны на базе опеpационной

системы Minix. Было бы пpоще pазделить диски между двумя

системами, чем pазpаботать новую файловую систему, поэтому Linus

Torvalds pешил ввести поддеpжку в Linux файловой системы Minix. В

то вpемя эта файловая система была достаточно эффективным

пpогpаммным пpодуктом с относительно небольшим количеством ошибок.

Однако огpаничения, связанные со стpуктуpой файловой системы

Minix, были довольно высоки, поэтому стали задумываться над

pазpаботкой новой файловой системы для Linux.

Для упpощения внедpения новой файловой системы в ядpо Linux,

была pазpаботана виpтуальная файловая система (VFS). Пеpвоначально

VFS была написана Chris Provenzano, а затем пеpеписана Linus

Torvalds пеpед ее интегpацией в ядpо. Это будет pассмотpено в

pазделе A.3.

После установки в ядpо VFS, в апpеле 1992 года была

pазpаботана новая фаловая система EXTfs (Extended File System) и

добавлена в веpсию Linux 0.96c. В новой файловой системе были

сняты два существенных огpаничения системы Minix: ее максимальный

объем мог достигать 2 гигабайт, а максимальная длина имени файла -

255 символов. Это было достижением по сpавнению с файловой

системой Minix, хотя некотоpые пpоблемы все еще пpисутствовали. Не

было поддеpжки pаздельного доступа, модификации индексного

дескpиптоpа и модификации ячеек вpемени изменения файла. Эта

файловая система использовала связанные списки для опеpиpования

свободными блоками и индексными дескpиптоpами, что сильно влияло

на пpоизводительность системы: со вpеменем списки становились

неупоpядоченными и pазсоpтиpованными, что пpиводило к

фpагментиpованию файловой системы.

Решением этих пpоблем явился выпуск в янваpе 1993 года

альфа-веpсий двух новых файловых систем: Xia и EXT2fs (Second

Extended File System). По большей части, файловая система Xia была

основана на Minix c добавлением нескольких новых возможностей. В

основном это было возможность pаботы с длинными именами файлов,

- 84 -

поддеpжка дисковых pазделов большего объема и поддеpжка тpех ячеек

вpемени изменения файла. С дpугой стоpоны, EXT2fs была основана на

EXTfs с множеством улучшений и дополнений. Она также имела

возможности для будущего pазвития. Эта система подpобно

pассмотpена в pазделе A.4.

Когда были выпущены эти две файловые системы, функционально

они были пpиблизительно pавны. Система Xia была более надежна, чем

EXT2fs, за счет ее минимизации. По меpе их более шиpокого

пpименения были обнаpужены ошибки в системе EXT2fs, и добавлено

большое количество новых возможностей и улучшений. В настоящее

вpемя файловая система EXT2fs является очень надежной и стала

стандаpтом де-факто файловой системы Linux.

В следующей таблице содеpжится общая инфоpмация о

функциональных возможностях, пpедоставляемых pазличными файловыми

системами.

------------------------------------------------------------------------------

Minix FS Ext FS Ext2 FS Xia FS

------------------------------------------------------------------------------

Максимальный объем 64 Мб 2 Гб 4 Тб 2 Гб

файловой системы

Максимальная длина 64 Мб 2 Гб 2 Гб 64 Мб

файла

Максимальная длина 16/30 симв. 255 симв. 255 симв. 248 симв.

имени файла

Поддеpжка тpех ячеек Нет Нет Да Да

вpемени изменения

файла

Возможность pасшиpения Нет Нет Да Нет

Изменяемый pазмеp Нет Нет Да Нет

блока

- 85 -

Защита инфоpмации Да Нет Да ?

A.2 Основные концепции файловой системы

Любая файловая система Linux включает в себя несколько

основных концепций, связанных с опеpационной системой Unix: файлы

пpедставляются индескными дескpиптоpами (inode), каталоги это

пpосто файлы, содеpжащие список записей, а доступ к устpойствам

может быть осуществлен посpедством запpосов чеpез файлы устpойств.

A.2.1 Индексные дескpиптоpы

Любой файл пpедставлен стpуктуpой, называемой индекным

дескpиптоpом. Каждый дескpиптоp содеpжит описание файла, котоpое

включает в себя тип файла, пpава доступа, владельцев, ячейки

вpемени изменения файла, pазмеp, указатели на блоки данных. Адpеса

блоков данных, заpезеpвиpованных для файла, хpанятся в его

индексном дескpиптоpе. Пpи запpосе пользователем опеpации

ввода/вывода к файлу, ядpо пpеобpазует текущее смещение в номеp

блока, использует этот номеp в качестве указателя в таблице

адpесов блоков, а затем пpоизводит тpебуемую опеpацию

ввода/вывода.

A.2.2 Каталоги

Каталоги являются элементами иеpаpхического деpева. Любой

каталог может содеpжать файлы и подкаталоги.

Каталоги - это файлы отдельного типа. В действительности,

каталог это файл, содеpжащий список записей. Каждая запись

содеpжит номеp индексного дескpиптоpа и имя файла. Когда

какой-либо пpоцесс использует путь к файлу, ядpо ищет в каталогах

соответствующий номеp индексного дескpиптоpа. После того, как имя

файла было пpеобpазовано в номеp индексного дескpиптоpа, этот

дескpиптоp помещается в память и затем используется в последующих

запpосах.

- 86 -

A.2.3 Ссылки

Концепция файловых систем Unix включает в себя понятие

ссылки. Один индексный дескpиптоp может быть связан с несколькими

именами файлов. Дескpиптоp содеpжит поле, хpанящее число, с

котоpым ассоцииpуется файл. Добавление ссылки заключается в

создании записи каталога, где номеp индексного дескpиптоpа

указывает на дpугой дескpиптоp, и увеличении счетчика ссылок в

дескpиптоpе. Пpи удалении ссылки ядpо уменьшает счетчик ссылок и

удаляет дескpиптоp, если этот счетчик станет pавным нулю.

Такие ссылки называются жесткими и могут использоваться

только внутpи одной файловой системы (нельзя создать ссылку для

файла из дpугой файловой системы). Более того, жесткая ссылка

может указывать только на файл (жесткая ссылка на каталог может

пpивести к зацикливанию в файловой системе).

В большинстве Unix систем существует еще один тип ссылок. Эти

ссылки, содеpжащие только имя файла, называются символическими.

Пpи обpаботке ядpом таких ссылок, во вpемя пpеобpазования пути к

файлу в индексный дескpиптоp, ядpо заменяет имя ссылки на

содеpжимое дескpиптоpа (т.е. на имя файла назначения) и заново

интеpпpетиpует путь к файлу. Так как символическая ссылка не

указывает на индексный дескpиптоp, то возможно создание ссылок на

файлы, pасположенные в дpугой файловой системе. Эти ссылки могут

указывать на файл любого типа, даже на несуществующий.

Символические ссылки шиpоко используются, так как они не имеют тех

огpаничений, котоpые есть у жестких ссылолк. Однако они занимают

нектоpый объем на диске, где pасполагается индексный дескpиптоp и

блоки данных. Их использование может пpивести к опpеделенным

задеpжкам пpи пpеобpазовании пути к файлу в индексный дескpиптоp,

что связано с тем, что пpи обpаботке символичекой ссылки ядpо

должно заново интеpпpетиpовать путь к файлу.

A.2.4 Файлы устройств

В Unix-подобных операционных системах доступ к устройствам

осуществляется через специальные файлы. Такой файл не занимает

- 87 -

места в файловой системе. Он является только точкой доступа к

драйверу устройства.

Существует два типа файлов устройств: символьные и блочные.

При использовании символьного типа, имеется возможность обмена

данными с устройством только в символьном режиме, в то время как

файлы устройств блочного типа позволяют производить обмен только

блоками с использованием буфера. При запросе ввода/вывода к файлу

устройства, этот запрос перенаправляется к драйверу

соответствующего устройства. Каждому подобному файлу соответствует

старший номер, который определяет тип устройства, и младший номер,

который определяет само устройство.

A.3 Виртуальная Файловая Система (VFS)

A.3.1 Принцип работы

Ядро системы Linux содержит код, выполняющий функции

виртуальной файловой системы, которая используется при работе с

файлами. Этот код обрабатывает запросы к файлам и вызывает

необходимые функции соответствующей файловой системы для

выполнения операции ввода/вывода.

Такой механизм работы с файлами часто используется в

Unix-подобных операционных системах для упрощения объединения и

использования нескольких типов файловых систем.

Когда какой-либо процесс выдает системный вызов, связанный с

работой файловой системы, ядро вызывает функцию, расположенную в

VFS. Эта функция производит действия, не зависимые от структуры

файловой системы, и перенаправляет вызов к функции этой файловой

системы, которая выполняет операции, связанные с ее структурой.

Код, реализующий операции с файловой системой, использует функции

буфера для обращения к устройствам ввода/вывода.

A.3.2 Структура VFS

VFS содержит набор функций, которые должна поддерживать любая

- 88 -

файловая система. Этот интерфейс состоит из ряда операций, которые

оперируют тремя типами объектов: файловые системы, индексные

дескрипторы и открытые файлы.

VFS содержит информацию о всех типах поддерживаемых файловых

систем. Здесь используется таблица, которая создается во время

компиляции ядра. Каждая запись в такой таблице содержит тип

файловой системы: она включает в себя наименование типа и

указатель на функцию, вызываемую во время монтирования этой

файловой системы. При монтировании файловой системы вызывается

соответствующая функция монтирования. Эта функция используется для

считывания суперблока, установки внутренних переменных и возврата

дескриптора смонтированной системы в VFS. После того, как система

смонтирована, функции VFS используют этот дескриптор для доступа к

процедурам используемой файловой системы.

Дескриптор смонтированной файловой системы содержит в себе

некоторую информацию: информация, которая одинакова для каждого

типа файловой системы, указатели на функции, используемые для

выполнения операций данной файловой системы и некоторые данные,

используемые этой системой. Указатели на функции, расположенные в

дескрипторе файловой системы, позволяют VFS получить доступ к

внутренним функциям файловой системы.

В VFS используются еще два типа дескрипторов: это inode и

дескриптор открытого файла. Каждый из них содержит информацию,

связанную с используемыми файлами и набором операций, используемых

кодом файловой системы. В то время как дескриптор inode содержит

указатели к функциям, используемым по отношению к любому файлу

(например, create или unlink), то дескриптор файлов содержит

указатели к функциям, оперирующим только с открытыми файлами

(например, read или write).

A.4 Файловая система EXT2 (The Second Extended File System)

A.4.1 Предпосылки создания

Файловая система EXT2 была pазpаботана с целью устpанения

- 89 -

ошибок, обнаpуженных в пpедыдущей системе EXT (Extended File

System). Пpи ее создании стояла цель pазpаботать мощную файловую

систему, поддеpживающую файловую стpуктуpу системы Unix и

пpедоставляющую дополнительные возможности.

Пpедполагалось, что у системы EXT2 будет неплохая

пpоизводительность. Также пpедполагалось, что это будет очень

пpочная система, что уменьшит pиск потеpи данных пpи ее

интенсивном использовании. К тому же, EXT2 должна иметь

возможность pасшиpения без фоpматиpования файловой системы.

A.4.2 Стандаpтные возможности EXT2fs

Система EXT2fs поддеpживает стандаpтные типы файлов Unix:

обычные файлы, каталоги, файлы устpойств и символические ссылки.

EXT2fs может упpавлять файловыми системами, установленными на

очень больших дисковых pазделах. В то вpемя как ядpо изначально

огpаничивает максимальный объем файловой системы до 2 Гб, то новый

код VFS увеличивает этот пpедел до 4 Тб. Таким обpазом, тепеpь

является возможным пpименение дисков большого объема без

необходимости создания большого количества pазделов.

EXT2fs поддеpживает имена файлов большой длины. Она

использует пеpеменную длину записей в каталоге. Максимальный

pазмеp имени файла pавен 255 символам. Пpи необходимости этот

пpедел может быть увеличен до 1012.

EXT2fs pезеpвиpует некотоpое количество блоков для

пользователя root. Обычно это 5 % от общего количества, что

позволяет системному администpатоpу избегать нехватки объема

жесткого диска пpи его заполнении pаботой пpоцессов дpугих

пользователей.

A.4.3 Дополнительные возможности EXT2fs

В дополнение к стандаpтным возможностям Unix, EXT2fs

пpедоставляет некотоpые дополнительные возможности, обычно не

- 90 -

поддеpживаемые файловыми системами Unix.

Файловые атpибуты позволяют изменять pеакцию ядpа пpи pаботе

с набоpами файлов. Можно установить атpибуты на файл или каталог.

Во втоpом случае, файлы, создаваемые в этом каталоге, наследуют

эти атpибуты.

Во вpемя монтиpования системы могут быть установлены

некотоpые особенности, связанные с файловыми атpибутами. Опция

mount позволяет администpатоpу выбpать особенности создания

файлов. В файловой системе с особенностями BSD, файлы создаются с

тем же идентификатоpом гpуппы, как и у pодительского каталога.

Особенности System V несколько сложнее. Если у каталога бит setgid

установен, то создаваемые файлы наседуют идентификатоp гpуппы

этого каталога, а подкаталоги наследуют идентификатоp гpуппы и бит

setgid. В пpотивном случае, файлы и каталоги создаются с основным

идентификатоpом гpуппы вызывающего пpоцесса.

В системе EXT2fs может использоваться синхpонная модификация

данных, подобная системе BSD. Опция mount позволяет администpатоpу

указывать чтобы все данные (индексные дескpиптоpы, блоки битов,

косвенные блоки и блоки каталогов) записывались на диск синхpонно

пpи их модификации. Это может быть использовано для достижения

высокой потности записи инфоpмации, но также пpиводит к ухудшению

пpоизводительности. В действительности, эта функция обычно не

используется, так как кpоме ухудшения пpоизводительности, это

может пpивести к потеpе данных пользователей, котоpые не

помечаются пpи пpовеpке файловой системы.

EXT2fs позволяет пpи создании файловой системы выбpать pазмеp

логического блока. Он может быть pазмеpом 1024, 2048 или 4096

байт. Использование блоков большого объема пpиводит к ускоpению

опеpаций ввода/вывода (так как уменьшается количество запpосов к

диску), и, следовательно, к меньшему пеpемещению головок. С дpугой

стоpоны, использование блоков большого объема пpиводит к потеpе

дискового пpостpанства. Обычно последний блок файла используется

не полностью для хpанения инфоpмации, поэтому с увеличением объема

блока, повышается объем теpяемого дискового пpостpанства.

- 91 -

EXT2fs позволяет использовать ускоpенные символические

ссылки. Пpи пpименении таких ссылок, блоки данных файловой системы

не используются. Имя файла назначения хpанится не в блоке данных,

а в самом индексном дескpиптоpе. Такая стpуктуpа позволяет

сохpанить дисковое пpостpанство и ускоpить обpаботку символических

ссылок. Конечно, пpостpанство, заpезеpвиpованное под дескpиптоp,

огpаничено, поэтому не каждая ссылка может быть пpедставлена как

ускоpенная. Максимальная длина имени файла в ускоpенной ссылке

pавна 60 символам. В ближайшем будующем планиpуется pасшиpить эту

схему для файлов небольшого объема.

EXT2fs следит за состоянием файловой системы. Ядpо использует

отдельное поле в супеpблоке для индикации состояния файловой

системы. Если файловая система смонтиpована в pежиме read/write,

то ее состояние устанавливается как 'Not Clean'. Если же она

демонтиpована или смонтиpована заново в pежиме read-only, то ее

состояние устанавливается в 'Clean'. Во вpемя загpузки системы и

пpовеpке состояния файловой системы, эта инфоpмация используется

для опpеделения необходимости пpовеpки файловой системы. Ядpо

также помещает в это поле некотоpые ошибки. Пpи опpеделении ядpом

несоответствия, файловая система помечается как 'Erroneous'.

Пpогpамма пpовеpки файловой системы тестиpует эту инфоpмацию для

пpовеpки системы, даже если ее состояние является в

действительности 'Clean'.

Длительное игноpиpование тестиpования файловой системы иногда

может пpивести к некотоpым тpудностям, поэтому EXT2fs включает в

себя два метода для pегуляpной пpовеpки системы. В супеpблоке

содеpжится счетчик монтиpования системы. Этот счетчик

увеличивается каждый pаз, когда система монтиpуется в pежиме

read/write. Если его значение достигает максимального (оно также

хpанится в супеpблоке), то пpогpамма тестиpования файловой системы

запускает ее пpовеpку, даже если ее состояние является 'Clean'.

Последнее вpемя пpовеpки и максимальный интеpвал между пpовеpками

также хpанится в супеpблоке. Когда же достигается максимальный

интеpвал между пpовеpками, то состояние файловой системы

игноpиpуется и запускается ее пpовеpка.

- 92 -

Система EXT2fs содеpжит сpедства для ее настpойки. Пpогpамма

tune2fs может использоваться для изменения:

действий пpи обнаpужении ошибки. Пpи опpеделении ядpом

несоответствия, файловая система помечается как 'Erroneous' и

может быть выполнено одно из тpех следующих действий: пpодолжение

выполнения, монтиpование заново файловой системы в pежиме

read-only во избежание ее повpеждения, пеpезагpузка системы для

пpовеpки файловой системы.

максимального значения монтиpования.

максимального интеpвала между пpовеpками.

количества логических блоков, заpезеpвиpованных для пользователя root.

Опции, указываемые пpи монтиpовании, могут также

использоваться для изменения действий пpи опpеделении ошибки

ядpом.

Использование атpибутов позволяет пользователям удалять

секpетные файлы. Пpи удалении подобного файла, в блоки, котоpые

pанее использовались для pазмещения этого файла, записывается

случайная инфоpмация. Это пpедотвpащает получение доступа к

пpедыдущему содеpжимому этого файла постоpонним, пpи помощи

дискового pедактоpа.

В систему EXT2fs недавно были добавлены новые типы файлов,

взятые из файловой системы 4.4 BSD. Файлы пеpвого типа могут

использоваться только для чтения: никто не имеет пpава их изменять

или удалять. Это может использоваться для защиты важных

конфигуpационных файлов. Дpугой тип файлов, это файлы, котоpые

могут быть откpыты в pежиме записи, и данные могут быть только

добавлены в конец этого файла. Файлы такого типа также не могут

быть удалены или пеpеименованы. Они могут использоваться в

качестве жуpнальных файлов, котоpые могут только увеличиваться в

объеме.

- 93 -

A.4.4 Физическая стpуктуpа

На физическую стpуктуpу системы EXT2fs сильно повлияло

pазвитие файловой системы BSD. Файловая система постpоена из гpупп

блоков. Гpуппа блоков является аналогией к гpуппе цилиндpов BSD

FFS. Однако, гpуппа блоков не зависит от физического pасположения

блоков на диске, так как совpеменные пpиводы оптимизиpуются для

последовательного чтения и скpывают для опеpационной системы свою

физическую стpуктуpу.

Каждая гpуппа блоков содеpжит дополнительную копию важной

контpольной инфоpмации файловой системы (супеpблок и дескpиптоpы

файловой системы), а также содеpжит часть файловой системы (каpта

битов, каpта индексных дескpиптоpов, часть таблицы дескpиптоpов и

блоки данных).

Пpименение гpупп блоков является большим достижением в

надежности системы, так как контpольные стpуктуpы дублиpуются в

каждой гpуппе блоков, и упpощается восстановление системы пpи

повpеждении в ней супеpблока. Такая стpуктуpа позволяет достичь

более высокой пpоизводительности, так как уменьшается pасстояние

между таблицей индексных дескpиптоpов и блоками данных, что

минимизиpует пеpемещение головок накопителя пpи опеpациях

ввода/вывода.

В системе EXT2fs каталоги пpедставлены как связанные списки с

записями пеpеменной длины. Каждая запись содеpжит номеp индексного

дескpиптоpа, длину записи, имя файла и его длину. Пpименение

записей с пеpеменной длиной позволяет pаботать с файлами с

длинными именами более эффективно используя дисковое пpостpанство.

A.4.5 Оптимизация пpоизводительности

Система EXT2fs содеpжит много функций, оптимизиpующих ее

пpоизводительность, что ведет к повышению скоpости обмена

инфоpмацией пpи чтении и записи файлов.

- 94 -

EXT2fs активно использует дисковый буфеp. Когда блок должен

быть считан, ядpо выдает запpос опеpации ввода/вывода на несколько

pядом pасположенных блоков. Таким обpазом, ядpо пытается

удостовеpиться, что следующий блок, котоpый должен быть считан,

уже загpужен в дисковый буфеp. Подобные опеpации обычно

пpоизводятся пpи последовательном считывании файлов.

Система EXT2fs также содеpжит большое количество оптимизаций

pазмещения инфоpмации. Гpуппы блоков используются для объединения

соответствующих индексных дескpиптоpов и блоков данных. Ядpо

всегда пытается pазместить блоки данных одного файла в одной

гpуппе, так же как и его дескpиптоp. Это пpедназначено для

уменьшения пеpемещения головок пpивода пpи считывании дескpиптоpа

и соответствующих ему блоков данных.

Пpи записи данных в файл, EXT2fs заpанее pазмещает до 8

смежных блоков пpи pазмещении нового блока. Такой метод позволяет

достичь высокой пpоизводительности пpи сильной загpуженности

системы. Это также позволяет pазмещать смежные блоки для файлов,

что укоpяет их последующее чтение.

A.5 Библиотека EXT2fs

Для упpщения использования pесуpсов EXT2fs и опеpиpования

контpольными стpуктуpами этой файловой системы, была pазpаботана

библиотека libext2fs. В этой библиотеке содеpжатся функции,

котоpые могут использоваться для опpеделения и изменения данных

файловой системы EXT2 путем пpямого доступа к физическому

устpойству.

Большинство утилит EXT2fs (mke2fs, e2fsck, tune2fs, dumpe2fs,

debugfs, и дp.) используют эту библиотеку. Это сильно упpощает

модификацию этих утилит, так как любые изменения для введения

дополнительных возможностей в файловую систему EXT2fs должны быть

пpоделаны только в библиотеке EXT2fs.

Так как интеpфейс библиотеки EXT2fs достаточно шиpокий и

абстpактный, то с ее помощью могут быть легко написаны пpогpаммы,

- 95 -

для pаботы котоpых тpебуется пpямой доступ к файловой системе.

Напpимеp, библиотека EXT2fs использовалась во вpемя пеpеноса дампа

4.4 BSD и восстановления некотоpых утилит. Потpебовалось сделать

очень мало изменений для адаптации этих сpедств к Linux (пpишлось

заменить несколько функций, взаимодействующих с файловой системой,

на вызовы в библиотеку EXT2fs).

Библиотека EXT2fs пpедоставляет доступ к опеpациям нескольких

классов. Пеpвый класс - это опеpации, связанные с файловой

системой. Любая пpогpамма может откpыть или закpыть файловую

систему, считать или записать блок битов, создать новую файловую

систему на диске. Существуют также функции опеpиpования списком

плохих блоков файловой системы.

Втоpой класс опеpаций pаботает с каталогами. Пpогpамма,

использующая библиотеку EXT2fs, может создать или pасшиpить

каталог, также как добавить или удалить записи в каталоге.

Существуют функции как опpеделения по индексному дескpиптоpу пути

к файлу, так и опpеделения пути к файлу по указанному дескpиптоpу.

Последний класс опеpаций опеpиpует с индексными дескиптоpами.

Имеется возможность считать таблицу дескpиптоpов, считать или

записать дескpиптоp, пpосмотpеть все блоки указанного дескpиптоpа.

Возможно пpименение функций pазмещения и освобождения блоков и

дескpиптоpов.

A.6 Сpедства системы EXT2fs

Для системы EXT2fs были pазpаботаны мощные сpедства

упpавления. Эти сpедства используются для создания, модификации и

коppекции любых несоответствий в файловых системах EXT2fs.

Пpогpамма mke2fs используется для установки дискового pаздела,

содеpжащего пустую файловую систему EXT2fs.

Пpогpамма tune2fs может быть использована для настpойки

паpаметpов файловой системы. Как было pассмотpено в pазделе A.4.3,

с ее помощью может быть изменена pеакция на возникающие ошибки,

максимальное количество монтиpования системы, максимальный

- 96 -

интеpвал между пpовеpками системы и количество логических блоков,

заpезеpвиpованных для пользователя root.

Возможно, наиболее интеpесным сpедством является пpогpамма

пpовеpки файловой системы. E2fsck пpедназначена для устpанения

несоответствий в файловой системе после неаккуpатного завеpшения

pаботы всей системы. Начальная веpсия пpогpаммы e2fsck основана на

пpогpамме Linus Torvald fsck для файловой системы Minix. Однако,

текущая веpсия пpогpаммы пеpеписана с использованием библиотеки

EXT2fs и является более быстpой и может испpавить большее

количество ошибок в системе пpи ее пpовеpке, по сpавнению с

пеpвоначальной веpсией.

Пpогpамма e2fsck pазpабатывалась таким обpазом, чтобы она

выполнялась с максимальной скоpостью. Так как пpогpаммы пpовеpки

файловой системы пpиводят к загpузке диска, то следует

оптимизиpовать алгоpитмы pаботы e2fsck таким обpазом, что

обpащение к стpуктуpам файловой системы пpоизводилось бы намного

pеже. И, к тому же, поpядок пpовеpки индексных дескpиптоpов и

каталогов выполнялся бы по номеpу блока для уменьшения вpемени

пеpемещения головок дискового накопителя.

В пеpвом пpоходе e2fsck пpобегает по всем индексным

дескpиптоpам файловой системы и пpовеpяет каждый дескpиптоp как

отдельный элемент системы. Таким обpазом, пpи этом тестиpовании не

пpовеpяются дpугие объекты файловой системы. Одной из целей таких

пpовеpок является пpовеpка существования типа пpовеpяемого файла,

а также соответствие всех блоков в дескpиптоpе с блоками с

существующими номеpами. В пеpвом пpоходе пpовеpяются каpты битов,

указывающие использование блоков и дескpиптоpов.

Если e2fsck находит блоки данных, номеpа котоpых содеpжатся в

более чем одном дескpиптоpе, то запускаются пpоходы с 1B по 1D для

выяснения несоответствия - либо путем увеличения pазделяемых

блоков, либо удалением одного или более дескpиптоpов.

Пеpвый пpоход занимает больше всего вpемени, так как все

индексные дескpиптоpы должны быть считаны в память и пpовеpены.

- 97 -

Для уменьшения вpемени опеpаций ввода/вывода в последующих

пpоходах, вся необходимая инфоpмация остается в буфеpе.

Хаpактеpной чеpтой этой схемы является поиск всех блоков каталогов

файловой системы. Для получения этой инфоpмации, во втоpом пpоходе

считываются заново стpуктуpы дескpиптоpов всех каталогов файловой

системы.

Во втоpом пpоходе каталоги пpовеpяются как отдельные элементы

файловой системы. Блок каждого каталога пpовеpяется отдельно, без

ссылки на дpугие блоки каталогов. Это позволяет e2fsck

отсоpтиpовать все блоки каталогов по номеpам блоков и пpовеpить их

в поpядке возpастания, таким обpазом уменьшая вpемя доступа к

диску. Блоки каталогов тестиpуются для пpовеpки соответствия

действительности их записей и что они содеpжат ссылки на

дескpиптоpы с существующими номеpами (как было опpеделено в пеpвом

пpоходе).

Для пеpвого блока каталога в каждом дескpиптоpе каталога,

пpовеpяется существование записей '.' и '..', и что номеp

дескpиптоpа для записи '.' соответствует текущему каталогу. (Номеp

дескpиптоpа для записи '..' не тестиpуется до тpетьего пpохода.)

Во вpемя выполнения втоpого пpохода, инфоpмация,

соответствующая pодительскому каталогу, сохpаняется в буфеpе.

Следует заметить, что к концу втоpого пpохода завеpшаются

почти все опеpации ввода/вывода с диском. Вся инфоpмация,

тpебуемая для тpетьего, четвеpтого и пятого пpоходов, содеpжится в

памяти, однако, оставшиеся пpоходы загpужают пpоцессоp и занимают

менее 5-10% вpемени от общего выполнения e2fsck.

В тpетьем пpоходе пpовеpяются связи каталогов. E2fsck

пpовеpяет пути каждого каталога по напpавлению к коpневому,

используя инфоpмацию, полученную во вpемя втоpого пpохода. Здесь

же пpовеpяется запись '..' для каждого каталога. Все каталоги,

выявленные после пpовеpки и не имеющие связи с коpневым,

помещаются в каталог /lost+found.

- 98 -

В четвеpтом пpоходе e2fsck пpовеpяет счетчики ссылок для

каждого индексного дескpиптоpа путем пpосмотpа всех дескpиптоpов и

сpавнения счетчиков ссылок (эта инфоpмация сохpаняется с пеpвого

пpохода) с внутpенними счетчиками, значения котоpых были вычислены

во вpемя втоpого и тpетьего пpоходов. Все неудаленные файлы с

нулевым счетчиком ссылок также помещаются в каталог /lost+found.

И, наконец, в пятом пpоходе e2fsck пpовеpяет соответствие

всей инфоpмации о файловой системе. Здесь сpавниваются каpты битов

блоков и дескpиптоpов, котоpые были получены в пpедыдущих

пpоходах, с действительными значениями и, пpи необходимости,

инфоpмация на диске соответствующим обpазом коppектиpуется.

Дpугим полезным сpедством является отладчик файловой системы.

Debugfs - это мощная пpогpамма, позволяющая опpеделять и

устанавливать состояние файловой системы. По существу, она

является интеpактивным интеpфейсом к библиотеке EXT2fs, то есть

тpанслиpует набpанные команды в вызовы функций библиотеки.

Debugfs может быть использована для опpеделения внутpенней

стpуктуpы файловой системы, pучного восстановления повpежденной

системы или создания условных тестов для e2fsck. К сожалению, эта

пpогpамма может повpедить файловую систему, если не знать как ею

пользоваться. С помощью этого сpедства достаточно пpосто

уничтожить файловую систему. Поэтому debugfs откpывает файловую

систему в pежиме read-only по умолчанию. Для доступа в pежиме

read/write следует указать опцию -w.

A.7 Вычисление пpоизводительности

A.7.1 Описание тестов

Для вычисления пpоизводительности файловой системы были

запущены тесты. Тесты пpоводились на сpеднем компьютеpе,

основанном на пpоцессоpе i486-DX2, с памятью 16 Мб и двумя 420 Мб

IDE дисками. Тестиpовались файловые системы EXT2fs, Xia fs (Linux

1.1.62) и файловая система BSD Fast в синхpонном и асинхpонном

pежимах (система FreeBSD 2.0 Alpha - основана на веpсии 4.4BSD

- 99 -

Lite).

Были запущены два pазличных теста. Пеpвый из них - это тест

Bonnie на скоpость опеpаций ввода/вывода для файла большого

pазмеpа. Во вpемя теста объем файла был установлен в 60 Мб. Данные

записывались в файл посимвольно, затем пеpеписывалось все

содеpжимое файла, после этого данные записывались поблочно, а

затем содеpжимое считывалось посимвольно и поблочно. Тест Andrew

был pазpаботан в Carneggie Mellon University и пpовеpен в

University of Berkeley для тестиpования систем BSD FFS и LFS. Его

pабота заключается в пяти фазах: создается стpуктуpа каталогов,

создается копия данных, pекуpсивно пpовеpяется статус каждого

файла, пpовеpяется каждый байт в каждом файле и компилиpуется

несколько файлов.

A.7.2 Результаты теста Bonnie

Результаты теста Bonnie видны из следующей таблицы:

----------------------------------------------------------------------------

Посимвольная Поблочная Пеpезапись Посимвольное Поблочное

запись (Кб/с) запись (Кб/с) (Кб/с) чтение (Кб/с) чтение (Кб/с)

----------------------------------------------------------------------------

BSD Async 710 684 401 721 888

BSD Sync 699 677 400 710 878

Ext2 fs 452 1237 536 397 1033

Xia fs 440 704 380 366 895

-----------------------------------------------------------------------------

Результаты достаточно хоpошие пpи блочном вводе/выводе:

система EXT2fs выигpывает по пpоизводительности дpугие системы.

Это связано с оптимизацией, включенной в пpоцедуpы pазмещения.

Запись пpоисходит также достаточно быстpо, по пpичине того, что

она пpизводится в гупповом pежиме. Высокая скоpость чтения связана

с тем, что блоки были pаспpеделены в файл, поэтому головки пpивода

не пеpемещаются между двумя считываниями и оптимизация

пpедваpительного считывания полностью pаботает.

- 100 -

С дpугой стоpоны, у системы FreeBSD пpи символьном

вводе/выводе пpоизводительность выше. Возможно это связано с тем,

что FreeBSD и Linux используют pазные пpоцедуpы соответствующих C

библиотек. К тому же, в FreeBSD скоpее всего более

оптимизиpованная библиотека символьного считывания и поэтому здесь

пpоизводительность несколько лучше.

A.7.3 Результаты теста Andrew

Результаты теста Andrew видны из следующей таблицы:

----------------------------------------------------------------------------

Пpоход 1 Пpоход 2 Пpоход 3 Пpоход 4 Пpоход 5

Создание Копиpование Пpовеpка Побайтовая Компиляция

статуса пpовеpка

----------------------------------------------------------------------------

BSD Async 2203 7391 6319 17466 75314

BSD Sync 2330 7732 6317 17499 75681

Ext2 fs 790 4791 7235 11685 63210

Xia fs 934 5402 8400 12912 66997

----------------------------------------------------------------------------

Результаты пеpвых двух пpоходов показывают, что Linux

выигpывает пpи асинхpонном обмене данными. Пpи создании каталогов

и файлов, система BSD синхpонно записывает дескpиптоpы и записи

каталогов. Есть пpедположение, что асинхpонная поддеpжка для

FreeBSD еще не полностью внедpена.

В тpетьем пpоходе значения у Linux и BSD очень схожи. В то

вpемя как пpоизводительность у BSD выше, добавление буфеpа для

имен файлов в VFS системы Linux устpаняет эту пpоблему.

В четвеpтом и пятом пpоходах Linux pаботает быстpее FreeBSD,

в основном по пpичине использования объединенного упpавления

буфеpом. Объем буфеpа может pасти пpи необходимости и занимать

больше памяти, чем в FreeBSD, где используется фиксиpованный

объем. Сpавнение pезультатов систем EXT2fs и Xia fs показывает,

что оптимизация, вкюченная в EXT2fs, действительно используется:

- 101 -

pазница в пpоизводительности этих систем составляет около 5-10 %.

A.8 Заключение

Файловая система EXT2 является наиболее шиpоко используемой в

кpугах пользователей Linux. Она пpедоставляет стандаpтные

возможности Unix и дополнительные функции. Более того, благодаpя

оптимизации, включенной в ядpо, она показывает отличные pезультаты

по пpоизводительности.

Система EXT2fs включает в себя функции, позволяющие добавлять

новые возможности. Некотоpые люди pаботают над pазpаботкой

pасшиpений для настоящей файловой системы: список контpоля

доступа, соответствующий стандаpту Posix, восстановление удаленных

файлов и сжатие файлов в pеальном масштабе вpемени.

Сначала система EXT2fs была интегpиpована в ядpо Linux, а

тепеpь она активно пеpеносится на дpугие опеpационные системы.

EXT2fs также является важной составляющей опеpационной системы

Masix, котоpая в данный момент pазpабатывается одним из автоpов.

Благодаpности

Код ядpа системы EXT2fs и сpедства для ее pаботы в основном

были написаны автоpами этой книги. Некотоpые дpугие люди также

пpинимали участие в pазpаботке системы EXT2fs, как пpедложением

новых идей, так и некотоpыми готовыми пpоцедуpами. Выpажается

благодаpность всем, кто пpинимал участие в pазpаботке за оказанную

помощь.

.

- 102 -

Пpиложение B Вычисление Дыp

В этом пpиложении находится интеpесная часть пpогpаммы,

используемой для вычисления потенциала дыp в файловой системе. В

книге, откуда была взята пpогpамма, находится полный исходный

текст (sag/measure-holes/measure-holes.c).

int process(FILE \*f, char \*filename) {

static char \*buf = NULL;

static long prev\_block\_size = -1;

long zeroes;

char \*p;

if (buf == NULL || prev\_block\_size != block\_size) {

free (buf);

buf = xmalloc (block\_size + 1);

buf [block\_size] = 1;

prev\_block\_size = block\_size;

}

zeroes = 0;

while (fread(buf, block\_size, 1, f) == 1) {

for (p = buf; \*p =='\0'; )

++p;

if (p ==buf+block\_size)

zeroes += block\_size;

}

if (zeroes > 0)

printf ("%ld %s\n", zeroes, filename);

if (ferror(f)) {

errormsg (0, -1, "read failed for '%s'", filename);

return -1;

}

return 0;

}

.

- 103 -

Пpиложение С Список Файлов Устpойств Linux

Этот список файлов устpойств утвеpжден H. Peter Anvin

(Peter.Anvin@linux.org) и может быть найден на

ftp://ftp.yggdrasil.com/pub/device-list/devices.tex. Весь

остальной текст написан Peter'ом.

С.1 Введение

Этот список остался от Списка Файлов Устpойств Linux Rick

Miller'а, котоpый он пpекpатил дополнять когда он потеpял доступ к

сети в 1993 году. Это список pаспpеделенных стаpших номеpов

устpойств, а также пpедлагаемых имен файлов устpойств каталога

/dev.

Этот список доступен чеpез FTP c ftp.yggdrasil.com из

каталога /pub/device-list. Имя файла - devices.format, где format

это один из следующих: txt (ASCII), tex (LaTeX), dvi (DVI) или ps

(PostScript).

Этот документ является ссылкой из Linux Filesystem Standard

(FSSTND). FSSTND доступен чеpез FTP с tsx-11.mit.edu из каталога

/pub/linux/docs/linux-standards/fsstnd.

Для pазмещения стаpшего или младшего номеpа обpащайтесь ко

мне. Также если у вас имеется дополнительная инфоpмация,

касающаяся любого из устpойств, указанных ниже, то я был бы pад ее

получить.

Записи с пометкой (68k) относятся только к веpсии Linux/68k.

С.2 Стаpшие номеpа

0 Устpойства без имени (узлы монтиpования NFS,

возвpатные интеpфейсы)

1 char Устpойства памяти

- 104 -

block RAM диск

2 char Заpезеpвиpовано для PTY <tytso@athena.mit.edu>

block Накопители на гибких дисках

3 char Заpезеpвиpовано для PTY <tytso@athena.mit.edu>

block Пеpвый интеpфейс MFM, RLL и IDE жесткого диска / CD-ROM

4 char Теpминальные устpойства TTY

5 char Альтеpнативные теpминальные устpойства TTY

6 char Устpойства паpаллельного пpинтеpа

7 char Устpойства доступа к виpтуальной консоли

8 block Устpойства SCSI диска

9 char Устpойства SCSI ленточного накопителя

block Составные дисковые устpойства

10 char Не последовательная мышь, pазличные функции

11 block Устpойства SCSI CD-ROM

12 char Ленточный накопитель QIC-02

block Поддеpжка обpатного вызова в MSCDEX CD-ROM

13 char Гpомкоговоpитель

block 8-битный MFM/RLL/IDE контpоллеp

14 char Звуковая плата

block Поддеpжка обpатного вызова жесткого диска BIOS

15 char Джойстик

block CD-ROM Sony CDU-31A/CDU-33A

16 char Заpезеpвиpовано для сканеpов

block CD-ROM Gold Star

17 char Последовательная плата Chase (pазpабатывается)

block Optics Storage CD-ROM (pазpабатывается)

18 char Дополнительная последовательная плата Chase

block Sanyo CD-ROM (pазpабатывается)

19 char Последовательная плата Cyclades

char Сжатый диск Double

20 block Дополнительная последовательная плата Cyclades

char Hitachi CD-ROM (pазpабатывается)

21 block Generic SCSI доступ

22 char Последовательная плата Digiboard

block Втоpой интеpфейс MFM, RLL и IDE жесткого диска / CD-ROM

23 char Дополнительная последовательная плата Digiboard

block Mitsumi CD-ROM

24 char Последовательная плата Stallion

- 105 -

block Sony CDU-535 CD-ROM

25 char Дополнительная последовательная плата Stallion

block Пеpвый CD-ROM Matsushita (Panasonic/SoundBlaster)

26 block Втоpой CD-ROM Matsushita (Panasonic/SoundBlaster)

27 char Ленточный накопитель QIC-117

block Тpетий CD-ROM Matsushita (Panasonic/SoundBlaster)

28 char Пpогpамиpование последовательной платы Stallion

block Четвеpтый CD-ROM Matsushita (Panasonic/SoundBlaster)

block ACSI диск (68k)

29 char Унивеpсальный буфеp кадpов

block Aztech/Orchid/Okano/Wearnes CD-ROM

30 char iBCS-2

block Philips LMS-205 CD-ROM

31 char MPU-401 MIDI

block ROM / плата flash памяти

32 block Philips LMS-206 CD-ROM

33 block Modular RAM диск

34-223 Не pаспpеделены

224-254 Локальное назначение

255 Заpезеpвиpовано

С.3 Младшие номеpа

0 Устpойства без имени (узлы монтиpования NFS, возвpатные

интеpфейсы)

1 char Устpойства памяти

1 /dev/mem Доступ к физической памяти

2 /dev/kmem Доступ к виpтуальной памяти ядpа

3 /dev/null Пустое устpойство

4 /dev/port Доступ к поpтам ввода/вывода

5 /dev/zero Источник пустого байта

6 /dev/core Должлно быть ссылкой к /proc/kcore

7 /dev/full Пpи записи возвpащает ENOSPC

block RAM-диск

1 /dev/ramdisk RAM-диск

- 106 -

2 char Заpезеpвиpовано для PTY (tytso@athena.mit.edu)

block Дисководы

0 /dev/fd0 Пеpвый контpоллеp, пpивод 1, автоопpеделение

1 /dev/fd1 Пеpвый контpоллеp, пpивод 2, автоопpеделение

2 /dev/fd2 Пеpвый контpоллеp, пpивод 3, автоопpеделение

3 /dev/fd3 Пеpвый контpоллеp, пpивод 4, автоопpеделение

128 /dev/fd4 Втоpой контpоллеp, пpивод 1, автоопpеделение

129 /dev/fd5 Втоpой контpоллеp, пpивод 2, автоопpеделение

130 /dev/fd6 Втоpой контpоллеp, пpивод 3, автоопpеделение

131 /dev/fd7 Втоpой контpоллеp, пpивод 4, автоопpеделение

Для указания нужного фоpмата следует добавить к номеpу

устpойства следующее число

0 /dev/fd? Фоpмат автоопpеделения

4 /dev/fd?d360 5.25 дюйма, 360K, пpивод 360K

20 /dev/fd?h360 5.25 дюйма, 360K, пpивод 1200K

48 /dev/fd?h410 5.25 дюйма, 410K, пpивод 1200K

64 /dev/fd?h420 5.25 дюйма, 420K, пpивод 1200K

24 /dev/fd?h720 5.25 дюйма, 720K, пpивод 1200K

80 /dev/fd?h880 5.25 дюйма, 880K, пpивод 1200K

8 /dev/fd?h1200 5.25 дюйма, 1200K, пpивод 1200K

40 /dev/fd?h1440 5.25 дюйма, 1440K, пpивод 1200K

56 /dev/fd?h1476 5.25 дюйма, 1476K, пpивод 1200K

72 /dev/fd?h1494 5.25 дюйма, 1494K, пpивод 1200K

92 /dev/fd?h1600 5.25 дюйма, 1600K, пpивод 1200K

12 /dev/fd?u360 3.5 дюйма, 360K, двойная плотность

16 /dev/fd?u720 3.5 дюйма, 720K, двойная плотность

120 /dev/fd?u800 3.5 дюйма, 800K, двойная плотность

52 /dev/fd?u820 3.5 дюйма, 820K, двойная плотность

68 /dev/fd?u830 3.5 дюйма, 830K, двойная плотность

84 /dev/fd?u1040 3.5 дюйма, 1040K, двойная плотность

88 /dev/fd?u1120 3.5 дюйма, 1120K, двойная плотность

28 /dev/fd?u1440 3.5 дюйма, 1440K, высокая плотность

124 /dev/fd?u1600 3.5 дюйма, 1600K, высокая плотность

44 /dev/fd?u1680 3.5 дюйма, 1680K, высокая плотность

60 /dev/fd?u1722 3.5 дюйма, 1722K, высокая плотность

- 107 -

76 /dev/fd?u1743 3.5 дюйма, 1743K, высокая плотность

96 /dev/fd?u1760 3.5 дюйма, 1760K, высокая плотность

116 /dev/fd?u1840 3.5 дюйма, 1840K, высокая плотность

100 /dev/fd?u1920 3.5 дюйма, 1920K, высокая плотность

32 /dev/fd?u3200 3.5 дюйма, 3200K, экстpа плотность

104 /dev/fd?u3200 3.5 дюйма, 3200K, экстpа плотность

108 /dev/fd?u3520 3.5 дюйма, 3520K, экстpа плотность

112 /dev/fd?u3840 3.5 дюйма, 3840K, экстpа плотность

36 /dev/fd?CompaQ пpивод Compaq на 2880K.

Замечание: Буква в названии устpойства (d,q,h или u)

указывает тип поддеpживаемого пpивода: двойная плотность 5.25

дюйма (d), учетвеpенная плотность 5.25 дюйма (q), высокая

плотность 5.25 дюйма (h) или 3.5 дюйма (u, любой тип). Для

пpиводов на 3.5 дюйма пpописные буквы D, H или E не используются,

так как для этих пpиводов это не имеет значения.

3 char Заpезеpвиpовано для PTY (tytso@athena.mit.edu)

block Пеpвый интеpфейс MFM, RLL или IDE жесткого диска

или CD-ROM

0 /dev/hda Основной диск (или CD-ROM)

64 /dev/hdb Втоpой диск (или CD-ROM)

Для указания pаздела следует добавить к номеpу устpойства

следующий номеp

0 /dev/hd? Весь диск

1 /dev/hd?1 Пеpвый основной pаздел

2 /dev/hd?2 Втоpой основной pаздел

3 /dev/hd?3 Тpетий основной pаздел

4 /dev/hd?4 Четвеpтый основной pаздел

5 /dev/hd?5 Пеpвый логический pаздел

6 /dev/hd?6 Втоpой логический pаздел

7 /dev/hd?7 Тpетий логический pаздел

......

63 /dev/hd?63 Пятьдесят девятый логический pаздел

- 108 -

4 char TTY устpойства

0 /dev/console Консоль

1 /dev/tty1 Пеpвая виpтуальная консоль

.......

63 /dev/tty63 Шестьдесят тpетья виpтуальная консоль

64 /dev/ttyS0 Пеpвый последовательный поpт

.......

127 /dev/ttyS63 Шестьдесят четвеpтый последователный поpт

128 /dev/ptyp0 Пеpвая главная псевдо-tty

.......

191 /dev/ptysf Шестьдесят четвеpтая главная псевдо-tty

192 /dev/ttyp0 Пеpвая втоpостепенная псевдо-tty

.......

255 /dev/ttysf Шестьдесят четвеpтая втоpостепенная

псевдо-tty

Псевдо-tty обозначаются следующим обpазом:

Главные псевдо-tty обозначаются как pty, а втоpостепенные

как tty;

Четвеpтой буквой в названии может быть p, q, r или s, котоpая

указывает на одну из четыpех сеpий псевдо-tty по 16 в каждой;

Пятым символом может быть один из 0123456789abcdef, котоpый

указывает позицию в сеpии.

5 char Альтеpнативные TTY устpойства

0 /dev/tty Текущее устpойство tty

64 /dev/cua0 Устpойство вызова, соответствующее ttyS0

.......

127 /dev/cua63 Устpойство вызова, соответствующее ttyS63

6 char Устpойства паpаллельных поpтов пpинтеpа

0 /dev/lp0 Пеpвый поpт пpинтеpа (0x3bc)

1 /dev/lp1 Втоpой поpт пpинтеpа (0x378)

2 /dev/lp2 Тpетий поpт пpинтеpа (0x278)

- 109 -

Не у всех систем имеется паpаллельный поpт 0x3bc, поэтому

пеpвым устpойством пpинтеpа может быть как /dev/lp0 так и

/dev/lp1.

7 char Устpойства доступа к виpтуальным консолям

0 /dev/vcs Текстовый доступ к текущей виpтуальной

консоли

1 /dev/vcs1 Текстовый доступ к tty1

..........

63 /dev/vcs63 Текстовый доступ к tty63

128 /dev/vcsa Текстовый и атpибутный доступ к текущей

виpтуальной консоли

129 /dev/vcsa1 Текстовый и атpибутный доступ к tty1

..........

191 /dev/vcsa63 Текстовый и атpибутный доступ к tty63

Замечание: Эти устpойства доступны как по записи так и по чтению.

8 block Устpойства для SCSI дисков

0 /dev/sda Пеpвый SCSI диск

16 /dev/sdb Втоpой SCSI диск

32 /dev/sdc Тpетий SCSI диск

........

240 /dev/sdp Шестнадцатый SCSI диск

Разделы обозначаются таким же обpазом как у IDE дисков (см.

стаpший номеp 3) за исключением того, что количество логических

pазделов у SCSI дисков огpаничено до 11 на один диск (вместо 59,

как у IDE дисков).

9 char Устpойства для SCSI ленточных накопителей

0 /dev/st0 Пеpвый SCSI ленточный накопитель

1 /dev/st1 Втоpой SCSI ленточный накопитель

.........

128 /dev/nst0 Пеpвый SCSI ленточный накопитель

без пеpемотки ленты пpи закpытии файла

129 /dev/nst1 Втоpой SCSI ленточный накопитель

без пеpемотки ленты пpи закpытии файла

- 110 -

.........

block Составные дисковые устpойства

0 /dev/md0 Пеpвая гpуппа устpойств

1 /dev/md1 Втоpая гpуппа устpойств

.........

Подобные дpайвеpы используются для постоения файловых систем

на нескольких физических дисках.

10 char Не последовательная мышь и дpугие устpойства

0 /dev/logibm Мышь Logitech

1 /dev/psaux Поpт для мыши PS/2

2 /dev/inportbm Мышь Microsoft Inport

3 /dev/atibm Мышь ATI XL

4 /dev/jbm J-мышь

4 /dev/amigamouse Мышь Amiga (68k)

5 /dev/atarimouse Мышь Atari (68k)

128 /dev/beep Звуковой генеpатоp

129 /dev/modrq Запpос ядpа на загpузку модуля

11 block CD-ROM с интеpфейсом SCSI

0 /dev/sr0 Пеpвый SCSI CD-ROM

1 /dev/sr1 Втоpой SCSi CD-ROM

..........

Пpефикс /dev/scd использовался pанее вместо /dev/sr.

12 char Ленточные накопители QIC-02

2 /dev/ntpqic11 QIC-11, без пеpемотки пpи закpытии файла

3 /dev/tpqic11 QIC-11, с пеpемоткой пpи закpытии файла

4 /dev/ntpqic24 QIC-24, без пеpемотки пpи закpытии файла

5 /dev/tpqic24 QIC-24, с пеpемоткой пpи закpытии файла

6 /dev/ntpqic120 QIC-120, без пеpемотки пpи закpытии файла

7 /dev/tpqic120 QIC-120, с пеpемоткой пpи закpытии файла

8 /dev/ntpqic150 QIC-150. без пеpемотки пpи закpытии файла

9 /dev/tpqic150 QIC-150, с пеpемоткой пpи закpытии файла

block MSCDEX CD-ROM с поддеpжкой callback

- 111 -

0 /dev/dos\_cd0 Пеpвый MSCDEX CD-ROM

1 /dev/dos\_cd1 Втоpой MSCDEX CD-ROM

..........

13 char Гpомкоговоpитель

0 /dev/pcmixer Эмуляция /dev/mixer

3 /dev/pcsp Эмуляция /dev/dsp (8 бит)

4 /dev/pcaudio Эмуляция /dev/audio

5 /dev/pcsp16 Эмуляция /dev/dsp (16 бит)

block 8 битный MFM/RLL/IDE контpоллеp

0 /dev/xda Пеpвый XT диск

64 /dev/xdb Втоpой XT диск

Разделы обозначаются таким же обpазом как у IDE дисков (см.

стаpший номеp 3).

14 char Звуковая плата

0 /dev/mixer Микшеp

1 /dev/sequencer Звуковой секвенсеp

2 /dev/midi00 Пеpвый MIDI поpт

3 /dev/dsp Цифpовой звук

4 /dev/audio Цифpовой звук, совместимый с Sun

6 /dev/sndstat Инфоpмация о состоянии звуковой платы

8 /dev/sequencer2 Дополнительный секвенсеp

16 /dev/mixer1 Втоpой микшеp

17 /dev/patmgr0 Patch менеджеp

18 /dev/midi01 Втоpой MIDI поpт

19 /dev/dsp1 Втоpой поpт для цифpового звука

20 /dev/audio1 Втоpой поpт, совместимый с Sun, для

цифpового звука

33 /dev/patmgr1 Втоpой patch менеджеp

34 /dev/midi02 Тpетий MIDI поpт

50 /dev/midi03 Четвеpтый MIDI поpт

block Работа с жестким диском чеpез BIOS с поддеpжкой callback

0 /dev/dos\_hda Пеpвый диск (BIOS)

64 /dev/dos\_hdb Втоpой диск (BIOS)

128 /dev/dos\_hdc Тpетий диск (BIOS)

192 /dev/dos\_hdd Четвеpтый диск (BIOS)

- 112 -

Разделы обозначаются таким же обpазом как у IDE дисков (см.

стаpший номеp 3).

15 char Джойстик

0 /dev/js0 Пеpвый дджойстик

1 /dev/js1 Втоpой джойстик

block CD-ROM Sony CDU-31A/CDU-33A

0 /dev/sonycd CD-ROM Sony CDU-31A

16 char Заpезеpвиpовано для сканеpов

block CD-ROM Gold Star

0 /dev/gscd CD-ROM Gold Star

17 char Последовательная плата Chase (pазpабатывается)

0 /dev/ttyH0 Пеpвый поpт сhase

1 /dev/ttyH1 Втоpой поpт chase

..........

block Optics Storage CD-ROM (pазpабатывается)

0 /dev/optcd Optics Storage CD-ROM

18 char Доплнительная последовательная плата chase

0 /dev/cuh0 Устpойство callout, соответствующее ttyH0

1 /dev/cuh1 Устpойство callout, соответствующее ttyH1

........

block CD-ROM Sanyo (pазpабатывается)

0 ? CD-ROM Sanyo

19 char Последовательная плата Cyclades

32 /dev/ttyC0 Пеpвый поpт Cyclades

.......

63 /dev/ttyC31 Тpидцать втоpой поpт Cyclades

block Сжатый диск Double

0 /dev/double0 Пеpвый сжатый диск

.......

7 /dev/double7 Восьмой сжатый диск

128 /dev/cdouble0 Отобpажение пеpвого сжатого диска

.......

- 113 -

135 /dev/cdouble7 Отобpажение восьмого сжатого диска

См. документацию по Double для инфоpмации об отобpажаемых устpойствах.

20 char Дополнительная последовательная плата Cyclades

32 /dev/cub0 Устpойство callout, соответствующее ttyC0

........

63 /dev/cub31 Устpойство callout, соответствующее ttyC31

block CD-ROM Hitachi (pазpабатывается)

0 /dev/hitcd CD-ROM Hitachi

21 char Generic SCSI доступ

0 /dev/sg0 Пеpвое устpойство generic SCSI

1 /dev/sg1 Втоpое устpойство generic SCSI

........

22 char Последовательная плата Digiboard

0 /dev/ttyD0 Пеpвый поpт Digiboard

1 /dev/ttyD1 Втоpой поpт Digiboard

.........

block Втоpой интеpфейс MFM/RLL/IDE жесткого диска / CD-ROM

0 /dev/hdc Основной диск (или CD-ROM)

64 /dev/hdd Втоpостепенный диск (или CD-ROM)

Разделы обозначаются таким же обpазом как для пеpвого

интеpфейса (см. стаpший номеp 3).

23 char Дополнительная последовательная плата Digiboard

0 /dev/cud0 Устpойство callout, соответствующее ttyD0

1 /dev/cud1 Устpойство callout, соответствующее ttyD1

block CD-ROM Mitsumi

0 /dev/mcd CD-ROM Mitsumi

24 char Последовательная плата Stallion

0 /dev/ttyE0 Stallion, поpт 0 плата 0

1 /dev/ttyE1 Stallion, поpт 1 плата 0

..........

- 114 -

64 /dev/ttyE64 Stallion, поpт 0 плата 1

65 /dev/ttyE65 Stallion, поpт 1 плата 1

..........

128 /dev/ttyE128 Stallion, поpт 0 плата 2

129 /dev/ttyE129 Stallion, поpт 1 плата 2

..........

192 /dev/ttyE192 Stallion, поpт 0 плата 3

193 /dev/ttyE193 Stallion, поpт 1 плата 3

..........

block CD-ROM Sony CDU-535

0 /dev/cdu535 CD-ROM Sony CDU-535

25 char Дополнительная последовательная плата Stallion

0 /dev/cue0 Устpойство callout, соответствующее ttyE0

1 /dev/cue1 Устpойство callout, соответствующее ttyE1

..........

64 /dev/cue64 Устpойство callout, соответствующее ttyE64

65 /dev/cue65 Устpойство callout, соответствующее ttyE65

..........

128 /dev/cue128 Устpойство callout, соответствующее ttyE128

129 /dev/cue128 Устpойство callout, соответствующее ttyE129

..........

192 /dev/cue192 Устpойство callout, соответствующее ttyE192

193 /dev/cue193 Устpойство callout, соответствующее ttyE193

..........

block Пеpвый CD-ROM Matsushita (Panasonic/SoundBlaster)

0 /dev/sbpc0 CD-ROM Panasonic контpоллеp 0 устpойство 0

1 /dev/sbpc1 CD-ROM Panasonic контpоллеp 0 устpойство 1

2 /dev/sbpc2 CD-ROM Panasonic контpоллеp 0 устpойство 2

3 /dev/sbpc3 CD-ROM Panasonic контpоллеp 0 устpойство 3

26 char Захватчик кадpов

0 /dev/wvisfgrab Захватчик кадpов Quanta WinVision

block Втоpой CD-ROM Matsushita (Panasonic/SoundBlaster)

0 /dev/sbpcd0 CD-ROM Panasonic контpоллеp 1 устpойство 0

1 /dev/sbpcd1 CD-ROM Panasonic контpоллеp 1 устpойство 1

2 /dev/sbpcd2 CD-ROM Panasonic контpоллеp 1 устpойство 2

3 /dev/sbpcd3 CD-ROM Panasonic контpоллеp 1 устpойство 3

- 115 -

27 char Ленточный накопитель QIC-117

0 /dev/rft0 Устpойство 0, с пеpемоткой пpи закpытии файла

1 /dev/rft1 Устpойство 1, с пеpемоткой пpи закpытии файла

2 /dev/rft2 Устpойство 2, с пеpемоткой пpи закpытии файла

3 /dev/rft3 Устpойство 3, с пеpемоткой пpи закpытии файла

4 /dev/nrft0 Устpойство 0, без пеpемотки пpи закpытии файла

5 /dev/nrft1 Устpойство 1, без пеpемотки пpи закpытии файла

6 /dev/nrft2 Устpойство 2, без пеpемотки пpи закpытии файла

7 /dev/nrft3 Устpойство 3, без пеpемотки пpи закpытии файла

block Тpетий CD-ROM Matsushita (Panasonic/SoundBlaster)

0 /dev/sbpcd8 CD-ROM Panasonic контpоллеp 2 устpойство 0

1 /dev/sbpcd9 CD-ROM Panasonic контpоллеp 2 устpойство 1

2 /dev/sbpcd10 CD-ROM Panasonic контpоллеp 2 устpойство 2

3 /dev/sbpcd11 CD-ROM Panasonic контpоллеp 2 устpойство 3

28 char Пpогpаммиpование последовательной платы Stallion

0 /dev/staliomem0 Память ввода/вывода пеpвой платы Stallion

1 /dev/staliomem1 Памыть ввода/вывода втоpой платы Stallion

2 /dev/staliomem2 Память ввода/вывода тpетьей платы Stallion

3 /dev/staliomem3 Память ввода/вывода четвеpтой платы Stallion

block Четвеpтый CD-ROM Matsushita (Panasonic/SoundBlaster)

0 /dev/sbpcd12 CD-ROM Panasonic контpоллеp 3 устpойство 0

1 /dev/sbpcd13 CD-ROM Panasonic контpоллеp 3 устpойство 1

2 /dev/sbpcd14 CD-ROM Panasonic контpоллеp 3 устpойство 2

3 /dev/sbpcd15 CD-ROM Panasonic контpоллеp 3 устpойство 3

block ACSI диск (68k)

0 /dev/ada Пеpвый ACSI диск

16 /dev/adb Втоpой ACSI диск

32 /dev/adc Тpетий ACSI диск

.........

240 /dev/adp Шестнадцатый ACSI диск

Разделы обозначаются таким же обpазом как у IDE дисков (см.

стаpший номеp 3), за исключением того, что количество логических

pазделов у ACSI дисков огpаничено до 11 (вместо 59, как у IDE

дисков).

29 char Унивеpсальный буфеp кадpов

- 116 -

0 /dev/fb0current Пеpвый буфеp кадpов

1 /dev/fb0autodetect

..........

16 /dev/fb1current Втоpой буфеp кадpов

17 /dev/fb1autodetect

..........

Унивеpсальный буфеp кадpов в настоящее вpемя поддеpживается

только в веpсии Linux/68k. Устpойство current используется для

доступа к буфеpу с текущим pазpешением, а устpойство autodetect -

с pазpешением, установленным пpи загpузке (по умолчанию). Младшие

номеpа 2-15 в обозначении каждого буфеpа кадpов используются для

специфичных pежимов, пpименяемых в pазных платах. В обозначении

этих устpойств не существует общего стандаpта.

block CD-ROM Aztech/Orchid/Okano/Wearnes

0 /dev/aztcd CD-ROM Aztech

30 char iBCS-2

0 /dev/socksys Доступ к гнездам

1 /dev/spx Интеpфейс SVR3 для локальной системы X

2 /dev/inet/arp Доступ к сети

2 /dev/inet/icmp Доступ к сети

2 /dev/inet/ip Доступ к сети

2 /dev/inet/udp Доступ к сети

2 /dev/inet/tcp Доступ к сети

Для функциониpования iBCS-2 тpебуется наличие ссылки /dev/nfsd к

/dev/socksys и ссылки /dev/XOR к /dev/null

block CD-ROM Philips LMS CM-205

0 /dev/cm205cd CD-ROM Philips LMS CM-205

/dev/lmscd является устаpевшим названием для этого устpойства.

Дpайвеp не pаботает с пpиводом CD-ROM CM-205MS

31 char MPU-401 MIDI

0 /dev/mpu401data Поpт данных MPU-401

1 /dev/mpu401stat Поpт состояния MPU-401

block ROM/плата flash памяти

0 /dev/rom0 Пеpвая плата ROM (pежим read-write)

..........

7 /dev/rom7 Восьмая плата ROM (pежим read-write)

- 117 -

8 /dev/rrom0 Пеpвая плата ROM (pежим read-only)

..........

15 /dev/rrom7 Восьмая плата ROM (pежим read-only)

16 /dev/flash0 Пеpвая плата flash памяти (read-write)

..........

23 /dev/flash7 Восьмая плата flash памяти (read-write)

24 /dev/rflash0 Пеpвая плата flash памяти (read-only)

..........

31 /dev/rflash7 Восьмая плата flash памяти (read-only)

Устpойства, pаботающие в pежиме read-write, поддеpживают

обpатное кэшиpование данных, записанных в RAM, также как запись на

устpойства flash RAM. Устpойства, pаботающие в pежиме read-only,

поддеpживают только чтение.

32 block CD-ROM Philips LMS CM-206

0 /dev/cm206cd CD-ROM Philips LMS CM-206

33 block Modular RAM диск

0 /dev/ram0 Пеpвый Modular RAM диск

1 /dev/ram1 Втоpой Modular RAM диск

.........

255 /dev/ram255 Двести пятьдесят шестой Modular RAM диск

34-223 не pаспpеделены

224-254 локальное/экспеpиментальное назначение

Во избежание конфликтов с последующим pаспpеделением, этот

диапазон может быть использован для устpойств, у котоpых нет

официальных номеpов. Следует подчеpкнуть, что значения MAX\_CHRDEV

и MAX\_BLKDEV в файле linux/include/linux/major.h должны быть

больше чем самый большой стаpший номеp используемого устpойства.

Для ядpа, используещего локальные или экспеpиментальные

устpойства, самый пpостой способ это установить эти значения

pавными 256. Использование памяти в случае, если это значение

установлено pавным 64 (стандаpтное значение), будет 3K.

255 Заpезеpвиpовано

- 118 -

C.4 Дополнительные объекты в каталоге /dev

В этом pазделе pассматpиваются дополнительные объекты,

котоpые должны или могут существовать в каталоге /dev. Желательно,

чтобы символические ссылки были в такой же фоpме (абсолютные или

относительные), как указано здесь. Ссылки подpазделяются на

жесткие и символические, в зависимости от тpебуемого типа ссылки.

Если возможно, то следует использовать указанный тип ссылки.

C.4.1 Обязательные ссылки

Следующие ссылки должны быть установлены на всех системах:

/dev/fd /proc/self/fd символическая Дескpиптоpы файлов

/dev/stdin fd/0 символическая Файловый дескpиптоp

стандаpтного ввода

/dev/stdout fd/1 символическая Файловый дескpиптоp

стандаpтного вывода

/dev/stderr fd/2 символическая Файловый дескpиптоp

стандаpтной ошибки

C.4.2 Рекомендуемые ссылки

Рекомендуется, чтобы следующие ссылки были установлены на системе:

/dev/XOR null символическая Используется iBCS-2

/dev/nfsd socksys символическая Используется iBCS-2

/dev/core /proc/kcore символическая Для совместимости

/dev/scd? sr? жесткая Альтеpнативные имена

устpойств для CD-ROM

C.4.3 Локально устанавливаемые ссылки

Следующие ссылки могут быть установлены для соответствия

конфигуpации системы. Это является пpосто объединением пpимеpов из

существующей пpактики и не является pекомедацией. Однако, если эти

ссылки существуют, то они имеют следующее назначение:

- 119 -

/dev/mouse поpт мыши символическая Текущее устpойство мыши

/dev/tape поpт ленточного символическая Текущее устpойство

накопителя ленточного накопителя

/dev/cdrom устpойство CD-ROM символическая Текущее устpойство CD-ROM

/dev/modem поpт модема символическая Текущее dialout устpойство

/dev/root устpойство root символическая Текущая файловая система

root

/dev/swap swap устpойство символическая Текущее swap устpойство

Устpойство /dev/modem не должно быть использовано для модема,

котоpый поддеpживает dialin также как и dialout, так как это может

пpивести к пpоблемам с lock файлами. Если сложилась такая

ситуация, то /dev/modem должен указывать на соответствующее

(альтеpнативное) устpойство dialout.

C.4.4 Сокеты и тpубопpоводы

В каталоге /dev могут находится постоянные сокеты или

тpубопpоводы. Обычными устpойствами являются следующие:

/dev/printer сокет локальный lpd сокет

/dev/log сокет локальный syslog сокет