OBSD.RU

14 Работа с дисками

printable page

Использование disklabel(8) OpenBSD

Что такое disklabel(8)?

Сначала следует прочитать руководство по disklabel(8).

Подробности настройки дисков в OpenBSD несколько меняются от платформы к платформе. На i386, amd64, macppc, zaurus и cats настройка диска делается в два приёма. Сначала выделяется квант (slice) под OpenBSD посредством fdisk(8), который затем делится на разделы OpenBSD посредством disklabel(8).

Однако, OpenBSD на всех платформах используют disklabel(8) как основное средство управления разделами OpenBSD. На платформах, на которых используется fdisk(8), разделы disklabel(8) располагаются в одном разделе fdisk.

Таблицы разделов содержат определённую информацию о диске, такую как геометрия и информация о файловых системах. Они также содержат информацию о самом диске - скорость вращения, чередование и т.д. - по чисто историческим соображениям и часто эта информация неверна. Не стоит об этом беспокоиться. Таблица разделов используется программой инициализации для доступа к устройству, определения местонахождения файловых систем. Более подробная информация о таблицах разделов - в руководстве disklabel(5).

На некоторых платформах, disklabel позволяет преодолеть архитектурные ограничения при разбиении диска. Например, на i386, бывает 4 основных раздела, но при помощи disklabel(8), можно использовать один из этих основных разделов для хранения в нём ВСЕХ разделов OpenBSD ('swap', '/', '/usr', '/var', etc.), и оставить 3 раздела доступными для других операционных систем.

disklabel(8) во время установки OpenBSD

Одна из основных частей процесса установки OpenBSD - создание разделов. Во время установки, используется disklabel(8) для создания отдельных разделов. Задать точки монтирования можно прямо из disklabel(8), хотя их можно с таким же успехом изменить и во время и после установки.

Нет единственного правильного пути разбиения диска, но есть много путей неправильных. Перед разбиентем диска, следует изучить FAQ4 - обсуждение разбиения и размеров разделов.

См. раздел [AQ4 | "Установка дисков" в FAQ4 | руководстве по установке в качестве примера использования disklabel(8) в процессе установки

Использование disklabel(8) после установки

После установки один из обычных поводов использования disklabel(8) - просмотр информации о структуре диска. Следующие команды отображают таблицу разделов без их изменения:

```
# disklabel wd0 <-- Or whatever disk device you'd like to view
# Inside MBR partition 3: type A6 start 63 size 29880837
# /dev/rwd0c:
type: ESDI
disk: ESDI/IDE disk
label: Maxtor 51536H2
flags:
bytes/sector: 512
sectors/track: 63
tracks/cylinder: 16
sectors/cylinder: 1008
cylinders: 16383
total sectors: 29888820
rpm: 3600
interleave: 1
trackskew: 0
cylinderskew: 0
headswitch: 0           # microseconds
track-to-track seek: 0  # microseconds
drivedata: 0

16 partitions:
#             size        offset  fstype [fsize bsize  cpg]
  a:        614817            63  4.2BSD   2048 16384  328 # Cyl     0*-   609
  b:        409248        614880    swap                   # Cyl   610 -  1015
  c:      29888820             0  unused      0     0      # Cyl     0 - 29651*
  d:       6291936       1024128  4.2BSD   2048 16384  328 # Cyl  1016 -  7257
  e:        409248       7316064  4.2BSD   2048 16384  328 # Cyl  7258 -  7663
 f:       1024128       9822960  4.2BSD   2048 16384  328 # Cyl  9745 - 10760
 h:       2097648       7725312  4.2BSD   2048 16384  328 # Cyl  7664 -  9744
```

Обратите внимание, этот диск в данный момент распределён не полностью. Disklabel позволяет редактировать таблицу разделов в одном из двух режимов - встроенного командного редактора (вызывается, например, при установке OpenBSD) или обычного редактора, такого как vi(1). Может показаться, что командный редактор "легче" - пошаговые действия, система помощи, но экранный редактор тоже бывает полезен.

Добавим раздел к представленной выше системе.

ВНИМАНИЕ: "Балуясь" с таблицей разделов, вы каждый раз рискуете всеми данными на диске. Убедитесь, что созданы резервные копии данных до изменения существующей!

Используем командный редактор, вызываемый флагом "-E" при запуске disklabel(8).

```
# disklabel -E wd0
...
> a k
offset: [10847088]
size: [19033812] 2g
Rounding to nearest cylinder: 4194288
FS type: [4.2BSD]
> p m
device: /dev/rwd0c
type: ESDI
disk: ESDI/IDE disk
label: Maxtor 51536H2
bytes/sector: 512
sectors/track: 63
tracks/cylinder: 16
sectors/cylinder: 1008
cylinders: 16383
total bytes: 14594.2M
free bytes: 7245.9M
rpm: 3600

16 partitions:
#             size        offset  fstype [fsize bsize  cpg]
  a:        300.2M          0.0M  4.2BSD   2048 16384  328 # Cyl     0*-   609
  b:        199.8M        300.2M    swap                    # Cyl   610 -  1015
  c:      14594.2M          0.0M  unused      0     0       # Cyl     0 - 29651*
  d:       3072.2M        500.1M  4.2BSD   2048 16384  328 # Cyl  1016 -  7257
  e:        199.8M       3572.3M  4.2BSD   2048 16384  328 # Cyl  7258 -  7663
  f:        500.1M       4796.4M  4.2BSD   2048 16384  328 # Cyl  9745 - 10760
  h:       1024.2M       3772.1M  4.2BSD   2048 16384  328 # Cyl  7664 -  9744
  k:       2048.0M       5296.4M  4.2BSD   2048 16384   16 # Cyl 10761 - 14921
> q
Write new label?: [y]
```

В данном случае, disklabel(8) любезно посчитал смещение для раздела. В большинстве случаев, он справится,

но если в диске есть "дыры" (например, разделы удалялись или вы - любитель приключений), может понадобиться посидеть с карандашом и бумагой чтобы высчитать нужное смещение. Хоть disklabel(8) и делает некоторые проверки на работоспособность таблицы, здесь можно напартачить. Необходимо быть осторожным и понимать смысл вводимых чисел.

На большинстве платформ, для OpenBSD доступны шестнадцать разделов в таблице разделов диска. Они помечены латинскими буквами от "a" до "p" (некоторые особые системы могут поддерживать только 8 разделов). Любая таблица разделов должна содержать раздел "c" с файловой системой "unused", которая содержит всё физическое устройство. Если таблица разделов отличается, это необходимо исправить - опция "D" (см. ниже) поможет в этом. Не следует пытаться использовать раздел "c" для чего угодно кроме прямого доступа к секторам диска. Не следует пытаться создавать файловую систему в разделе "c". На загрузочном диске, раздел "a" зарезервирован для корневого раздела, а "b" - для раздела подкачки, но это только н азагрузочном устройстве - остальные устройства могут использовать все пятнадцать разделов кроме раздела "c" для файловых систем.

Disklabel - советы и рекомендации

Изучите помощь: В режиме коммандной строки "?" выводит на экран список доступных комманд. "M" показывает страницы руководства по disklabel(8).

Сброс по умолчанию: В некоторых случаях вы можете захотеть начать все заново и удалить всю существующую информацию о таблице разделов. "D" сбрасывает таблицу разделов в состояние по умолчанию, так как будто на диске ее никогда и не было.

Копирование таблицы разделов: В некоторых случаях вам может понадобиться скопировать таблицу разделов с одного диска на другой, но не один в один (например, вы хотите получить одинаковые разделы на дисках разного размера). Используйте disklabel(8) в режиме '-e' (полноэкранный редактор), чтобы скопировать разделы с эталонного диска, вставьте ее на новый диск, удалите раздел 'c', оставшийся от эталонного диска из таблицы разделов, сохранитесь. Вы скопировали таблицу разделов на другой диск, не изменяя ее базовых параметров.

(sparc/sparc64) **Не располагайте swap в самом начале диска.**

(i386, amd64) **Оставьте первую дорожку пустой**: На некоторых платформах вы должны оставить первую логическую дорожку не использованной ни disklabel(8) ни fdisk(8). Эта рекомендация иногда видоизменяется в "начинайте разделы с 63 сектора", но это верно ТОЛЬКО если это размер дорожки вашего винчестера. Не допустите ошибку, это не всегда верно, disklabel укажет, сколько по его мнению приходится секторов на дорожку. Многие другие платформы ожидают начало OpenBSD разделов в 0-м секторе.

Устройства без разделов: Если устройство не имеет OpenBSD разделов, но на нем есть другая файловая система (например, диск с существующей системой FAT32), ядро OpenBSD "создает" их в памяти, и это формирует формальную таблицу разделов OpenBSD на диске. В другом случае, если таблица разделов создана и сохранена на диске, и не-OpenBSD файловая система добавлена позже, таблица разделов не будет автоматически обновлена. Вы должны сделать это самостоятельно, если хотите чтобы OpenBSD могла получить доступ к этой файловой системе. Более подробно об этом ниже.

"q" против "x": По историческим причинам, в режиме коммандной строки "q" сохраняет изменения и выходит из программы, "x" выходит без сохранения. Это противоположно тому, что многие люди используют в настоящее время. disklabel(8) предупреждает перед сохранением изменений, хотя по команде "x" выходит быстро и тихо.

14.2 - Использование fdisk(8)

Прочитайте руководство по fdisk(8).

fdisk(8) используется на некоторых платформах (i386, amd64, macppc, zaurus and cats) для создания разделов, распознаваемых системным загрузчиком, в которые могут быть записаны OpenBSD`шные разделы disklabel. Другие платформы не нуждаются в использовании fdisk(8). fdisk(8) может также быть использован для манипуляции с Главной Загрузочной Записью (MBR), что может повлиять на все установленные на компьютер операционные системы. В отличие от fdisk-подобных программ в других операционных системах, fdisk в OpenBSD подразумевает, что вы знаете что делаете и, в большинстве случаев, позволяет вам делать то, что хотите, являясь мощным инструментом в умелых руках. Он также позволит вам сделать вещи, которые вы не должны были или не собирались делать, так что он должен использоваться с острожностью.

Нормально, только один OpenBSD fdisk раздел создается на диске. Этот раздел делится с помощью disklabel на разделы файловой системы OpenBSD.

Чтобы просто просмотреть вашу таблицу разделов при помощи fdisk используйте:

```
# fdisk sd0
```

Что выведет на экран что то похожее на это:

```
Disk: sd0       geometry: 553/255/63 [8883945 Sectors]
Offset: 0       Signature: 0xAA55
         Starting       Ending       LBA Info:
 #: id    C   H  S -    C   H  S [       start:      size   ]
------------------------------------------------------------------------
*0: A6    3   0  1 -  552 254 63 [       48195:     8835750 ] OpenBSD
 1: 12    0   1  1 -    2 254 63 [          63:       48132 ] Compaq Diag.
 2: 00    0   0  0 -    0   0  0 [           0:           0 ] unused
 3: 00    0   0  0 -    0   0  0 [           0:           0 ] unused
```

В этом примере мы видим вывод fdisk`а для первого SCSI диска. Мы можем видеть раздел OpenBSD (A6) и его размер. * показывает, что раздел OpenBSD является загрузочным.

В предыдущем примере мы просто просмотрели информацию. Что если мы захотим отредактировать таблицу размещения? Чтобы сделать это мы должны использовать флаг "-e". Это запустит командную строку fdisk.

```
# fdisk -e wd0
Enter 'help' for information
fdisk: 1> help
       help            Command help list
       manual          Show entire OpenBSD man page for fdisk
```

```
        reinit          Re-initialize loaded MBR (to defaults)
        setpid          Set the identifier of a given table entry
        disk            Edit current drive stats
        edit            Edit given table entry
        flag            Flag given table entry as bootable
        update          Update machine code in loaded MBR
        select          Select extended partition table entry MBR
        swap            Swap two partition entries
        print           Print loaded MBR partition table
        write           Write loaded MBR to disk
        exit            Exit edit of current MBR, without saving changes
        quit            Quit edit of current MBR, saving current changes
        abort           Abort program without saving current changes
fdisk: 1>
```

Вот обзор комманд, которые вы можете использовать, когда используете флаг "**-e**".

help Показывает список комманд, которые fdisk понимает в интерактивном режиме.
reinit Инициализирует выбранную в данный момент находящуюся в памяти копию загрузочного блока. Это удобный способ быстро создать полнодисковый раздел OpenBSD, обновить код загрузки и подготовить систему к установке OpenBSD (и ничего кроме OpenBSD).
disk Отображает текущую геометрию диска, распознанную fdisk. У вас есть шанс поменять ее, если хотите.
setpid Изменяет идентификатор раздела заданной записи таблицы размещения разделов. Команда полезна для переназначения существующего раздела под OpenBSD.
edit Изменяет заданную запись в таблице находящейся в памяти копии текущего загрузочного блока. Вы можете редактировать в режиме геометрии BIOS, или в секторах, смещениях и размерах.
flag Делает заданный раздел загрузочным. Только один раздел может быть помечен как загрузочный. Если хотите загружаться с дополнительного (extended) раздела, вы должны пометить запись дополнительного (extended) раздела как загрузочную.
update Обновить машинные коды в находящейся в памяти копии текущего загрузочного блока.
select Выбирает и загружает в память загрузочный блок указанный в дополнительном (extended) разделе в текущем загрузочном блоке.
swap Свопит две главных загрузочных записи (MBR), так что вы можете поменять местами MBR.
print Распечатывает выбранную в данный момент находящуюся в памяти копию загрузочного блока и ее MBR в терминал.
write Записывает находящуюся в памяти копию загрузочного блока на диск. Потребуется подтверждение операции.
exit Выходит с текущего уровня fdisk, возвращаясь в до этого выбранную находящуюся в памяти копию загрузочного блока, или завершая программу, в противном случае.
quit Выходит с текущего уровня fdisk, возвращаясь в до этого выбранную находящуюся в памяти копию загрузочного блока, или завершая программу, в противном случае. В отличие от "exit" записывает на диск модифицированный блок.
abort Выходит из программы без сохранения изменений.

fdisk - советы и рекомендации

fdisk(8) предлагает возможность редактировать разделы и в raw sector режиме и в CHS формате. Обе опции доступны по следующей причине — некоторые задачи легче выполняются одним способом, другие — другим. Не ограничивайте себя использованием только одной опции.

Полностью чистый диск нуждается в записи загрузочного кода в MBR для того, чтобы с него можно было загружаться. Вы можете использовать "reinit" или "update" чтобы сделать это. Если вы не сделаете этого, вы можете записать верную таблицу размещения разделов с помощью fdisk, но у вас не будет загрузочного диска. Вы можете захотеть обновить существующий загрузочный код в случае если на уверены в его происхождении.

Если у вас в системе есть "служебный" или "диагностический" раздел, рекомендуется оставить его на месте или устанавливать его ДО установки OpenBSD.

По историческим причинам "q" сохраняет изменения и выходит из программы, а "x" выходит без сохранения, в противоположность тому, что многие люди используют в других средах. fdisk(8) не предупреждает перед сохранением изменений, так что используйте его с осторожностью.

14.3 - Добавление дополнительных дисков в OpenBSD

Теперь после того, как вы ПРАВИЛЬНО настроили свой жесткий диск, вам потребуются утилиты fdisk(8) (только для архитектуры i386) и disklabel(8) для того, чтобы использовать ваш диск в OpenBSD.

Для пользователей i386 все начинается с fdisk'a. Остальные могут это смело проигнорировать. Ниже приведен пример, где мы добавляем третий SCSI-привод в систему.

```
# fdisk -i sd2
```

Данное действие определит "реальную" таблицу разделов для эксклюзивного использования системой OpenBSD. Далее вам необходимо создать раздел. Кого-то это может сбить с толку.

```
# disklabel -e sd2

(screen goes blank, your $EDITOR comes up)
type: SCSI
...bla...
sectors/track: 63
total sectors: 6185088
...bla...
16 partitions:
#        size   offset    fstype   [fsize bsize   cpg]
  c:  6185088        0    unused        0     0         # (Cyl.    0 - 6135)
  d:  1405080       63    4.2BSD     1024  8192    16   # (Cyl.    0*- 1393*)
  e:  4779945  1405143    4.2BSD     1024  8192    16   # (Cyl. 1393*- 6135)
```

Первое, не обращайте внимания на раздел 'c', он всегда присутствует и предназначен для нормального функционирования программ типа disklabel! Тип файловой системы (fstype) для OpenBSD - 4.2BSD. Общее количество секторов - это общий объем диска. Сказано, что есть жесткий диск объемом 3 Гигабайта. Три гигабайта в терминах производителя равны 3000 Мегабайт. Так что разделив 6185088/3000 (с помощью bc(1)), вы получите 2061. Теперь, для определения размеров разделов a, d, e, f, g, ... просто умножьте X*2061 - и вы

получите X Мегабайт места для раздела. Смещение вашего нового первого раздела должно совпадать со значением "секторов/дорожек" из результатов disklabel. Для нашей ситуации это 63. Смещение для каждого следующего раздела должно складываться из размера и смещения раздела (Кроме раздела 'c', он никоим образом не участвует в этих вычислениях.)

Или, если вы просто хотите оставить один раздел, просто отвечаете "использовать все доступное место" в качестве хранилища веб-содержимого, домашней директории или чего-либо еще, тогда отнимите из общего размера диска количество треков (дорожек). 6185088-63 = 6185025. Ваш раздел

```
d:  6185025       63    4.2BSD     1024  8192    16
```

Если вам совсем не хочется всех этих сложностей, вы можете просто выполнить команду disklabel -E для получения такого же режима разбиения на разделы, как и на вашем загрузочном диске! Здесь вы можете задать "96 Мегабайт" просто набрав "96M", или, если у вас достаточно большой жесткий диск, 96G для задания 96 Гигабайт.

Этого уже достаточно. Но вы еще не закончили. Наконец, вам нужно создать файловую систему на вашем диске, используя утилиту newfs(8).

```
# newfs sd2a
```

Или укажите имя своего диска в системе нумерации имен OpenBSD's. (Смотрите выходные данные утилиты dmesg(8), чтобы узнать, как OpenBSD назвала ваш диск.)

Теперь вы монтируете этот только что созданный раздел. Скажем, вы хотите поместить его в /u. Сначала создайте директорию /u. Затем, смонтируйте раздел в эту директорию.

```
# mount /dev/sd2a /u
```

Наконец, добавьте раздел в /etc/fstab(5).

```
/dev/sd2a /u ffs rw 1 1
```

Что если вам нужно перенести уже существующую директорию /usr/local? Вы должны смонтировать новый диск в /mnt и, используя cpio -pdum, скопировать /usr/local в директорию /mnt. Отредактируйте файл /etc/fstab(5) чтобы показать, что /usr/local теперь находится в /dev/sd2a (ваш новый отформатированный раздел.) Пример:

```
/dev/sd2a /usr/local ffs rw 1 1
```

Перезагрузитесь в однопользовательском режиме с помощью boot -s, переместите существующую /usr/local в usr/local-backup (или удалите, если верите в свою удачу) и создайте пустую директорию /usr/local. Затем, перезагрузите систему, и, вуаля! - файлы оказались там, где нужно!

14.4 - Как организовать своппинг в файл

(Замечание: Если вы хотите организовать своппинг в файл по причине получения сообщения об ошибке "virtual memory exhausted", вам сначала следует попробовать установить лимиты на процесссы с помощью: в csh - unlimit(1), в sh - ulimit(1).)

Своппинг в файл не требует сборки собственного ядра, хотя можно поступить и так, здесь мы вам рассскажем как увеличить емкость swap в обоих случаях.

Своппинг в файл

.

Своппинг в файл - это самый простой и самый быстрый способ получения дополнительной емкости swap. Этот файл не может находиться на файловой системе со включенным SoftUpdates (по умолчанию они выключены). Для начала давайте выясним, какой у нас размер swap и на сколько он используется, для этого выполним swapctl(8). Выполним команду для этого:

```
$ swapctl -l
Device      512-blocks     Used    Avail Capacity  Priority
swap_device      65520        8    65512     0%    0
```

Вывод нам показывает, что у нас в настоящий момент используется для своппинга и статистику его использования. В данном примере - только одно устройство "swap_device". Это та область на диске, которая используется по умолчанию (Та самая, которую мы видим в выводе disklabel как раздел b). В вышеприведенном примере мы видим, что его размер значительно больше требуемого системой. Однако, для примера, будем считать, что нам требуется добавить 32M.
Первым шагом для использования файла как устройство для своппинга создадим сам файл. Лучше всего это сделать с помощью dd(1). Приведем пример создания файла */var/swap* размером 32M.

```
$ sudo dd if=/dev/zero of=/var/swap bs=1k count=32768
32768+0 records in
32768+0 records out
33554432 bytes transferred in 20 secs (1677721 bytes/sec)
```

Теперь мы должны подключить этот файл к устройствам своппинга. Приведенная команда это сделает:

```
$ sudo chmod 600 /var/swap
$ sudo swapctl -a /var/swap
```

Проверим корректность добавления файла к устройствам своппинга.

```
$ swapctl -l
Device      512-blocks     Used    Avail Capacity  Priority
swap_device      65520        8    65512     0%    0
/var/swap        65536        0    65536     0%    0
Total           131056        8   131048     0%
```

Теперь, когда мы подготовили использование файла для своппинга, вам надо будет добавить строку в */etc/fstab* для использования после перезагрузки. Если вы не добавите эту строку ваше устройства своппинга не будет отконфигурировано.

```
$ cat /etc/fstab
/dev/wd0a / ffs rw 1 1
/var/swap /var/swap swap sw 0 0
```

Своппинг используя устройство vnode

Это решение для более длительного использования и оно может добавить больше емкости для своппинга. Для постоянного свопинга в файл сначала надо собрать ядро для использования vnd0c как своп. Если ваша корневая система wd0a, а предыдущий своп wd0b, для конфигурации ядра используйте следующую строку (если не знаете как или есть сомнения - читайте раздел о сборке ядра):

```
config          bsd     root on wd0a swap on wd0b and vnd0c dumps on wd0b
```

Как это будет сделано, надо создать файл для своппинга. Той же командой, как в примере выше.

```
$ sudo dd if=/dev/zero of=/var/swap bs=1k count=32768
32768+0 records in
32768+0 records out
33554432 bytes transferred in 20 secs (1677721 bytes/sec)
```

Файл создан, конфигурируем */etc/fstab*. Ниже прмиер, как отконфигурировать для использования файла для свопинга после перезагрузки.

```
$ cat /etc/fstab
/dev/wd0a / ffs rw 1 1
/dev/vnd0c none swap sw 0 0
```

Теперь компьютер придется перегрузить для загрузки нового ядра. Когда все выполнено, нам надо отконфигурировать наше виртуальное устройство для своппинга. Для этого используйте vnconfig(8).

```
$ sudo vnconfig -c -v vnd0 /var/swap
```

```
vnd0: 33554432 bytes on /var/swap
```

И на последнем этапе подключим это устройство для своппинга. Как и в в примерах выше, используем swapctl(8). Ну и затем проверим, все ли корректно.

```
$ sudo swapctl -a /dev/vnd0c
$ swapctl -l
Device      512-blocks     Used    Avail Capacity  Priority
swap_device      65520        8    65512     0%    0
/dev/vnd0c       65536        0    65536     0%    0
Total           131056        8   131048     0%
```

Soft Update

Идея Soft Updates принадлежит Greg Ganger и Yale Patt, разработка для FreeBSD - Kirk McKusick. Soft Updates частично меняют порядок операций с буферным кэшем без удаления кода FFS синхронизации записи в каталоги. Таким образом достигается значительное ускорение операций записи.

Включение механизма soft updates должно происходить на этапе монтирования. При монтировании с помощью mount(8), вы должны указать необходимость применения soft updates для данного раздела. Ниже пример записи в /etc/fstab(5) для включения soft updates для одного раздела sd0a.

/dev/sd0a / ffs rw,softdep 1 1

Примечание для пользователей sparc: не включайте soft updates на sun4 или sun4c. Эти архитекруры сильно ограничивают размер памяти для ядра и данная возможность не может быть включена. Однако, на sun4m все хорошо работает.

Как загружается OpenBSD/i386?

Процесс загрузки в OpenBSD/i386 нетривиален, и понимание его работы может быть полезным в разрешении проблем, когда что-то не работает. Существует четыре ключевых для процесса загрузки компонента:

1. ***Главная загрузочная запись (Master Boot Record (MBR))***: Главная загрузочная запись - это первый физический сектор (512 байт) на диске. Он содержит первичную таблицу разделов и маленькую программу для загрузки Partition Boot Record (PBR) - загрузочной записи раздела. Заметим, что в некоторых средах термин "MBR" используется только для обозначения кодовой части этого первого блока на диске вместо всего первого блока (включающего и таблицу разделов). Критически важно понимать значение "инициализировать MBR" (initialize the MBR) -- в терминологии OpenBSD вызовет перезапись всего MBR сектора, а не просто кода, как это возможно в некоторых системах. Вам нечасто придётся это делать. Вместо этого используйте опцию "-u" командной строки fdisk(8) ("fdisk -u wd0").

Хотя OpenBSD и включает в себя MBR, вовсе не обязательно использовать именно его, поскольку практически любой MBR может загрузить OpenBSD. Управление MBR осуществляется программой fdisk(8), которая используется для редактирования таблицы разделов. Она также может установить MBR на диск.

MBR OpenBSD информирует о своей работе следующим сообщением

```
Using drive 0, partition 3.
```

указывая с какого дисчка и с какого раздела будет производиться загрузка PBR. В дополнение к очевидной информации в конце строки выводится ("."), которая говорит о том, что на данной машине может использоваться LBA для загрузки. Если машина не позволяет использование LBA, вместо точки будет выводиться (";"), сообщающая об использовании CHS:

```
Using Drive 0, Partition 3;
```

Заметим, что точка или точка с запятой используются только в "новом" MBR OpenBSD, начиная с OpenBSD 3.5.

2. ***Partition Boot Record (PBR)***: Partition Boot Record, также называемый PBR или biosboot(8) - первый физический сектор раздела OpenBSD диска (после содержащего код имени файла). PBR есть "загрузчик первой ступени" для OpenBSD. Он загружается MBR и сам, в свою очередь, загружает загрузчик OpenBSD второй ступени, boot(8). Как и MBR, PBR очень невелик, всего только 512 байт. Этого недостаточно для работы полнофункционального приложения для работы с файловой системой, поэтому PBR находится в отдельном BIOS-доступном разделе диска /boot.

PBR устанавливается installboot(8), который будет описан в данном руководстве ниже. PBR сообщает о своей работе

```
Loading...
```

выводя точку при каждой попытке чтения блока файловой системы. Затем PBR сообщает, используется ли LBA или CHS для загрузки, то есть когда используется CHS сообщение будет выглядеть так:

```
Loading;...
```

Старая версия (ранее v3.5) biosboot(8) выводила сообщение "reading boot...".

3. ***Загрузчик второго уровня, /boot***: /boot загружается PBR, задача которого обеспечить доступ к файловой системе OpenBSD средствами аппаратного BIOS, обнаружения местонахождения и загрузки ядра. boot(8) передает опции загрузки и информацию ядру.

boot(8) программа интерактивная. После загрузки он пытается найти и прочитать /etc/boot.conf, если он есть (что установки отличаются от установок по умолчанию) и обрабатывает его инструкции. Когда инструкции /etc/boot.conf выполнены, выдается приглашение:

```
probing: pc0 com0 com1 apm mem[636k 190M a20=on]
disk: fd0 hd0+
>> OpenBSD/i386 BOOT 2.10
```

```
boot>
```

Пользователю (по умолчанию) дается пять секунд на ввод инструкций, и, по истечению времени, исполняются задачи по умолчению: грузится ядро, bsd, с корневого раздела первого жесткого диска. Загрузчик второго уровня обнаруживает (проверяет) устройства, используя средства BIOS (поскольку ядро OpenBSD еще не загружено). Выше мы можем видеть что он обнаружил:

pc0 - стандартная клавиатура и дисплей системы i386.

com0, com1 - Два серийных порта

apm - Функции расширенного управления питанием APM BIOS

636k 190M - Обычное ОЗУ (до 1M) и расширение (свыше 1M)

fd0 hd0+ - Обнаружены дисковые устройства BIOS, в данном случае, один гибкий и один жесткий диски.

Знак '+' после "hd0" сообщает о возможности доступа средствами BIOS к /boot в режиме LBA. Первый раз, при установке, мы можем увидеть знак '*' после обозначения жесткого диска -- это говорит о том, что жесткий не имеет корректного OpenBSD disk label.

4. ***Kernel: /bsd***: Это окончательный этап, загрузка ядра OpenBSD в ОЗУ и полноценный запуск. Только когда ядро занружено OpenBSD сможет работать с аппартными устройствами напрямую, не используя средства BIOS.

Итак, весь процесс загрузки будет выглядеть следующим образом:

```
Using drive 0, partition 3.                      <- MBR
Loading....                                      <- PBR
probing: pc0 com0 com1 apm mem[636k 190M a20=on] <- /boot
disk: fd0 hd0+
>> OpenBSD/i386 BOOT 2.10
boot>
booting hd0a:/bsd 4464500+838332 [58+204240+181750]=0x56cfd0
entry point at 0x100120

[ using 386464 bytes of bsd ELF symbol table ]
Copyright (c) 1982, 1986, 1989, 1991, 1993       <- ЯДРО
        The Regents of the University of California.  All rights reserved.
Copyright (c) 1995-2008 OpenBSD.  All rights reserved.  http://www.OpenBSD.org

OpenBSD 4.3 (GENERIC) #698: Wed Mar 12 11:07:05 MDT 2008
    deraadt@i386.openbsd.org:/usr/src/sys/arch/i386/compile/GENERIC
   ...
```

Что может быть неправильно

Нерабочий/неправильный/несовместимый MBR: Как правило, используемый жесткий диск имеет уже какой-то MBR, но если диск новый или использовался ранеее на другой платформе, И вы не ответили "Yes" на вопрос "Use entire disk" в процессе инсталляции, ваш диск может оказаться с некорректной MBR, и, как следствие, загрузка окажется невозможной, несмотря на корректную таблицу разделов.

Можно установить OpenBSD MBR на ваш диск с помощью fdisk. Загрузитесь с вашего инсталляционного носителя, выберите "Shell" и выполните команду:

```
# fdisk -u wd0
```

Можно установить определенный MBR используя fdisk

```
# fdisk -u -f /usr/mdec/mbr wd0
```

что установит файл /usr/mdec/mbr как системный MBR. Этот файл из поставки OpenBSD является стандартным MBR, который может установить fdisk, но мы в данном случае можем установить и любой другой MBR, на который укажем.

Неправильное местонахождение /boot в PBR: При установке PBR installboot(8) записывает номер блока и смещение /boot's inode в PBR. Поэтому удаление и замена /boot без повторного выполнения installboot(8) сделают загрузку вашей системы невозможной, поскольку несмотря ни на что PBR пытается работать с указывающим на дескриптор файла inode, а там вряд ли будет ваш загрузчик второго уровня! /boot читается вызовами функций BIOS, старые версии PBR весьма чувствительны к работе BIOS. Измения в геометрии диска (перенести, например, диск с компьютера, на котором использовалось CHS на использующий LBA или внести подобные изменения в BIOS)приведут к обращению BIOS к другому месту на диске (различные номера блоков будут указывать на нужное место на диске), потому придется снова запустить installboot(8) перед перезагрузкой системы. Новые (OpenBSD 3.5 и более новые) PBR менее чувствительны к работе.

Поскольку PBR очень мал выводимая информация об ошибках очень лаконична и использует сокращения, и их число мало. Наиболеее частые:

ERR R -- BIOS возвратил ошибку при попытке чтения диска. Наиболее часто происходит из-за невозможности чтения диска.

ERR M -- Неправильный номер magic(5) second-stage bootloader's header. Обычно это происходит из-за чтения НЕ /boot, по причинам некорректного выполнения installboot(8), файл /boot изменен или ваш BIOS не в состоянии произвести чтение с диска большого размера.

Другие сообщения подробно описаны в руководстве biosboot(8).

Для подробной информации о прцессе загрузки на i386:

boot_i386(8)
http://www.ata-atapi.com/hiw.htm Hale Landis' "Как это работает".

14.7 - Какие проблемы бывают при использовании больших файловых систем в OpenBSD?

OpenBSD поддерживает файловые системы FFS и FFS2 (также известные как UFS и UFS2). FFS файловая система, исторически используемая OpenBSD, FFS2 используется начиная с 4.3. Перед тем как рассмотреть вопросы ограничений, накладываемых конкретной системой, мы рассмотрим более общие вопросы.

Следует различать возможности файловой системы и возможности аппаратного обеспечения. Новые диски IDE 250G могут иметь проблемы на старых (до стандартов >137G) интерфейсах (хотя чаще все работает без проблем), а рекоторые очень старые SCSI адаптеры могут иметь проблемы при работе с новыми дисками, некоторые старые BIOS-ы могут сойти с ума от размеров современных дисков. Знайте возможности своего аппарактного обеспечения.

Размер раздела и ограничения на размещение раздела

К сожалению, полностью возможности ОС становятся доступными только после загрузки ядра в память. Процесс загрузки использует (и им ограничен) системным boot-ROM.

Поэтому весь файл ядра /bsd должен находиться в зоне, адресуемой boot-ROM. На некоторых старых i386 весь корневой раздел должен быть в пределах первых 504M, и даже более новые колмпьютеры могут накладывать ограничения в 2G, 8G, 32G, 128G или другие. Следует отметить, что и сравнительно новые компьютеры, поддерживающие диски более 128G на самом деле имеют BIOS с ограниченниями на расположение загрузчика в рамках начальных 128G. Вы можете на них использовать диски большого объема, но ваш корневой раздел должен находиться в зоне ПЗУ загрузчика.

Заметим, что может оказаться возможным установить привод размером 40G на древнюю 486 и загрузить OpenBSD на один большой раздел, посчитав, что мы смогли обойти ограничения. Все это может обернуться совершенно неприятными вещами:

Вы установите все в один 40G / раздел. Это заработает, поскольку вся ваша ОС (включая /bsd) находится в первых 504M.
Вы пользуетесь системой, и е размер становится более 504M.
Делаете upgrade, собираете собственное ядро, устанавливаете новое ядро /bsd вместо старого.
Перегружаетесь.
Получаете что-нибудь вроде "ERR M" или еще какой-то "сюрприз" при загрузке.

Почему? Потому что когды вы записываете новое ядро "поверх" (переписываете) старого, новый файл /bsd отнюдь не записывается на то же место на диске, где был старый, а размещается, вероятнеее всего, за пределами доступных базовой системе ввода-вывода (BIOS) 504M. Загрузчик не может загрузить /bsd и система "зависает".

Чтобы OpenBSD могла загрузиться, boot loaders (biosboot(8) и /boot в случае i386/amd64), ядро (/bsd) должны находиться в поддерживаемой ПЗУ-загрузчиком области, и в рамках его возможностей. Чтобы "кататься на санях", правила просты:

Весь корневой раздел должен быть в видимом базовой системой ввода-вывода компьютера (BIOS) (или boot ROM) адресном пространстве.

Некоторые не-i386 полагают, что защищены от подобного, но большинство boot ROM накладывают ограничения на размер диска. Однако зачастую узнать, какое оно в данном конкретном случае, бывает весьма затруднительно.

Это еще одна причина, согласно которой следует разбивать диск на разделы, а не использовать один большой общий.

fsck(8) требования времени и памяти

Другой момент относительно больших файловых систем - время и ресурсы памяти. которые потребуются для работы по восстановлению файловой системы fsck(8) после аварии электропитания. Не слудует создавать файловые системы размером 120G на компьютере с 32M ОЗУ, поскольку не приходится рассчитывать на корректную работу fsck(8) после аварии. Грубо потребности ОЗУ для проверки после сбоя можно оценить так: 1M доступной памяти на каждый 1G дискового пространства необходимо для работы fsck. Своп может использоваться, но это настолько замедляет процесс, что может оказаться неприемлимым, за исключением крайней необходимости, конечно.

Время работы fsck для проверки диска может стать проблемой по мере роста размеров файловой системы, но имейте ввиду - fsck работает только с дисковым пространством актуальной файловой системы. Это еще одна причина ответа "НЕТ" на отведение всего диского пространства под один раздел. Монтирование файловой системе в режиме "только чтение" (RO) или не (постоянное) монтирование спасает от необходимости запуска fsck(8) после выдергивания кабеля питания.

Не забудьте, что если ваша фавйловая система состоит из нескольких дисков, то после аварии fsck(8) будет работать медленнее и потребует больше памяти, чем для одного диска такорго размера.

Когда размер файловой системы превысит 1TB со размерами фрагментов и блоков fsck для запуска потребует 1GB ОЗУ, что может попасть на ограничения OpenBSD на выделямую под одно приложение память. Увеличение размеров фрагментов и/или блоков приведет к уменьшению числа inodes и позволит пользоваться бОльшими файловыми системами.

FFS vs. FFS2

При использовании FFS OpenBSD поддерживает файловую систему до 231-1, или 2,147,483,647 секторов размером 512 bytes, что позволяет получить почти 1T. FFS2 позволяет гораздо больший размер файловой системы, мы столкнемся в этом случае с другими ограничениями гораздо раньше, нежели лимит файловой системы.

Загрузка/установка ядра может быть произведена только с FFS, не FFS2, системные разделы (/, /usr, /var, /tmp) поэтому не могут быть FFS2, or severe maintenance problems can arise (в любом случае нет никаких оснований делать эти разделы столь большими). По этой причине большие разделы могут быть использованы только для "несистемных" разделов, к примеру, /home, /var/www/, /bigarray, и т.д.

Перед обновлениями (upgrades) вы должны пометить все разделы FFS2 как "noauto" поскольку в данноем случае используется инсталляционное ядро (не поддерживающее разделы FFS2).

Отметим, что не все драйвера и контроллеры поддерживают большие диски. Например, ami(4) имеет ограничение 2TB на логический диск. Многие просто не проверены, к примеру, когда писалась эта документация, у нас не имелось >1TB IDE SATA приводов для тестирования, и мы не можем сказать наверняка, как они будут работать.

14.8 - Установка Bootblocks - i386/amd64 специфика

Современные версии OpenBSD (3.5 и более новые) весьма понятливый системный загрузчик, более терпимый к геометрии диска чем более старый, однако, он чувствителен к расположению /boot на диске. Если ваши действия могут привести к изменению местонахождения boot(8) (конкретно, к новому inode), вы должны принять меры для возможности корректной загрузки. Для правки boot block для нормальной загрузки, загрузитесь с гибкого диска (или загрузочного CD) и при получении приглашения загрузчика, введите "b hd0a:/bsd" для принудельной загрузки с первого жесткого диска (а не гибкого). Машина должна стартовать нормально. Теперь вам надо переустановить загрузчик первой ступени (biosboot(8)) исходя из нахождения файла /boot при помощи программы installboot(8).

В нашем примере для загрузки sd0 (но в случае IDE было бы wd0 и т.д.):

```
# cd /usr/mdec; ./installboot /boot biosboot sd0
```

Если потребуется более новая версия bootblock, еее надо будет собрать самостоятельно. Это просто:

```
# cd /sys/arch/i386/stand/
# make && make install
# cd /usr/mdec; cp ./boot /boot
# ./installboot /boot biosboot sd0 (or whatever device your hard disk is)
```

14.9 - Готовимся к бедствиям: Создание резервных копии и восстановление со стриммера

Введение:

Если вы планируете сервер использовать для серъезных задач, не лишним будет подумать о системе резервного копирования на случай выхода из строя ваших жеских дисков.

Эта информация поможет вам в работе со стандартыми утилитами dump(8)/restore(8), поставляемыми с OpenBSD. Утилита с большими возможностями под названием "Amanda" также доступна в коллекциии пакетов и позволяет делать резервные копии с нескольких серверов на один стриммер. В большинстве случаев вполне хватит возможностей dump(8)/restore(8). Однако, если у вас есть необходимость резервного копирования нескольких машин, обратите внимание на Amanda.

В нашем примере мы расссмотрим конфигурирование для SCSI дисков и магнитной ленты. Для серъезных задач, как правило, используются SCSI диски, а использование IDE не рекомендуется, ввиду их работы с неисправными блоками. Ну, конечно, информация не будет бесполезной для пользователей IDE или других, поменяйте только названия устройств в приведенных примерах и спользуйте на здоровье! Например sd0a на wd0a для IDE.

Резервное копирование на ленту:

Наша задача резервного копирования на ленту первым делом предполагает знание исходных данных, куда смонтированы наши файловые системы (ФС). Узнаем это с помощью команды mount(8). Получим что-то вроде:

```
# mount
/dev/sd0a on / type ffs (local)
/dev/sd0h on /usr type ffs (local)
```

В данном примере корневая ФС (/) физически находится на sd0a, то есть на SCSI 0-м жестком диске, раздел a. ФС /usr размещается на sd0h, SCSI жесткий диск 0, раздел h.

Другой пример, более сложной таблицы разделов:

```
# mount
/dev/sd0a on / type ffs (local)
/dev/sd0d on /var type ffs (local)
/dev/sd0e on /home type ffs (local)
/dev/sd0h on /usr type ffs (local)
```

В этом чуть болеее сложном примере корневая ФС (/) физически находится на sd0a. ФС /var на sd0d, /home на sd0e и /usr на sd0h.

Для создания резервных копии надо указать имя каждого раздела dump. Приводим команды для нашего простой таблицы разделов:

```
# /sbin/dump -0au -f /dev/nrst0 /dev/rsd0a
# /sbin/dump -0au -f /dev/nrst0 /dev/rsd0h
# mt -f /dev/rst0 rewind
```

Для более сложной таблицы разделов команды ниже:

```
# /sbin/dump -0au -f /dev/nrst0 /dev/rsd0a
# /sbin/dump -0au -f /dev/nrst0 /dev/rsd0d
# /sbin/dump -0au -f /dev/nrst0 /dev/rsd0e
# /sbin/dump -0au -f /dev/nrst0 /dev/rsd0h
# mt -f /dev/rst0 rewind
```

На странице руководства dump(8) можно посмотреть, какой параметр за что отвечает. А вот краткое описание использованных нами:

- **0** - Выполнить dump уровня 0, все полностью

a - Попытаться автоматически определить tape media length
u - Обновить файл /etc/dumpdates - когда последний разх осуществлялось резервное копирование
f - Указать используемый стриммер (/dev/nrst0 в нашем случае)

В конце строки указываем, какой раздел резервировать (/dev/rsd0a, и так далеее)

Команда mt(1) в самом конце используется для перемотки ленты. Обратитесь к странице руководства mt для ознакомления с другими параметрами (например, eject).

Если вы не уверены в названии вашего стриммера, используйте dmesg . Пример стриммера в выводе dmesg или что-то вроде:

```
st0 at scsibus0 targ 5 lun 0: <ARCHIVE, Python 28388-XXX, 5.28>
```

Наверное, вы заметили, что мы использовали для доступа к стриммеру "nrst0" вместо выводимого в dmesg "st0". когда мы обращаемся к st0 как nrst0 are accessing the same physical tape drive but telling the drive to not rewind at the end of the job and access the device in raw mode. To back up multiple file systems to a single tape, be sure you use the non-rewind device, if you use a rewind device (rst0) to back up multiple file systems, you'll end up overwriting the prior filesystem with the next one dump tries to write to tape. You can find a more elaborate description of various tape drive devices in the dump man page.

If you wanted to write a small script called "backup", it might look something like this:

```
echo "  Starting Full Backup..."
/sbin/dump -0au -f /dev/nrst0 /dev/rsd0a
/sbin/dump -0au -f /dev/nrst0 /dev/rsd0d
/sbin/dump -0au -f /dev/nrst0 /dev/rsd0e
/sbin/dump -0au -f /dev/nrst0 /dev/rsd0h
echo
echo -n "  Rewinding Drive, Please wait..."
mt -f /dev/rst0 rewind
echo "Done."
echo
```

If scheduled nightly backups are desired, cron(8) could be used to launch your backup script automatically.

It will also be helpful to document (on a scrap of paper) how large each file system needs to be. You can use "df -h" to determine how much space each partition is currently using. This will be handy when the drive fails and you need to recreate your partition table on the new drive.

Restoring your data will also help reduce fragmentation. To ensure you get all files, the best way of backing up is rebooting your system in single user mode. File systems do not need to be mounted to be backed up. Don't forget to mount root (/) r/w after rebooting in single user mode or your dump will fail when trying to write out dumpdates. Enter "bsd -s" at the boot> prompt for single user mode.

Viewing the contents of a dump tape:

After you've backed up your file systems for the first time, it would be a good idea to briefly test your tape and be sure the data on it is as you expect it should be.

You can use the following example to review a catalog of files on a dump tape:

```
# /sbin/restore -tvs 1 -f /dev/rst0
```

This will cause a list of files that exist on the 1st partition of the dump tape to be listed. Following along from the above examples, 1 would be your root (/) file system.

To see what resides on the 2nd tape partition and send the output to a file, you would use a command similar to:

```
# /sbin/restore -tvs 2 -f /dev/rst0 > /home/me/list.txt
```

If you have a mount table like the simple one, 2 would be /usr, if yours is a more advanced mount table 2 might be /var or another fs. The sequence number matches the order in which the file systems are written to tape.

Restoring from tape:

The example scenario listed below would be useful if your fixed drive has failed completely. In the event you want to restore a single file from tape, review the restore man page and pay attention to the interactive mode instructions.

If you have prepared properly, replacing a disk and restoring your data from tape can be a very quick process. The standard OpenBSD install/boot floppy already contains the required restore utility as well as the binaries required to partition and make your new drive bootable. In most cases, this floppy and your most recent dump tape is all you'll need to get back up and running.

After physically replacing the failed disk drive, the basic steps to restore your data are as follows:

Boot from the OpenBSD install/boot floppy. At the menu selection, choose Shell. Write protect and insert your most recent back up tape into the drive.

Using the fdisk(8) command, create a primary OpenBSD partition on this newly installed drive. Example:

```
# fdisk -e sd0
```

See fdisk FAQ for more info.

Using the disklabel command, recreate your OpenBSD partition table inside that primary OpenBSD partition you just created with fdisk. Example:

```
# disklabel -E sd0
```

(Don't forget swap, see disklabel FAQ for more info)

Use the newfs command to build a clean file system on each partition you created in the above step. Example:

```
# newfs /dev/rsd0a
# newfs /dev/rsd0h
```

Mount your newly prepared root (/) file system on /mnt. Example:

```
# mount /dev/sd0a /mnt
```

Change into that mounted root file system and start the restore process. Example:

```
# cd /mnt
# restore -rs 1 -f /dev/rst0
```

You'll want this new disk to be bootable, use the following to write a new MBR to your drive. Example:

```
# fdisk -i sd0
```

In addition to writing a new MBR to the drive, you will need to install boot blocks to boot from it. The following is a brief example:

```
# cp /usr/mdec/boot /mnt/boot
# /usr/mdec/installboot -v /mnt/boot /usr/mdec/biosboot sd0
```

Your new root file system on the fixed disk should be ready enough so you can boot it and continue restoring the rest of your file systems. Since your operating system is not complete yet, be sure you boot back up with single user mode. At the shell prompt, issue the following commands to unmount and halt the system:

```
# umount /mnt
# halt
```

Remove the install/boot floppy from the drive and reboot your system. At the OpenBSD boot> prompt, issue the following command:

```
boot> bsd -s
```

The bsd -s will cause the kernel to be started in single user mode which will only require a root (/) file system.

Assuming you performed the above steps correctly and nothing has gone wrong you should end up at a prompt asking you for a shell path or press return. Press return to use sh. Next, you'll want to remount root in r/w mode as opposed to read only. Issue the following command:

```
# mount -u -w /
```

Once you have re-mounted in r/w mode you can continue restoring your other file systems. Example:

```
(simple mount table)
# mount /dev/sd0h /usr; cd /usr; restore -rs 2 -f /dev/rst0

(more advanced mount table)
# mount /dev/sd0d /var; cd /var; restore -rs 2 -f /dev/rst0
# mount /dev/sd0e /home; cd /home; restore -rs 3 -f /dev/rst0
# mount /dev/sd0h /usr; cd /usr; restore -rs 4 -f /dev/rst0
```

You could use "**restore rvsf**" instead of just rsf to view names of objects as they are extracted from the dump set.

Finally after you finish restoring all your other file systems to disk, reboot into multiuser mode. If everything went as planned your system will be back to the state it was in as of your most recent back up tape and ready to use again.

14.10 - Монтирование образов дисков в OpenBSD

Для подключения(монтирования) образа диска (ISO образ, образ диска созданный dd, т..п) в OpenBSD вы должны сконфигурировать устройство vnd(4). Для примера , если у вас есть ISO образ находящийся в */tmp/ISO.image*, вам необходимо проделать следующие действия для подключения образа.

```
# vnconfig svnd0 /tmp/ISO.image
# mount -t cd9660 /dev/svnd0c /mnt
```

Примечание, что это соответствует только для ISO-9660 образов, такие как используются в CD и DVD, вам необходимо использовать тип диска *cd9660* когда подключаете их. Это так, но не соответствует когда тип, например: нужно использовать тип *ext2fs* когда подключается образ диска Linux.

Для отключения(размонтирования) образа используйте следующие команды.

```
# umount /mnt
# vnconfig -u svnd0
```

Для большей информации, обратитесь к странице руководства vnconfig(8).

14.11 - Помогите! У меня ошибки с IDE DMA!

Передача данных по IDE с использованием DMA поддерживается pciide(4) , однако могут быть проблемы с некоторыми комбинациями оборудования. До недавнего времени большинство "основных" операционных систем имели выключенную поддержку DMA IDE по умолчанию из-за некачественной работы оборудования. Теперь это оборудование используется с OpenBSD.

OpenBSD "агрессивна" и пытается использовать, насколько это возможно, "наивысший" режим DMA Mode. Это может приводить к ошибкам на некоторых конфигурациях оборудования, контроллерах материнских плат, "глючных" приводах и/или электропомехах на кабелях. К счастью, режим Ultra-DMA проверяет контрольные суммы и зищищает данные. Когда Ultra-DMA CRC ошибочны OpenBSD выдаст сообщение об ошибке и повторит операцию.

```
wd2a:  aborted command, interface CRC error reading fsbn 64 of 64-79
(wd2 bn 127; cn 0 tn 2 sn 1), retrying
```

После повторения ошибки несколько раз OpenBSD изменит режим Ultra-DMA на более медленый (предполагаемо более надежный). Если режим Ultra-DMA mode 0, то устройство будет использоваться в режиме PIO.

Ошибки UDMA чаще всего вызываются некачественными кабелями. Cable problems should usually be the first suspect if you get many DMA errors or unexpectedly low DMA performance. It is also a bad idea to put the CD-ROM on the same channel with a hard disk.

If replacing cables does not resolve the problem and OpenBSD does not successfully downgrade, or the process causes your machine to lock hard, or causes excessive messages on the console and in the logs, you may wish to force the system to use a lower level of DMA or UDMA by default. This can be done by using UKC or config(8) to change the flags on the wd(4) device.

14.13 - Возможности RAID в OpenBSD

RAID (Redundant Array of Inexpensive Disks) gives an opportunity to use multiple drives to give better performance, capacity and/or redundancy than one can get out of a single drive alone. While a full discussion of the benefits and risks of RAID are outside the scope of this article, there are a couple points that are important to make here:

- RAID has nothing to do with backup.
- By itself, RAID will not eliminate down-time.

If this is new information to you, this is not a good starting point for your exploration of RAID.
Software Options
OpenBSD includes RAIDframe, a software RAID solution. Documentation for it can be found in the following places:

- Disk Optimization, RAID

- RAIDframe Homepage
- man page for raidctl(8)
- man page for raid(4)

The root partition can be directly mirrored by OpenBSD using the "Autoconfiguration" option of RAIDframe.

OpenBSD 3.7-stable and later also includes mirroring as a feature of the ccd(4) driver. This system is built into the GENERIC kernel and is in the bsd.rd kernel of some platforms (amd64, hppa, hppa64, i386), so it can be much easier to use, though it has some limitations regarding rebuilding the array. See:

- ccd(4) man page
- ccdconfig(8) man page

Hardware Options

Many OpenBSD platforms include support for various hardware RAID products. The options vary by platform, see the appropriate hardware support page (listed here).

Another option available for many platforms is one of the many products which make multiple drives act as a single IDE or SCSI drive, and are then plugged into a standard IDE or SCSI adapter. These devices can work on virtually any hardware platform that supports either SCSI or IDE.

Some manufacturers of these products:

- Arco
- Accusys
- Maxtronic
- Infortrend

(Note: these are just products that OpenBSD users have reported using -- this is not any kind of endorsement, nor is it an exhaustive list.)

Non-Options

An often asked question on the mail lists is "Are the low-cost IDE or SATA RAID controllers (such as those using Highpoint, Promise or Adaptec HostRAID chips) supported?". The answer is "No". These cards and chips are not true hardware RAID controllers, but rather BIOS-assisted boot of a software RAID. As OpenBSD already supports software RAID in a hardware-independent way, there isn't much desire among the OpenBSD developers to implement special support for these cards.

Almost all on-board SATA or IDE "RAID" controllers are this software-based style, and will typically work fine as a SATA or IDE controller using the standard IDE driver (pciide(4)), but are not going to work as a hardware RAID system on OpenBSD.

Почему df(1) сообщает мне об использовании более 100% дискового пространства?

Люди иногда удивляются, когда видят отрицательное место, доступное на диске, или когда df(1) показывает более 100% использованного дискового пространства.

Когда с помощью newfs(8) создается раздел, некоторое доступное место резервируется. Это позволяет избежать ошибок, когда диск заполняется и помогает свести фрагментацию диска к минимуму. По умолчанию, в качестве резерва используется 5% от размера диска, поэтому, если пользователь непредусмотрительно заполняет диск, то он может видеть, что он заполняется до 105%.

Если использование 5% диска в качестве резерва для вас неприемлемо, то вы можете сменить его с помощью команды tunefs(8).

14.15 - Восстановление разделов после удаления таблицы разделов

Если вы испортили таблицу разделов вашего диска, существует несколько вариантов действий, которые можно предпринять для ее восстановления.

Во-первых, паникуйте. Вы обычно так и поступаете. Только не делайте ничего глупого. Паникуйте подальше от вашей машины. Потом расслабьтесь и посмотрите, не смогут ли описанные ниже шаги помочь вам.

Копия таблицы разделов каждого диска хранится в /var/backups как часть ежедневной работы системы. Если вы все ещё имеете раздел var, можете просто просмотреть этот вывод и поместить его в disklabel.

В случае, когда вы не можете увидеть этот раздел, существуют ещё две возможности. Fix enough of the disc so you can see it, or fix enough of the disc so that you can get your data off. Depending on what happened, one or other of those may be preferable (with dying discs you want the data first, with sloppy fingers you can just have the label)

The first tool you need is scan_ffs(8) (note the underscore, it isn't called "scanffs"). scan_ffs(8) will look through a disc, and try and find partitions and also tell you what information it finds about them. You can use this information to recreate the disklabel. If you just want /var back, you can recreate the partition for /var, and then recover the backed up label and add the rest from that.

disklabel(8) will update both the kernel's understanding of the disklabel, and then attempt to write the label to disk. Therefore, even if area of the disk containing the disklabel is unreadable, you will be able to mount(8) it until the next reboot.

14.16 - Возможен ли доступ к другим файловым системам, кроме FFS?

Да. В OpenBSD включена поддержка следующих файловых систем: ext2 (Linux), ISO9660 and UDF (CD-ROM, DVD media), FAT (MS-DOS и Windows), NFS, NTFS (Windows), AmigaDOS. Некоторые из них ограничены возможностью доступа "только на чтения". Заметим, что UFS2 FreeBSD не поддерживается.

Сейчас мы дадим вам общее понятие о том, как использовать эти файловые системы в OpenBSD. Чтобы использовать файловую систему, она должна быть примонтирована. За получением информации об опциях монтирования и других деталей, пожалуйста ознакомьтесь с руководством mount(8) и руководствами команд файловых систем, которые вы будете монтировать, например, mount_msdos, mount_ext2fs, ...

Во-первых, вы должны знать, на каком устройстве расположена ваша файловая система. Это может быть просто ваш первый жесткий диск, wd0 или sd0. Информацию обо всех найденных и сконфигурированных системой устройствах можно найти в выводе команды dmesg(1): название устройства и следующее за ним описание этого устройства. Например, нахождение системой моего первого CD-ROM сопровождается следующими строками:

```
cd0 at scsibus0 targ 0 lun 0: <COMPAQ, DVD-ROM LTD163, GQH3> SCSI0 5/cdrom removable
```

Для получения более короткого списка доступных дисков, используйте sysctl(8). Команда

```
# sysctl hw.disknames
```

покажет все текущие диски, известные вашей системе, например:

```
hw.disknames=cd0,cd1,wd0,fd0,cd2
```

Сейчас самое время узнать, какие разделы существуют на устройстве и какие файловые системы используются на этих разделах. Здесь мы проверим наше устройство, используя disklabel(8). Disklabel содержит список разделов. Максимальное количество разделов – 16. Раздел "c" всегда показывает все устройство. Разделы a-b и d-p используются OpenBSD. Разделы i-p могут быть автоматически установлены как файловые системы других операционных систем. В данном случае, я просматриваю таблицу разделов моего жесткого диска, который содержит несколько разных файловых систем.

NOTE: OpenBSD was installed after the other operating systems on this system, and during the install a disklabel containing partitions for the native as well as the foreign filesystems was installed on the disk. However, if you install foreign filesystems after the OpenBSD disklabel was already installed on the disk, you need to add or modify them manually afterwards. This will be explained in FAQ14.

```
# disklabel wd0

# using MBR partition 2: type A6 off 20338290 (0x1365672) size 29318625 (0x1bf5de1)
# /dev/rwd0c:
type: ESDI
disk: ESDI/IDE disk
label: ST340016A
flags:
```

```
bytes/sector: 512
sectors/track: 63
tracks/cylinder: 16
sectors/cylinder: 1008
cylinders: 16383
total sectors: 78165360
rpm: 3600
interleave: 1
trackskew: 0
cylinderskew: 0
headswitch: 0           # microseconds
track-to-track seek: 0  # microseconds
drivedata: 0

16 partitions:
#             size        offset  fstype [fsize bsize  cpg]
  a:        408366      20338290  4.2BSD   2048 16384   16 # Cyl 20176*- 20581
  b:       1638000      20746656    swap                    # Cyl 20582 - 22206
  c:      78165360             0  unused      0     0      # Cyl     0 - 77544
  d:       4194288      22384656  4.2BSD   2048 16384   16 # Cyl 22207 - 26367
  e:        409248      26578944  4.2BSD   2048 16384   16 # Cyl 26368 - 26773
  f:      10486224      26988192  4.2BSD   2048 16384   16 # Cyl 26774 - 37176
  g:      12182499      37474416  4.2BSD   2048 16384   16 # Cyl 37177 - 49262*
  i:         64197            63 unknown                   # Cyl     0*-    63*
  j:      20274030         64260 unknown                   # Cyl    63*- 20176*
  k:       1975932      49656978   MSDOS                   # Cyl 49262*- 51223*
  l:       3919797      51632973 unknown                   # Cyl 51223*- 55111*
  m:       2939832      55552833  ext2fs                   # Cyl 55111*- 58028*
  n:       5879727      58492728  ext2fs                   # Cyl 58028*- 63861*
  o:      13783707      64372518  ext2fs                   # Cyl 63861*- 77535*
```

As can be seen in the above output, the OpenBSD partitions are listed first. Next to them are a number of ext2 partitions and one MSDOS partition, as well as a few 'unknown' partitions. On i386 and amd64 systems, you can usually find out more about those using the fdisk(8) utility. For the curious reader: partition i is a maintenance partition created by the vendor, partition j is a NTFS partition and partition l is a Linux swap partition.

Once you have determined which partition it is you want to use, you can move to the final step: mounting the filesystem contained in it. Most filesystems are supported in the GENERIC kernel: just have a look at the kernel configuration file, located in the */usr/src/sys/arch/**<arch>**/conf* directory. However, some are not, e.g. the NTFS support is experimental and therefore not included in GENERIC. If you want to use one of the filesystems not supported in GENERIC, you will need to build a custom kernel.

When you have gathered the information needed as mentioned above, it is time to mount the filesystem. Let's assume a directory /mnt/otherfs exists, which we will use as a mount point where we will mount the desired filesystem. In this example, we will mount the ext2 filesystem in partition m:

```
# mount -t ext2fs /dev/wd0m /mnt/otherfs
```

If you plan to use this filesystem regularly, you may save yourself some time by inserting a line for it in /etc/fstab, for example something like:

```
/dev/wd0m /mnt/otherfs ext2fs rw,noauto,nodev,nosuid 0 0
```

Notice the 0 values in the fifth and sixth field. This means we do not require the filesystem to be dumped, and checked using fsck. Generally, those are things you want to have handled by the native operating system associated with the filesystem.

14.16.1 - Разделы не отображаются в disklabel! Что делать?

If you install foreign filesystems on your system (often the result of adding a new operating system) after you have already installed OpenBSD, a disklabel will already be present, and it will not be updated automatically to contain the new foreign filesystem partitions. If you wish to use them, you need to add or modify these partitions manually using disklabel(8).

As an example, I have modified one of my existing ext2 partitions: using Linux's fdisk program, I've reduced the size of the 'o' partition (see disklabel output above) to 1G. We will be able to recognize it easily by its starting position (offset: 64372518) and size (13783707). Note that these values are sector numbers, and that using sector numbers (not megabytes or any other measure) is the most exact and safest way of reading this information.

Before the change, the partition looked like this using OpenBSD's fdisk(8) utility (leaving only relevant output):

```
# fdisk wd0
. . .
Offset: 64372455        Signature: 0xAA55
         Starting       Ending       LBA Info:
 #: id    C   H  S -    C   H  S [       start:      size   ]
------------------------------------------------------------------------
 0: 83 4007   1  1 - 4864 254 63 [    64372518:    13783707 ] Linux files*
. . .
```

As you can see, the starting position and size are exactly those reported by disklabel(8) earlier. (Dont' be confused by the value indicated by "Offset": it is referring to the starting position of the extended partition in which the ext2 partition is contained.)

After changing the partition's size from Linux, it looks like this:

```
# fdisk wd0
. . .
Offset: 64372455        Signature: 0xAA55
         Starting       Ending       LBA Info:
 #: id    C   H  S -    C   H  S [       start:      size   ]
```

```
-------------------------------------------------------------------------
 0: 83 4007   1  1 - 4137 254 63 [    64372518:     2104452 ] Linux files*
. . .
```

Now this needs to be changed using disklabel(8). For instance, you can issue disklabel -e wd0, which will invoke an editor specified by the EDITOR environment variable (default is vi). Within the editor, change the last line of the disklabel to match the new size:

```
o:       2104452      64372518  ext2fs
```

Save the disklabel to disk when finished. Now that the disklabel is up to date again, you should be able to mount your partitions as described above.

You can follow a very similar procedure to add new partitions.

14.7 - Могу ли я использовать флэш-память в OpenBSD?

Убычно устройство памяти нормально распознается при включении в порт компьютера. Через незначительное время на системной консоли ядро выведет несколько строк с информацией. В данном случае, когда я вставил USB флэшпамять в разъем, в моей консоли появилось следующее:

```
umass0 at uhub1 port 1 configuration 1 interface 0
umass0: LEXR PLUG DRIVE LEXR PLUG DRIVE, rev 1.10/0.01, addr 2
umass0: using SCSI over Bulk-Only
scsibus2 at umass0: 2 targets
sd0 at scsibus2 targ 1 lun 0: <LEXAR, DIGITAL FILM, /W1.> SCSI2 0/direct removable
sd0: 123MB, 123 cyl, 64 head, 32 sec, 512 bytes/sec, 251904 sec total
```

Эти строки сообщают о том, что мой драйвер umass(4) (USB mass storage) подключил устройство памяти, исполоьзуя систему SCSI. Последние две строки содержат наиболее важную информацию: они сообщают о device node и размере доступной памяти. Если вы не успели просмотреть эти строки, их можно просмотреть с помощью dmesg(1). Сообщение о геометрии CHS является фиктивным, это следствие того, что флэшпамять сейчас рассматривается как SCSI диск.

Мы рассмотрим оба возможных случая.

Устройство новое/чистое и вы планируете использовть его только с OpenBSD

Вам необходимо инициализировать disklabel на этом устройстве, и создать как минимум один раздел (partition). Пожалуйста прочтите руководство Using OpenBSD's disklabel and the disklabel(8) для выяснения подробностей этой процедуры.

В этом примере я создам один раздел с файловой системой FFS:

```
# newfs sd0a
Warning: inode blocks/cyl group (125) >= data blocks (62) in last
    cylinder group. This implies 1984 sector(s) cannot be allocated.
/dev/rsd0a:     249856 sectors in 122 cylinders of 64 tracks, 32 sectors
        122.0MB in 1 cyl groups (122 c/g, 122.00MB/g, 15488 i/g)
super-block backups (for fsck -b #) at:
 32,
```

Смонтируем созданную файловую систему в разделе на /mnt/flashmem. Предварительно создайте точку монтирования если её еще нет.

```
# mkdir /mnt/flashmem
# mount /dev/sd0a /mnt/flashmem
```

Вы получили флеш диск от кого-то с кем хотели-бы обменяться данными

Есть существенная вероятность, что другие пользователи не не пользуется OpenBSD, поэтому на диске может оказаться чуждая файловая система. Поэтому, первое, что необходимо сделать - это узнать какие разделы есть на диске, как описано в соответствующем разделе FAQ 14.

```
# disklabel sd0
# /dev/rsd0c:
type: SCSI
disk: SCSI disk
label: DIGITAL FILM
flags:
bytes/sector: 512
sectors/track: 32
tracks/cylinder: 64
sectors/cylinder: 2048
cylinders: 123
total sectors: 251904
rpm: 3600
interleave: 1
trackskew: 0
cylinderskew: 0
headswitch: 0           # microseconds
track-to-track seek: 0  # microseconds
drivedata: 0

16 partitions:
#             size        offset  fstype [fsize bsize  cpg]
```

```
  c:        251904              0  unused      0     0      # Cyl     0 -   122
  i:        250592             32   MSDOS                   # Cyl     0*-   122*
```

Как видно в выводе disklabel, на устройстве только один раздел *i*, содержащий файловую систему FAT созданную на Windows машине. Как обычно раздел с описывает все устройство.

Смонтируем файловую систему раздела i в /mnt/flashmem.

```
# mount -t msdos /dev/sd0i /mnt/flashmem
```

Теперь можете начинать пользоваться им как любым другим диском.

ВНИМАНИЕ: Всегда отмонтируйте файловую систему **перед отключением** устройства памяти. Если вы это не сделаете, файловая система может быть повреждена, и вы можете потеряеть свои данные.

После извлечения устройства памяти из разъема ядро выведет следующие несколько сток:

```
umass0: at uhub1 port 1 (addr 2) disconnected
sd0 detached
scsibus2 detached
umass0 detached
```

14.8 - Оптимизация производительности дисковой подсистемы

Производительность диска является важной характеристикой, влияющей на общую производительность компьютера. Ее важность значительно возрастает, когда компьютер выполняет фунции многопользоватеьской среды (пользователи всех мастей, начиная с тех, кто заходит через консоль, заканчивая теми, кто видит компьютер как файловый или веб-сервер). Хранилище данных постоянно требует внимания, особенно когда свободное место на разделах(диска) заканчивается или когда диск выходит из строя. OpenBSD имеет несколько вариантов способных увеличить скорость ваших дисков и повысить отказоустойчивость.

CCD - Драйвер "продолжающих" дисков.
RAID
Soft Updates
Size of the namei() cache

14.8.1 - CCD

Первый вариант - это использование ccd(4), драйвера объединенного диска. Это позволяет объединить несколько разделов в один виртуальный диск, таким же образом можно объединить и целые диски в один виртуальный диск. Эта концепция подобна той, которая используется в LVM (Управление логическими

томами/разделами). LVM является частью многих коммерческих Unix'ов.

В случае использования GENERIC ядра, ccd уже доступен (в файле /usr/src/sys/conf/GENERIC). Если у Вас собственное(customized) ядро, возможно понадобится включить соответсвующую опцию в конфигурацию ядра. В любом случае, строка приведенная ниже должна присутствовать в файле конфигурации:

```
pseudo-device   ccd     4       # concatenated disk devices
```

Приведенный выше пример дает до четырех ccd устройств (виртуальных дисков). Теперь необходимо определить какие разделы на реальных дисках Вы хотите выделить для ccd. Используйте disklabel для маркировки этих разделов как 'ccd' (type 'ccd'). На некоторых архитектурах disklabel может не разрешить сделать это. В этом случае отметьте их как 'ffs' (type 'ffs').

Если вы используете ccd для большей производительности, получить наибольшую производительность можно при использовании дисков одной и той модели, с одинаковыми настройками disklabel.

Отредактируйте примерно так /etc/ccd.conf: (для дополнительной информации о конфигурированиии ccd обратитесь к ccdconfig(8))

```
# Configuration file for concatenated disk devices
#
# ccd   ileave  flags   component devices
ccd0   16      none    /dev/sd2e /dev/sd3e
```

To make your changes take effect, run

```
# ccdconfig -C
```

Пока /etc/ccd.conf существует, ccd будет автоматически конфигурироваться после загрузки. Теперь вы имеете новый диск, ccd0, состоящий из /dev/sd2e и /dev/sd3e. Воспользуйтесь disklabel как вам необходимо как на обычном диске. Как всегда, не используйте раздел 'c' как обычный раздел. Создайте используемые разделы, отступив хотя бы на один цилиндр от начала диска.

14.18.2 - RAID

Еще одним вариантом является raid(4), который с помощью raidctl(8) управляет устройствами raid. RAID OpenBSD основан на NetBSD port Greg-a Oster-a CMU RAIDframe. OpenBSD поддерживает RAID 0-го, 1-го, 4-го и 5-го уровня.

Как raid, так и ccd должны поддерживаться ядром. В отличие от ccd, поддержка RAID не включена GENERIC, так что ядро должно быть скомпилировано с его поддержкой (поддержка RAID на 500K увеличивает размер ядра i386).

```
pseudo-device   raid   4       # RAIDframe disk device
```

Более подробно raid(4) и raidctl(8). Имеется много параметров и опций, подробное объяснение выходит за рамки этого документа.

14.18.3 - Soft updates

Еще одним инструментом для повышения быстродействия - softupdates. Одной из самых медленных операций в традиционных файловых системах BSD является обновление метаданных (что происходит, в частности, когда вы создаете или удаляете каталоги и файлы). Softupdate пытаются произвести обновление метаданных в ОЗУ вместо операций непосредственно каждый раз на диске. Другим эффектом является то, что метаданные на диске всегда полная, хотя и не всегда актуальная. Поэтому после аварии система не должна требовать выполнения fsck(8) при загрузке, просто fsck запускается в фоне, внося изменения в метаданные в ОЗУ (примерно как softupdates). В результате перезагрузка сервера происходит быстрее, вам не придется ждать, когда отработает fsck! (OpenBSD пока не имеет этой функции). Больше о softupdates можно узнать в разделе Softupdates настоящего руководства.

14.18.4 - Размер кэша namei()

The name-to-inode translation (a.k.a., namei()) cache controls the speed of pathname to inode(5) translation. A reasonable way to derive a value for the cache, should a large number of namei() cache misses be noticed with a tool such as systat(1), is to examine the system's current computed value with sysctl(8), (which calls this parameter "kern.maxvnodes") and to increase this value until either the namei() cache hit rate improves or it is determined that the system does not benefit substantially from an increase in the size of the namei() cache. After the value has been determined, you can set it at system startup time with sysctl.conf(5).

14.19 - Почему не используется асинхронное монтирование?

Вопрос: "I simply do "mount -u -o async /" which makes one package I use (which insists on touching a few hundred things from time to time) usable. Why is async mounting frowned upon and not on by default (as it is in some other unixen)? Isn't it a much simpler, and therefore, a safer way of improving performance in some applications?"

Ответ: "Асинхронное монтирование действительно быстрее работает, чем синхронное, но оно и менее безопасно. Что будет в случае аварии электропитания? Выхода из строя аппаратного обеспечения? Ускорение скорости не должно сопровождаться снижением надежности и стабильности работы. Читаем страницу руководства mount(8)."

```
async   All I/O to the file system should be done asynchronously.
        This is a dangerous flag to set since it does not guaran-
```

tee to keep a consistent file system structure on the disk. You should not use this flag unless you are prepared to recreate the file system should your system crash. The most common use of this flag is to speed up restore(8) where it can give a factor of two speed increase.

(async Все операции ввода вывода файловой системы будут асинхронными. Это опасная опция, ее установка не гарантирует сохранение файловой структуры на диске в целости. Вы не должны использовать эту опцию, если заранее не подготовились к восстановлению системы после аварии. Чаще всего эту опцию используют при воссстановлении системы, она может увеличить скорость восстановления в два раза.)

С другой стороны, имеет смысл использовать асинхронное монтирование в случаях работы с временными данными, потеря которых в случае аварии является некритичной. В этом случае, mfs(8) partitions are mounted asynchronously, as they will get wiped and recreated on a reboot anyway.

Перевод соответствует $OpenBSD: faq14.html,v 1.172 2008/05/20 09:02:34 otto Exp $